Very Low Bit Rate Video Coding Based on Matching Pursuits

Ralph Neff and Avideh Zakhor

Both authors are with the Department of Electrical Engineering and Computer Science at the University of California, Berkeley CA 94720

Abstract

We present a video compression algorithm which performs well on generic sequences at very low bit rates. This algorithm was the basis for a submission to the November 1995 MPEG4 subjective tests. The main novelty of the algorithm is a matching-pursuit based motion residual coder. The method uses an inner-product search to decompose motion residual signals on an overcomplete dictionary of separable Gabor functions. This coding strategy allows residual bits to be concentrated in the areas where they are needed most, providing detailed reconstructions without block artifacts. Coding results from the MPEG4 Class A compression sequences are presented and compared to H.263. We demonstrate that the matching pursuit system outperforms the H.263 standard in both PSNR and visual quality.

I. Introduction

The MPEG4 standard is being developed to meet a wide range of current and anticipated needs in the area of low bit rate audio-visual communications. Although the standard will be designed to be flexible, there are two main application classes which are currently motivating its development. The first involves real-time two way video communication at very low bit rates. This requires real-time operation at both the encoder and decoder, as well as robustness in noisy channel environments such as wireless networks or telephone lines. The second class involves multimedia services and remote database access. While these applications may not require real-time encoding, