Matching Pursuit Low-Bit Rate Video Coding with Dictionary Optimized by Shape-Gain Vector Quantizer

Yao-Tang Chou‡, Wen-Liang Hwang† and Chung-Lin Huang‡
† Institute of Information Science, Academia Sinica, Taiwan
‡ Electrical Engineering Department, National Tsing Hua University

Abstract

We show that the techniques of the shape-gain product vector quantizer(VQ) can be used to optimize the dictionary of the matching pursuit codec and produce good results. We start with an initial dictionary and obtain a dictionary that is well adapted to the training sequences. The performance is evaluated based on a comparison of our method with other methods.

1 Introduction

There has been increasing demand for video communications using low bit rate channels (with bit rates ranging from 10 kbit/s to 60 kbit/s) in wireless channels or telephone lines, and for video services. Current standards, such as H.261, H.263, MPEG-1, and MPEG-2, are not quite suitable when operated at low bit rates (with rates below 20 kbit/s) because these systems are based on the motion compensated discrete cosine transform (DCT). When motion residuals are coded at low bit rates, the choice of basis function is extremely important because a coder must use only a limited number of coarsely quantized coefficients for the residuals. If individual elements of the basis set are not well matched to inner structures in the residuals, then the basis functions will become visible in the reconstructed images. This explains the blocky artifacts produced by block-DCT based coders in smoothly varying image regions and high-frequency noise patterns around the edges of moving objects.

Several results [1][2][3][11][10][9] have indicated that better visual quality at low bit rates can be obtained if block-DCT residual coders are replaced with matching pursuit-based coders. In matching pursuit-based coders, a larger and more flexible set of basis functions, called a dictionary, is employed to encode the motion residual frames. Thus, better matching of the residuals results in fewer artifacts than is the case with the block-DCT based coder as the bit rate decrease. The matching pursuit algorithm is an iterative process, and it requires the inner product of the residuals with each basis (presumably a very large basis set) at each iteration. In practice, concepts similar to those used in VQ are usually adopted to speed up the algorithm [4]. In Neff and Zakhor's matching pursuit codec[1], an efficient subset of a large basis set is selected as a dictionary, according to the residual patterns in some video sequences, to reduce the number of inner products at each iteration. However, the method is simply a selection of an efficient subset from these sequences, and the subset is not necessarily optimized to the sequences. From this reason, we introduce optimization techniques for the dictionary of a matching pursuit codec. Our algorithm starts with an efficient basis set and modifies the basis set to obtain the training sequences using the jointly optimized algorithm, popular in shape-gain product VQ design. The resultant basis set is our dictionary, which is optimized to the training sequences. The advantage of this method is that once an optimized dictionary is found from a subset of a class of video sequences (sequences with "similar" properties), the dictionary can then be used to play a sequence in the class and may be able to obtain good performance.

Our method differs from Neff and Zakhor's procedure. In their method, an efficient subset is selected from a large dictionary while in our method, we begin with a small, efficient subset of a large dictionary, modify the elements in the set through VQ, and end up with an optimized basis set which does not necessarily belong to the large dictionary. Fig. 1 compares the block diagrams of both methods.

The implementation of our matching pursuit video codec mostly follows that proposed in [1]. A detailed implementation of our video codec will be given in the following sections. Comparing the PSNR performance, in experiments on some video sequences containing mostly static or slow motion foreground objects, we obtained an average 0.5dB PSNR gain per frame using our basis set compared to that obtained using the basis set of Neff and Zakhor.

In Section 2, we will dicuss our matching pursuit codec. In Section 3, the shape-gain product VQ techniques for optimizing the dictionary of a matching pursuit codec will be introduced. In Section 4, experimental results will be given. Conclusions will be drawn in Section 5.

2 Matching Pursuit Video Coding

We will first review the matching pursuit theory proposed by Mallat and Zhang[4]. Then, we will discuss the implementation of our matching pursuit video codec. Our implementation is close to that proposed in [1].

2.1 Matching Pursuit Theory

The matching pursuit algorithm was proposed to identify efficiently any signal structure from its expansion coefficients using an over-complete bases. In the following, for the sake of simplicity, we will only review the one-dimensional matching pursuit algorithm. One can extend it to a similar algorithm in two dimensions for image applications. The matching pursuit algorithm represents a signal using linear expansion of the signal with respect to a dictionary which is a collection of many complete bases; for example, both the Fourier and wavelet bases can belong to the dictionary, and the dictionary is dense for all $L^2(R)$ functions. The following matching pursuit algorithm was proposed by Mallat and Zhang:

Let $g_{\gamma 0}(t) \in D$, the dictionary. The function $f(t)$ can be decomposed into

$$f(t) = \langle f, g_{\gamma 0} \rangle g_{\gamma 0}(t) + Rf(t),$$

where $Rf(t)$ is the residual function after approximating $f(t)$ in the direction of $g_{\gamma 0}(t)$. To minimize $\|Rf(t)\|$, $g_{\gamma 0}(t)$ is chosen such that $|\langle f, g_{\gamma 0} \rangle|$ is maximum in D. The matching pursuit algorithm then decomposes the residue $Rf(t)$ by projecting it on a basis function of D that best matches $Rf(t)$, as was done for $f(t)$. This procedure is repeated each time on the following residue that is obtained. It was showed that when the dictionary is dense in $L^2(R)$,

$$f(t) = \sum_{n=0}^{+\infty} \langle R^n f, g_{\gamma_n} \rangle g_{\gamma_n}(t),$$

and

$$\|f(t)\|^2 = \sum_{n=0}^{+\infty} |\langle R^n f, g_{\gamma_n} \rangle|^2.$$

The most common basis of a matching pursuit dictionary was proposed by B. Torresani[8], which is a family generated by scaling, translating, and modulating a single function $g(t)$:

$$g_\gamma(t) = \frac{1}{\sqrt{s}} g(\frac{t-u}{s}) e^{i\xi(t-u)},$$

where we assume that $\|g_\gamma(t)\| = 1$ and scale $s > 0$, ξ is frequency modulation, u is translation, and $\gamma = (s, u, \xi)$.

The matching pursuit algorithm provides extremely flexible signal representation since dictionary is done arbitrary. With such a flexible choice of basis functions, the matching pursuit algorithm can be applied to compact coding problems. In these applications, it is important to use strategies that will adapt the dictionary to the class of signals to be decomposed. Optimizing a dictionary is equivalent to finding the important inner structures of training signals.

2.2 Matching Pursuit Video Codec

Two matching pursuit video codecs will be reviewed: (1) Banham and Braillean[3] proposed a motion residual encoding method which is a hybrid of the matching pursuit and block-DCT methods. Selection in which method, obtained using matching pursuit or using block-DCT and used to encode the current motion residual, is based on a comparison of the current residual frame's energy with the average energy in the previous residual frames; (2) Neff and Zakhor[1] first found an efficient subset of the separable Gabor dictionary with training; then, the motion residuals of a testing sequence were decomposed against the bases in the subset. They demonstrated that their matching pursuit video codec outperforms the H.263[13] standard in both the peak signal-to-noise ratio (PSNR) and visual quality. The major contribution of this paper lies in improving their work by employing the shape-gain product VQ techniques to optimize the dictionary of a matching pursuit video codec. Detailed discussion of our implementation will be given, which in many places similar to that of Neff and Zakhor [1].

2.2.1 System Description

The system presented in this section is a hybrid motion-compensated video codec in which motion residual is coded using matching pursuits. Simplified block diagrams of the encoder and decoder are shown in Fig. 2.

2.2.2 Motion Compensation

The system is operated on quarter common intermediate format (QCIF) images, which consist of 176×144 luminance and 88×72 chrominance components. The luminance portion of the image is divided into 16×16 blocks, and to each block is assigned either a single motion vector, a set of four motion vectors, or a set of intra block parameters. The motion model used by our matching pursuit system is identical to the advanced prediction model used in TMN[12], which is a specific implementation of H.263. For more details on these methods, one can consult the H.263[13] and TMN[12] documents.

2.2.3 Matching Pursuit Residual Coding

After the motion prediction image is formed, it is subtracted from the original image to produce the motion residual. This residual image is coded using a discrete two-dimensional matching pursuit algorithm with a properly chosen dictionary. Neff and Zakhor retained a reduced subset of the 2-D separable Gabor dictionary as their dictionary. Table 1 gives the parameters of their reduced 1-D basis set. Here, the parameter $\alpha = (s, \xi, \phi)$ is a triplet consisting, respectively, of a positive scale, a modulation frequency, and a phase shift. The constant K_α is chosen such that the following 1-D Gabor sequence is of unit norm:

$$g_\alpha(i) = K_\alpha \cdot g(\frac{i - \frac{N}{2} + 1}{s}) \cdot cos(\frac{2\pi\xi(i - \frac{N}{2} + 1)}{16} + \phi)$$
$$i \in \{0, 1, \cdots, N-1\}.$$

Neff and Zakhor truncated their basis functions to various lengths, mostly smaller than 35×35. We have truncated all of them to 35×35, and the resulting functions are formed as an image for the poupose of visualization. A visualization of their basis set is shown in the middle part of Fig. 5,

Finding an Atom

To further speed up the encoding procedure, we have adopted a method similar to that proposed in [1], where the inner product search of the matching pursuit algorithm was restricted around the image patch with the largest averaged energy. The search procedure is shown in Fig. 3. In (a), a partition of the motion residual frame is shown. The block of the largest averaged energy in the partitioned image is highlighted in (b). (c) shows the portion of the image around the highlighted block in (b) where the matching pursuit algorithm is applied. The highlighted image patch in (d) indicates the place in (c) where the first atom is found. We use the notation F to denote this procedure consisting of (a),(b), (c), and (d):

$$F : I(x,y) \rightarrow P(x,y), \tag{1}$$

where $I(x,y)$ is the image and $P(x,y)$ is the image patch shown in (d). The image patch $P(x,y)$ is the place where the first matching pursuit atom on $I(x,y)$ is obtained by F. Its size is equal to that of the truncated basis function. An atom is a set of five parameters: the largest inner product, the corresponding horizontal and vertical indices(*e.g.,* k in Table 1), and the locations of the patch $P(x,y)$ (indicated by solid dots in (d)).

Coding Atom Parameters and Buffer Regularization

In order to compare the performance of our matching pursuit codec with that of the TMN H.263 codec, we need to synchronize the rate control of the two systems: The TMN encoder is first run with a target bit rate of 24 kbit/s and a frame rate of 10 frames/s, and a record is kept of the number of bits used by the TMN to represent each residual frame. The matching pursuit system then uses this record to encode the same subsets of the original frames with approximately the same number of bits for each residual frame. We will explain our implementation further in Section 4.

3 Dictionary Optimization using Shape-Gain VQ

We employ the VQ techniques to optimize the matching pursuit dictionary. Before describing our method, we will briefly review the shape-gain product VQ since it is used in practice to design optimal VQ codebooks for a class of sequences.

3.1 Shape-Gain Vector Quantizer

The purpose of a vector quantizer(VQ) is to approximate a random input vector to its close representative vector (called codeword); then, the index of the codeword is effectively encoded and transmitted[5][16][14]. The decoder can easily, with the help of codebooks, recover the corresponding codeword from an index. However, in high dimensional space, the vector space is too big to be managed efficiently, in terms of both computational storage and time, by a VQ. Therefore, the VQ is usually constrained. The constrained VQ increases the efficiency in terms of time and storage, but the accuracy of the VQ is degraded. The shape-gain product VQ is one of the constrained VQs. It has been successfully used to code the waveforms as well as the parameters of speech signals[15][6][7]. The shape-gain product VQ assumes that the element X in the training random variable $\underline{X}$ has a product code structure:

$$X = gS,$$

where g is a scale (called gain) and S is a normalized vector (called shape). For a detailed description of the codec structure and the optimized codebook design for a shape-gain product VQ, the reader can refer to [16][15].

3.2 Dictionary Optimization

Before presenting our VQ for a matching pursuit dictionary, we should describe the training random variable $\underline{X}$ for our application. Let the original dictionary be $\{(g_\alpha g_\beta)(x,y)\}$, where $\|(g_\alpha g_\beta)(x,y)\| = 1$ and the size of the basis function in the dictionary is truncated to $b \times b$. Let $f_k(x,y)$ be the k-th motion residual frame in the training sequence, and let the l-th matching pursuit residual of the k-th frame be $R^l f_k(x,y)$. The matching pursuit decomposition, described in Subsection 2.2.3, of $f_k(x,y)$ up to $N(k)$ atoms is

$$f_k(x,y) = \sum_{n=0}^{N(k)-1} \langle F(R^n f_k), (g_{\alpha_n} g_{\beta_n}) \rangle (g_{\alpha_n} g_{\beta_n})(x,y) + R^{N(k)} f_k(x,y), \qquad (2)$$

where F is the mapping given in Eq. (1). F uses the image patch (called the F-patch for convenience) of size $b \times b$, equal to the truncated basis function, from which the first atom in the matching pursuit residual image $R^n f_k(x,y)$ is selected. This image patch contains important inner structures of the residual images. Our training random variable $\underline{X}$ is the collection of F-patches from all the matching pursuit residuals in our training sequence:

$$\underline{X} = \{F(R^n f_k)(x,y); k = 1, 2, \cdots; n = 0, 1, \cdots, N(k) - 1\}.$$

The vector space of our random variable $\underline{X}$ is of dimension $b \times b$, which is the size of a basis function in our dictionary. A basis function S is a vector mapped to the points s on the radial 1 sphere in a vector space. Similarly, an F-patch P in the random variable $\underline{X}$ is mapped to a point, say p, in the vector space. The inner product of the image patch P and the basis function S is equivalent to the projection of p in the direction of s. In our implementation, we adopt a basis function of size 35×35; thus, we require a VQ design in a vector space with a dimension of 35×35, which is too big to manage efficiently in practice. Thus, our VQ design must be constrained. We will next consider the task of codebook design for shape-gain product VQ. Since the basis functions are of norm 1, they form the shape part, and the inner product values are the gain part of our VQ.

Our objective is to find the shape and gain codebooks that minimize the average distortion incurred in encoding the training vectors in $\underline{X}$. Since the dictionary used in our implementation is constructed from the tensor product of two one-dimensional basis sets, two shape codebooks, corresponding to the shape along the x-dimension and that along the y-dimension, respectively, are required. Let N_g and N_s indicate, respectively, the number of regions of the gain and shape partitions, and let R be the regions in the vector space. We have $R = \{R_{i,j,k}; i, j = 1, \cdots, N_s; k = 1, \cdots, N_g\}$.

Three codebooks are adopted in our method:

- The gain codebook $C_g = \{\hat{g}_k; k = 1, 2, \cdots, N_g\}$,
  where $\hat{g}_k$ is the centroid of the region $G_k = \bigcup_{i,j=1}^{N_s} R_{i,j,k}$, which is the region of the input vector space that maps into the gain codeword $\hat{g}_k$.

- The x-axis shape codebook $C_{sx} = \{\hat{S}x_i; i = 1, \cdots, N_s\}$,
  where $\hat{S}x_i$ is the centroid of the region $Ax_i = \bigcup_{j=1}^{N_s} \bigcup_{k=1}^{N_g} R_{i,j,k}$, which is the region of the input vector space that maps into the x-axis shape codeword $\hat{S}x_i$.

- The y-axis shape codebook $C_{sy} = \{\hat{S}y_j; j = 1, \cdots, N_s\}$,
  where $\hat{S}y_j$ is the centroid of the region $Ay_j = \bigcup_{i=1}^{N_S} \bigcup_{k=1}^{N_g} R_{i,j,k}$, which is the region of the input vector space that maps into the y-axis shape codeword $\hat{S}y_j$.

Finally, we adopt a partition $R = \{R_{i,j,k}; i, j = 1, \cdots, N_s; k = 1, \cdots, N_g\}$ of $N_s \times N_s \times N_g$ cells describing the encoder; that is, if X is an element of $\underline{X}$ and $X \in R_{i,j,k}$, then X is mapped into (i, j, k). We express this as $g(X) = \hat{g}_k$, $Sx(X) = \hat{S}x_i$, and $Sy(X) = \hat{S}y_j$. The resulting reproduction is formed from the shape-gain vector $(\hat{S}x_i, \hat{S}y_j, \hat{g}_k)$. Let $Pr(e)$ be the probability of event e. The average distortion is defined by

$$D(C_g, C_{sx}, C_{sy}, R) = Ed(\underline{X}, g(\underline{X})Sx(\underline{X})Sy(\underline{X})),$$

where d is the Euclidean distance.

There are four necessary conditions for optimal codebook design: Let $R^*(C_g, C_{sx}, C_{sy})$ denote the minimum distortion partition.

- Optimal partition for fixed C_g, C_{sx}, and C_{sy} :

  For any encoder partition R,

  $$D(C_g, C_{sx}, C_{sy}, R) \geq D(C_g, C_{sx}, C_{sy}, R^*(C_g, C_{sx}, C_{sy})).$$

  This condition can be implemented by assigning X to the codeword (i, j, k) with minimal distortion between X and all the representative vectors in codebooks.

- Optimal C_g for fixed C_{sx}, C_{sy}, and partition R:

  This necessary condition for optimality conditions the input on the gain cells G_k alone. We have that for any partition R,

  $$D(C_g, C_{sx}, C_{sy}, R) = \sum_{k=1}^{N_g} E[d(\underline{X}, \hat{g}_k Sx(\underline{X})Sy(\underline{X}) | \underline{X} \in G_k)] Pr(\underline{X} \in G_k),$$

  $$D(C_g, C_{sx}, C_{sy}, R) \geq \sum_{k=1}^{N_g} \inf_{g_k} E[d(\underline{X}, g_k Sx(\underline{X})Sy(\underline{X}) | \underline{X} \in G_k)] Pr(\underline{X} \in G_k).$$

  The value of g_k yields the new centroid for a particular k and R. These conditional centroids can be evaluated (see Appendix I), and the collection can be denoted by

  $$g^*(C_{sx}, C_{sy}, R) = \{g_k; k = 1, \cdots, N_g\}.$$

  Thus, we have the condition

  $$D(C_g, C_{sx}, C_{sy}, R) \geq D(g^*(C_{sx}, C_{sy}, R), C_{sx}, C_{sy}, R). \quad (3)$$

- Optimal C_{sx} for fixed C_{sy}, C_g and partition R:

  We condition on the x-shape partition Ax_i for a particular i and R and, write

$$D(C_g, C_{sx}, C_{sy}, R) = \sum_{i=1}^{N_s} E[d(\underline{X}, g(\underline{X})\hat{S}x_i Sy(\underline{X})|\underline{X} \in Ax_i)]Pr(\underline{X} \in Ax_i),$$

$$D(C_g, C_{sx}, C_{sy}, R) \geq \sum_{i=1}^{N_s} \inf_{Sx_i} E[d(\underline{X}, g(\underline{X})Sx_i Sy(\underline{X})|\underline{X} \in Ax_i)]Pr(\underline{X} \in Ax_i),$$

  which defines a new conditional x-shape centroid, Sx_i, given the overall partition. The new x-shape centroid can be evaluated and is given in Appendix I. If the collection of these conditional x-shape centroids is denoted by

$$Ax^*(C_g, C_{sy}, R) = \{Sx_i; i = 1, \cdots, N_s\},$$

  then we have the condition

$$D(C_g, C_{sx}, C_{sy}, R) \geq D(C_g, Ax^*(C_g, C_{sy}, R), C_{sy}, R). \tag{4}$$

- Optimal C_{sy} for fixed C_{sx}, C_g and partition R:

  Similarly, we condition on the y-shape partition Ay_j for a particular j and R, and write

$$D(C_g, C_{sx}, C_{sy}, R) = \sum_{j=1}^{N_s} E[d(\underline{X}, g(\underline{X})Sx(\underline{X})\hat{S}y_j|\underline{X} \in Ay_j)]Pr(\underline{X} \in Ay_j),$$

$$D(C_g, C_{sx}, C_{sy}, R) \geq \sum_{j=1}^{N_s} \inf_{Sy_j} E[d(\underline{X}, g(\underline{X})Sx(\underline{X})Sy_j|\underline{X} \in Ay_j)]Pr(\underline{X} \in Ay_j),$$

  which defines a new conditional y-shape centroid Sy_j given the overall partition. The evaluation of the new y-shape centroid Sy_j is given in Appendix I. If the collection of these conditional shape centroids is denoted by

$$Ay^*(C_g, C_{sx}, R) = \{Sy_j; j = 1, \cdots, N_s\},$$

  then we have the condition

$$D(C_g, C_{sx}, C_{sy}, R) \geq D(C_g, C_{sx}, Ay^*(C_g, C_{sx}, R), R). \tag{5}$$

3.2.1 Joint Optimization Algorithm

Given an initial set of three codebooks, the four necessary equations, mentioned previously, can be used in an iterative algorithm to find a locally optimal quantizer. The iterative algorithm is usually referred to as a joint optimization algorithm[16]. In the literature, the joint optimal algorithm has been applied to find the best local quantizer for one gain and one shape codebook. We will extend the algorithm to two shape codebooks and one gain codebook. Our algorithm is given in the following:

Step 0. Initialization: Assume N_g, N_s, and $\epsilon > 0$, initial product codebooks $C_g(0), C_{sx}(0)$, and $C_{sy}(0)$, and a training set $\underline{X}$. Set $m = 0$ and $D_{-1} = \infty$.

Step 1. Find the optimum partition $R^*(C_g(m), C_{sx}(m), C_{sy}(m))$ of $\underline{X}$.

Step 2. Compute the average distortion:
$D_m = D(C_g(m), C_{sx}(m), C_{sy}(m), R^*(C_g(m), C_{sx}(m), C_{sy}(m)))$.

Step 3. If $|(D_{m-1} - D_m)/D_m| < \epsilon$, then halt with
$(C_g(m), C_{sx}(m), C_{sy}(m), R^*(C_g(m), C_{sx}(m), C_{sy}(m)))$ describing the final quantizer; else continue.

Step 4. Compute the optimum x-axis shape codebook:
$C_{sx}(m+1) = \{\hat{S}x_i\} = Ax^*(C_g(m), C_{sy}(m), R^*(C_g(m), C_{sx}(m), C_{sy}(m)))$.

Step 5. Compute the optimum y-axis shape codebook:
$C_{sy}(m+1) = \{\hat{S}y_j\} = Ay^*(C_g(m), C_{sx}(m), R^*(C_g(m), C_{sx}(m), C_{sy}(m)))$.

Step 6. Compute the optimum partition: $R^*(C_g(m), C_{sx}(m+1), C_{sy}(m+1))$.

Step 7. Compute the optimum gain codebook:
$C_g(m+1) = \{\hat{g}_k\} = G^*(C_{sx}(m+1), C_{sy}(m+1), R^*(C_g(m), C_{sx}(m+1), C_{sy}(m+1)))$.

Step 8. Set $m = m + 1$ and go to Step 1.

In steps 4, 5, and 7, we need to compute, respectively, the centroids of the x-shape, y-shape, and gain. We denote $x_l \in R_{i,j,k}$ as the element x_l of random variable $\underline{X}$ appearing in partition region $R_{i,j,k}$. The x-shape centroid is given by

$$\hat{S}x_i = \frac{E[g(\underline{X})Sy(\underline{X})\underline{X}|\underline{X} \in Ax_i]}{\|E[g(\underline{X})Sy(\underline{X})\underline{X}|\underline{X} \in Ax_i]\|},$$

which is calculated by

$$\hat{S}x_i = \frac{\sum_{k=1}^{N_g} \sum_{j=1}^{N_s} \sum_{l:x_l \in R_{i,j,k}} g(x_l) Sy(x_l) x_l}{\| \sum_{k=1}^{N_g} \sum_{j=1}^{N_s} \sum_{l:x_l \in R_{i,j,k}} g(x_l) Sy(x_l) x_l \|}.$$

Similarly, for the y-shape centroid, we have the y-shape centroid

$$\hat{S}y_j = \frac{E[g(\underline{X}) Sx(\underline{X}) \underline{X} | \underline{X} \in Ay_j]}{\|E[g(\underline{X}) Sx(\underline{X}) \underline{X} | \underline{X} \in Ay_j]\|},$$

which is calculated by

$$\hat{S}y_j = \frac{\sum_{k=1}^{N_g} \sum_{i=1}^{N_s} \sum_{l:x_l \in R_{i,j,k}} g(x_l) Sx(x_l) x_l}{\| \sum_{k=1}^{N_g} \sum_{i=1}^{N_s} \sum_{l:x_l \in R_{i,j,k}} g(x_l) Sx(x_l) x_l \|}.$$

The gain centroid is

$$\hat{g}_k = E(\underline{X}^t Sx(\underline{X}) Sy(\underline{X}) | \underline{X} \in G_k),$$

which is calculated by

$$\hat{g}_k = \frac{1}{\|G_k\|} \sum_{i,j=1}^{N_s} \sum_{l:x_l \in R_{i,j,k}} Sx(x_l) Sy(x_l) x_l.$$

The detailed derivations of the centroids are given in Appendix I. Each of the steps 4, 5, and 7, results in a quantizer whose average distortion is better or at least as good as the previous quantizer (See Eqs. (3), (4), and (5), and Appendix I). Since the distortion is always positive, the algorithm will converge to a local minimum.

Remark. Let us recall Neff and Zakhor's method in parlance of shape-gain product VQ. A basis is a vector in the radial 1 sphere in a vector space. Their large training dictionary corresponds to many vectors spread throughout the sphere. They selected N_s basis vectors from the vectors.These N_s basis vectors form their shape codebook. Their gain codebook is obtained by means of uniform scalar quantization. Their method produces one shape and one gain codebook. Unlike ours, their quantizer is not optimal for the training sequences.

4 Experimental Results

Various performance evaluations were carried out to compare the performance of H.263 with that of our matching pursuit with the basis set obtained using shape-gain VQ, and that of our matching pursuit algorithm with Neff and Zakhor's basis set.

The experimental results of H.263 were obtained from the TMN encoder, available publicly at http://www.nta.no/brukere/DVC/. The motion estimation part of our method is identical to that of H.263. The rate control of the two codecs is synchronized. They consume approximately the same number of bits in encoding each motion residual frame. Five video sequences: Akiyo, Mother and Daughter, Sean, Miss America, and Salesman, were used on our experiments. The common features of these sequences are that they contain mostly one or two head-and-shoulder type objects, and that there is not much fast global motion of these objects against their backgrounds. A snapshot from each sequence was taken and is shown in Fig. 4.

The performance evaluation was done with the video sequences were coded at 24 kbit/s, 10 frame/s, and with the QCIF format. The dictionary used to test a video sequence was obtained from the remaining 4 sequences under the same conditions. In order to compare the performance with that of H.263, we needed to synchronize the rate control in our experiments. Let A, B, C, and D be sequences used to train our dictionary, and let E be the sequence used to test the performance. We give our detailed implementation in the following two algorithms:

- **Dictionary Optimization :**
  For each sequence A, B, C, or D:

  1. Run the TMN H.263 encoder with a target bit rate of 24 kbit/sec and 10 frames/s, and keep a record of the number of bits used in each residual frame.
  2. Let B_{curr} be the bit budget for encoding the current residual frame. This number is obtained by running the TMN H.263. We encode each atom with a fixed number of bits, A_{atom}. In our implementation, we used 16 bits for image locations, a total of 10 bits for horizontal and vertical indices, and 5 bits for quantizing inner product values. Then, the number of atoms for the current residual frame is
  $$N_{curr} = \lfloor \frac{B_{curr}}{A_{atom}} \rfloor.$$
  3. The horizontal and vertical indices, and the inner product values of the set of atoms $\{N_{curr}\}$ are collected. As to the image location of an atom, we determine the block (see (b) in Fig. 3) to which the locations belong and record only the upper left positions in the block.

The initial shape codebooks are the basis functions whose indices are given in Table 1. Each shape codebook has $20(= N_s)$ codewords. Totally, we have 400 basis functions. The initial gain codebook is a uniform scalar quantizer of size $10(= N_g)$. The codebooks are modified in steps 4, 5, and 7 in our joint optimization algorithm since each of them is an evaluation of the new centroids. This leads to an optimized basis set for our sequences A, B, C, and D, and it is later used to evaluate the performance of sequence E.

- **Performance Evaluation:**

  The parameters of the atoms collected during **Dictionary Optimization** can be encoded more efficiently by means of entropy coding: We used separate code tables for the horizontal and vertical indexes (*e.g.,* k in Table 1), and they were coded with fixed Huffman codes, respectively. The quantized inner product values were also encoded by a fixed Huffman code. The locations of the upper left corner of a block were numbered and encoded using a fixed Huffman code. Then, we applied the following steps to sequence E:

  1. Run the TMN H.263 encoder on video sequence E with a target bit rate of 24 kbit/sec and 10 frames/s, and keep a record of the number of bits used in each residual frame.
  2. Let B_{curr} be the bit budget for encoding the current residual frame. We run our matching pursuit codec on this residual frame such that atoms are collected up to the point where they consume approximately the same number of bits for each frame as does the standard implementation of H.263.

     In this run, the number of bits for each atom vary since the horizontal and vertical indices and the inner product value and the block positions to whcih the atom belongs are entropy-coded. Then, the offsets the atom locations with respect to the upper left corner of the block shown in (c) of Fig. 3 are recorded based on a fixed number of bits. There are 4 bits in each direction and 8 bits in total.

The top part of Fig. 5 compares the luminance PSNR of the sequence Mother and Daughter based on our matching pursuit codec with that of TMN H.263. The middle part of the figure shows an image of the initial dictionary, and the bottom part of the figure shows an image of our dictionary used for evaluation of the performance of Mother and Daughter. Once we compare the bottom two images, the differences between them are obvious.

Three curves in Fig. 6 give the average luminance PSNR performance of various methods. One of them was obtained by averaging the luminance PSNR in tests on any of the abovementioned five sequences using our matching pursuit codec with the dictionary obtained from the other 4 sequences. Table 2 gives the average PSRN performance of H.263 and of our matching pursuit codec with Neff and Zakhor's dictionary and with the optimized dictionary from the other 4 sequences. The average PSNR improved about 0.5dB per frame with the shape-gain optimized dictionary compared to the results with the initial dictionary.

The following two experiments were performed by comparing the PSNR performance of our codec with that of H.263 on foreman and Staphan sequences. Our dictionary was obtained from the abovementioned five sequences.

The top part of Fig.7 is a snapshot of the foreman sequence. The sequence is of the head-and-should type with fast body motion. The bottom subfigure compares the PSNR performance of our method with that of H.263 for the video sequence. Our method has an average PSNR gain of about 0.3 dB over that obtained using H.263.

The foreground objects in the Staphan sequence begin with slow motion, followed by fast motion. The foreground objects take up only a small portion of a frame and the background is also changing. A snapshot of it is shown at the top of Fig. 8. In the bottom part of the figure, one can clearly see that the PSRN begins at about 32dB and decreases to about 23dB when the motion becomes fast.

Finally, a blilnd test was carried out to evaluate the perceptal quality of the sequences obtained based on H.263 and our method, respectively. For any sequence listed in the Table 3, we synchronized the sequences, by H.263 and by ours, and played them twice to 20 blind-selected subjects in a classroom. The subjects did not know which sequence was obtained by which method. Then, we asked the subjects to vote for the sequence with better visual quality. The results were given in Table 3. Except for the sequences Salesman and Mother & Daughter, the other sequences obtained based on the matching pursuit method have obtained more votes than those of H.263.

5 Conclusion

We have shown that the techniques used to design shape-gain product VQ codebooks can be used to optimize the dictionary of matching pursuits. Experimental results showed that, compared to results obtained using Neff and Zakhor's basis set, 0.5 dB improvement could be achieved on video sequences with mostly one or two head-and-should type foreground objects with slow motion against the background using a dictionary trained by the shape-gain VQ. Experiments also showed that the reconstructed frames of these sequences had better visualization quality than did those obtained using H.263. However, as with most training type algorithms, the performance depends on the statistics of the input and training sequences. This can be seen from our experimental results for the Stefan sequence. Thus, an adaptive algorithm must be developed which will modify a dictionary that is suitable for slow motion video sequences to obtain one suitable for fast moving objects. This topic is currently under investigation.

Acknowledge: We would like to thank Dr. Zakhor, Dr. Neff and anonymous reviewers for many valuable discussions on our first draft. In our original implementation, there was a serious mistake which was pointed out by Dr. Neff. We would also like to thank Mr. Jian-Liang Lin and Mr. Chien-Kai Peng for helping us to prepare simulation data.

Appendix I Centroid Calculations

This appendix gives the centroid calculations. We assume that the gain mapping is $g(\underline{X})$, which maps $\underline{X}$ to its closest gain representative, that the x-axis shape mapping is $Sx(\underline{X})$, and that the y-axis shape mapping is $Sy(\underline{X})$. $F(\underline{X})$ is the distribution of the random variable $\underline{X}$, and d is the Euclidean distance.

x-axis Shape Centroid: The centroid $\hat{S}x_i$ at shape cell Ax_i is used to solve

$$\hat{S}x_i = arg \min_{u_i \in Ax_i} \int_{Ax_i} d(\underline{X}; g(\underline{X}) u_i Sy(\underline{X})) dF(\underline{X}).$$

The quantity to be minimized becomes

$$\int_{Ax_i} [\underline{X}^T \underline{X} + g^2(\underline{X})] dF(\underline{X}) - 2u_i^T \int_{Ax_i} g(\underline{X}) Sy(\underline{X}) \underline{X} dF(\underline{X}).$$

The first term does not depend on u_j . After applying the Cauchy-Schwarz inequality to the second term, the minimized u_i is in the direction of $\int_{Ax_i} g(\underline{X}) \underline{X} Sy(\underline{X}) dF(\underline{X})$, i.e., in the direction of $E\{g(\underline{X}) \underline{X} Sy(\underline{X}) | \underline{X} \in Ax_i\}$. Since the shape centroid is normalized to 1, we have

$$\hat{S}x_i = \frac{E[g(\underline{X}) Sy(\underline{X}) \underline{X} | \underline{X} \in Ax_i]}{\| E[g(\underline{X}) Sy(\underline{X}) \underline{X} | \underline{X} \in Ax_i] \|}.$$

y-axis Shape Centroid: Similar to the x-axis Shape centroid, the y-shape centroid is

$$\hat{S}y_j = \frac{E[g(\underline{X}) Sx(\underline{X}) \underline{X} | \underline{X} \in Ay_j]}{\| E[g(\underline{X}) Sx(\underline{X}) \underline{X} | \underline{X} \in Ay_j] \|}.$$

Gain Centroid: The gain centroid $\hat{g_k}$ at gain cell G_k is used to solve

$$\hat{g}_k = arg \min_{u_k \in G_k} \int_{G_k} d(\underline{X}; u_k Sx(\underline{X}) Sy(\underline{X})) dF(\underline{X}).$$

The quantity to be minimized becomes

$$\int_{G_k} [\underline{X}^T \underline{X} + g^2(\underline{X})] dF(\underline{X}) - 2u_k \int_{G_k} \underline{X}^T Sx(\underline{X}) Sy(\underline{X}) dF(\underline{X}).$$

The first term does not depend on u_j . After applying the Cauchy-Schwarz inequality to the second term, the minimized u_k is in the direction of $\int_{G_k} \underline{X}^T Sx(\underline{X}) Sy(\underline{X}) dF(\underline{X})$, i.e., in the direction of $E\{\underline{X}^T Sx(\underline{X}) Sy(\underline{X}) | \underline{X} \in G_k\}$. Thus, we have

$$\hat{g}_k = E[\underline{X}^T Sx(\underline{X}) Sy(\underline{X}) | \underline{X} \in G_k].$$

References

[1] R. Neff and A. Zakhor, “Very low bit-rate video coding based on matching pursuits”, *IEEE Trans. Circuit and systems for video technology,* Vol. 7, No.1, Feb. 1997, pp. 158-171.

[2] O. K. Al-Shaykh, E. Miloslavsky, T. Nomura, R. Neff, and A. Zakhor, “Video Compression Using Matching Pursuits”, *IEEE Trans. Circuit and systems for video technology,* Vol. 9, No1, Feb. 1999, pp. 123-143.

[3] M. R. Banham and J. C. Braillean, ”A selective update approach to matching pursuits video coding”, *IEEE Trans. Circuit and systems for video technology,* Vol. 7,No. 1, Feb. 1997, pp. 119-129.

[4] S. Mallat and Z. Zhang, ”Matching pursuits with time-frequency dictionaries”, *IEEE Trans. Signal Processing,* Vol. 41, Dec 1993, pp. 3397-3415.

[5] Y. Linde, A. Buzo, and R. M. Gray, ”An algorithm for vector quantizer design”, *IEEE Trans. Commun.,* Vol. COM-28, Jan 1980, pp. 84-95.

[6] R. M. Gray, J. C. Kieffer, and Y. Linde, ”Locally optimal block quantizer design”, *Inform. Contr.,* Vol.45, May 1980, pp. 178-198.

[7] M. J. Sabin, “Product code vector quantization”, *Ph.D. dissertation, Stanford Univ.,* Stanford, CA.

[8] B. Torresani, ”Wavelet associated with representations of the affine Weyl-Heisenberg group”, *J. Math. Physics,* Vol. 32, May 1991, pp. 1273-1279.

[9] M. Vetterli and T. Kalker, “Matching pursuit for compression and application to motion compensated video coding”, *IEEE Image Processing Conf.,* 1994, pp. 725-729.

[10] R. Neff and A. Zakhor, ”Matching pursuit video coding at very low bit rates”, *IEEE Data Compression Conf.,* Snowbird, UT, Mar. 1995, pp. 441-420.

[11] R. Neff and A. Zakhor, ”Very low bit rate video coding using matching pursuits”, *Proc. of SPIE Conf. on Visual Comm. and Image Proc.,* Vol. 2308, No. 1, Sept. 1994, pp. 47-60.

[12] ITU Telecommunication Standardization Sector LBC-95, Video Codec Test Model TMN5.

[13] ITU-T Study Group 15, Working Party 15/1, Draft Recommendation H.263, Dec. 1995.

[14] S. Ramakrishnan, K. Rose, and A. Gersho, "Constrained-Storage Vector Quantization with a Universal Codebook," *IEEE Transaction on Image Processing,* vol. 7, no. 6, June 1998.

[15] M. J. Sabin and R. M. Gray, "Product code vector quantizers for waveform and voice coding", *IEEE Transactions on ASSP,* Vol. 32, No. 3, June 1984.

[16] A. Gersho and R. M. Gray, "Vector Quantization and Signal Compression", *Kluwer Academic Publishers,* 1992.

k	s_k	η_k	ϕ_k
0	1	0	0
1	3	0	0
2	5	0	0
3	7	0	0
4	9	0	0
5	12	0	0
6	14	0	0
7	17	0	0
8	20	0	0
9	1.4	1	$\frac{\pi}{2}$
10	5	1	$\frac{\pi}{2}$
11	12	1	$\frac{\pi}{2}$
12	16	1	$\frac{\pi}{2}$
13	20	1	$\frac{\pi}{2}$
14	4	2	0
15	4	3	0
16	8	3	0
17	4	4	0
18	4	2	$\frac{\pi}{4}$
19	4	4	$\frac{\pi}{4}$

Table 1: The 1-D Basis Set used by Neff and Zakhor.

Avg. PSNR	Akiyo	Mother and Daughter	Sean	Miss America	Salesman
H.263	36.40566	33.1172	32.32657	36.43167	29.99369
Our Initial	35.88975	32.9032	32.01495	35.95505	29.89868
Our Final	35.39164	33.6069	32.88242	36.12128	30.1233

Table 2: Performance comparisons between H.263, our codec with Neff and Zakhor's bases (Our Initial), and our codec with shape-gain optimized bases, obtained by training the other 4 sequences (Our Final).

Sequence	H.263	Ours	Dictionary
Akiyo	1	19	M,S,MA,SM
Mother and Daughter	11	9	A,S,MA,SM
Sean	5	15	A,M,MA,SM
Miss America	5	15	A,M,S,SM
Salesman	12	7	A,M,S,MA
Foreman	1	19	A,M,S,MA,SM

Table 3: The results of a blind perceptual quality test on sequences encoded by H.263 and by our method. The dictionary was used to encode each sequence by our method was given in the last column where A, M, S, MA, SM stand for Akiyo, Mother and Daughter, Sean, Miss America, Salesman, respectively.

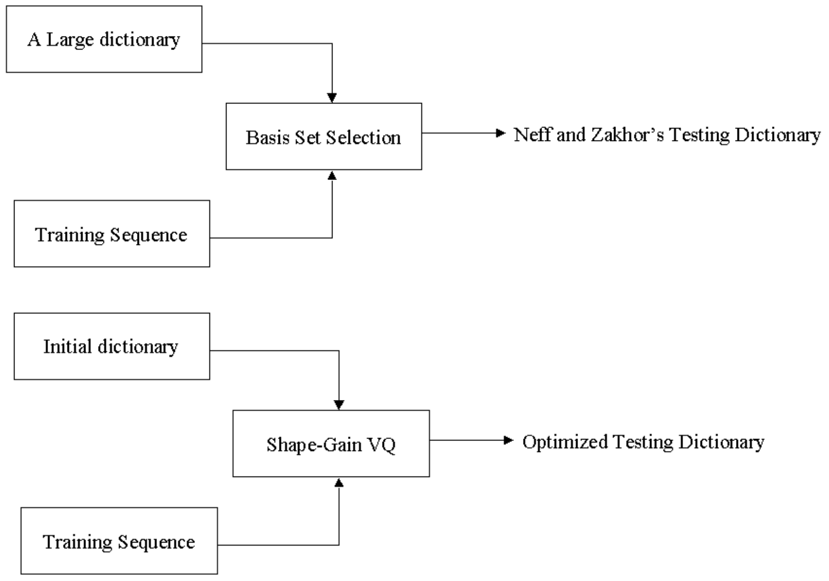


Figure 1: Block diagrams to compare the methods. Top: Neff and Zakhor's. Bottom: Ours.

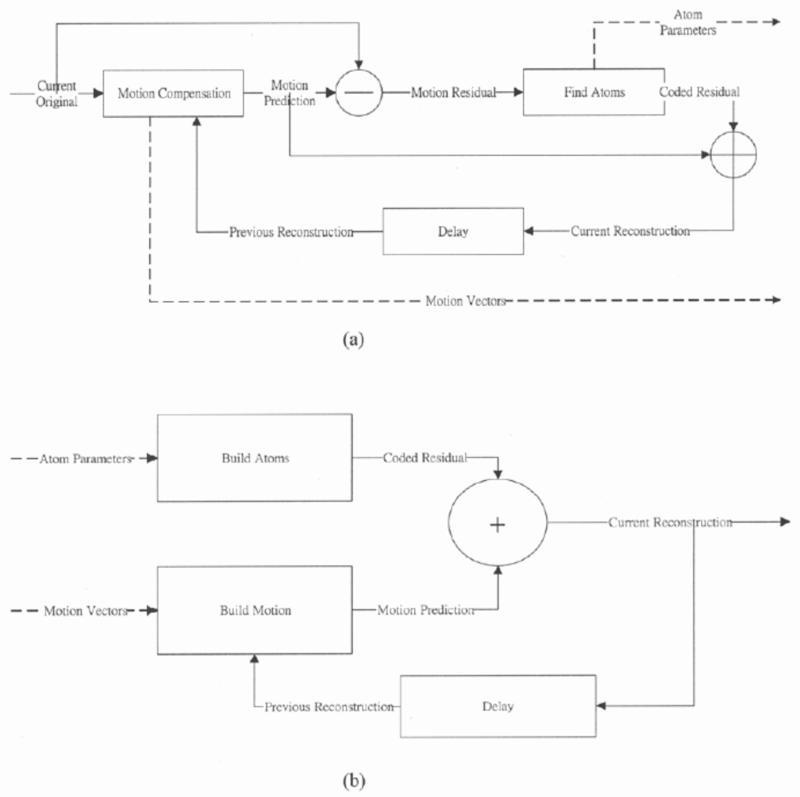


Fig.2 (a) Block Diagram of Matching Pursuit Encoder. (b) Decoder.
(from Neff and Zakhor)

Figure 2: Block diagram of the matching pursuit codec. (a) Encoder. (b) Decoder.

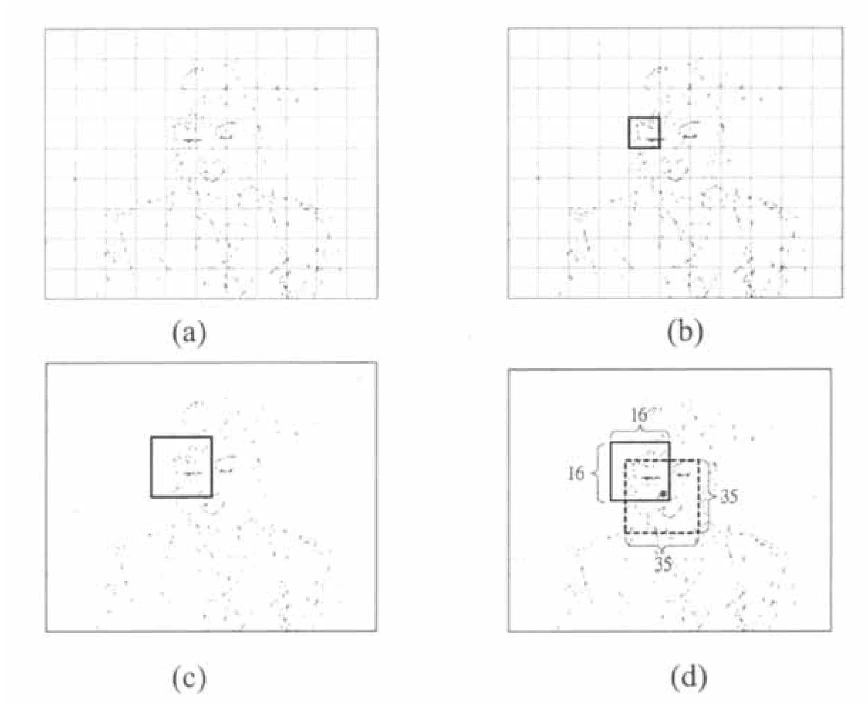


Figure 3: Inner product block search determination.

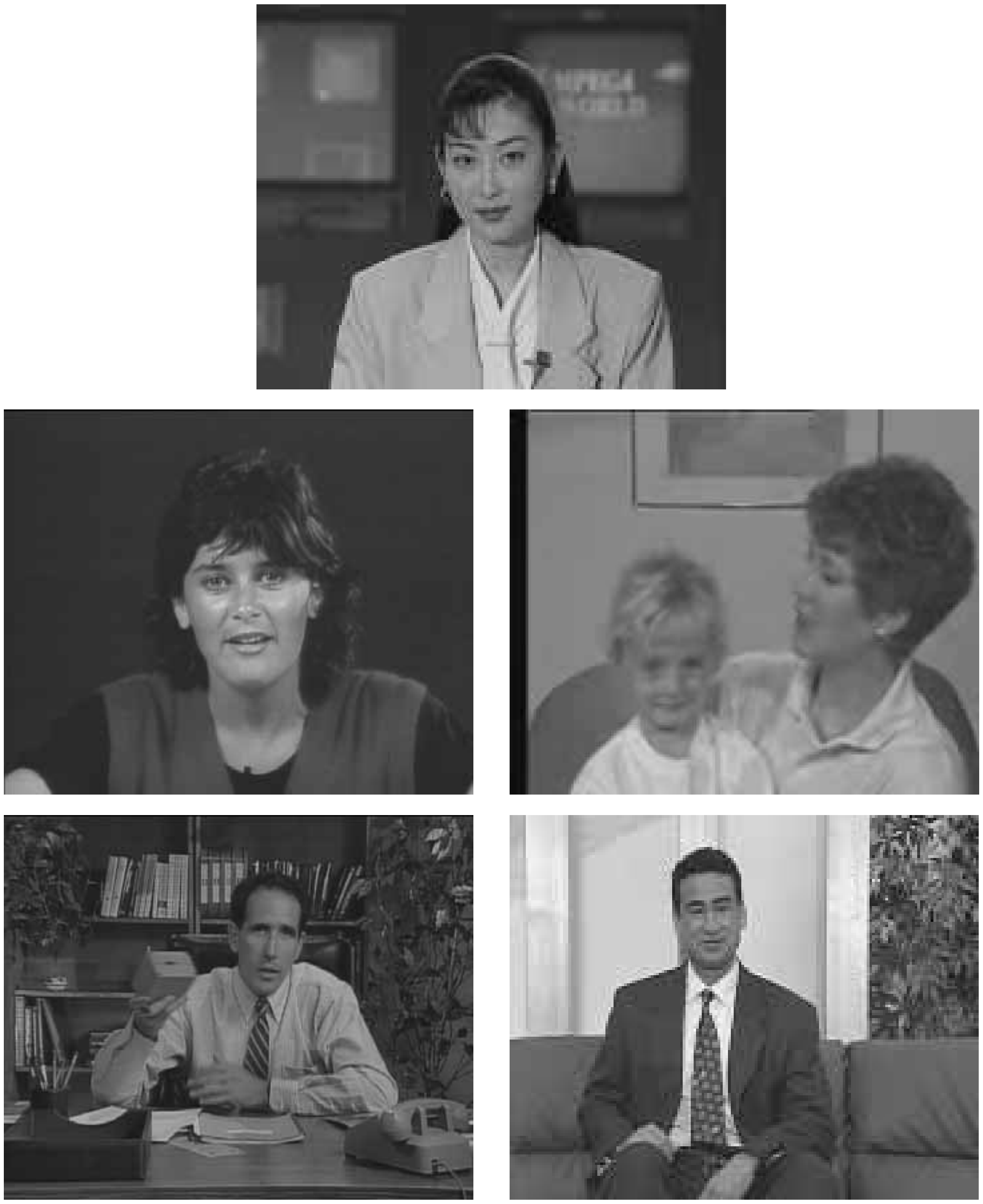
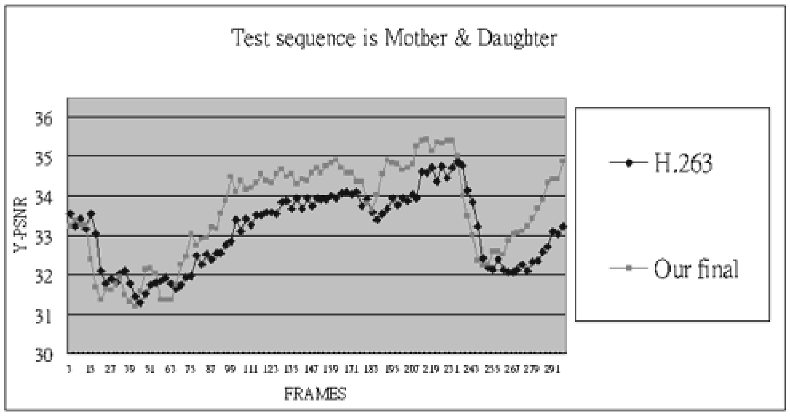
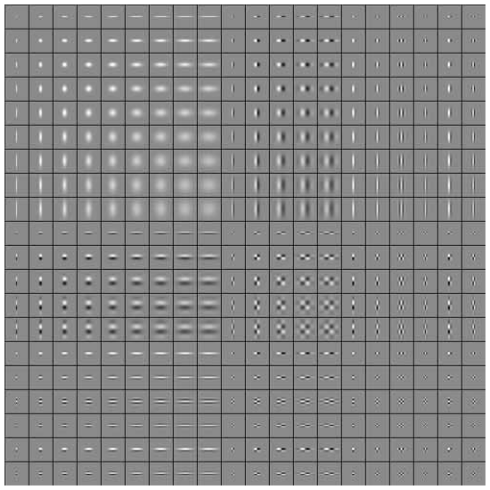


Figure 4: Snapshots of Akiyo (top), Miss America (middle left), Mother and Daughter (middle right), Salesman (bottom left), and sean (bottom right).





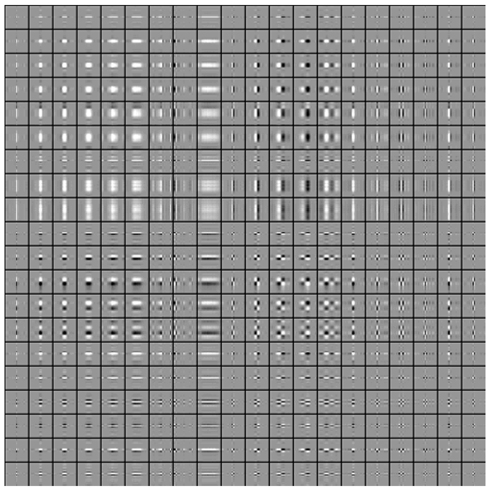


Figure 5: Top: PSNR performance comparison of Mother and Daughter with H.263 and ours. Our dictionary is shown in the bottom image. Middle: Image of the initial dictionary. There are 400 bases. Bottom: Image of the final dictionary obtained after shape-gain VQ optimization.

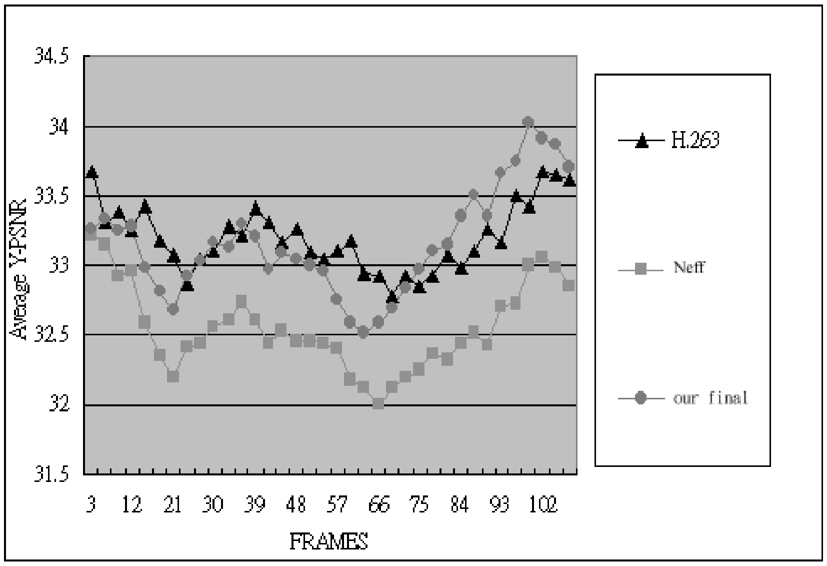


Figure 6: Average PSRN performance obtained by H.263, our matching pursuit with optimized dictionary and with the Neff and Zakhor's dictionary, respectively.

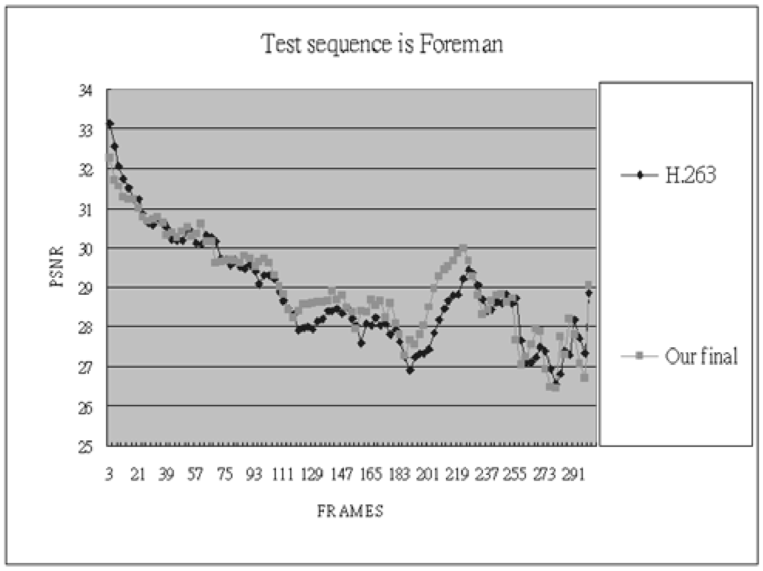


Figure 7: Top: A snapshot from the foreman sequence. Bottom: PSNR comparison of the sequence by H.263 and by ours.

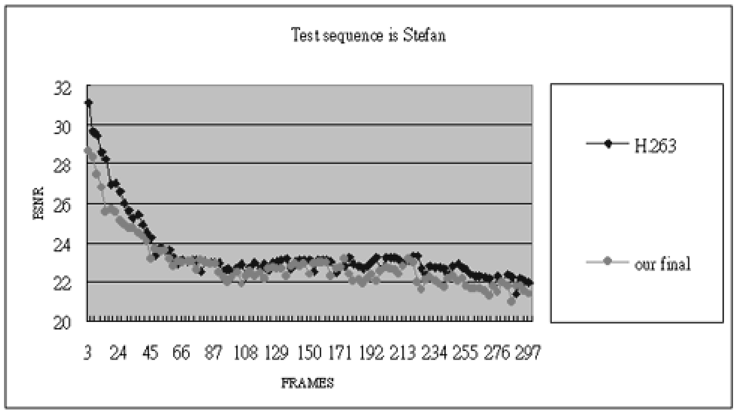


Figure 8: Top: A snapshot from the Stefan sequence. Bottom: PSNR comparison of the sequence by H.263 and by ours.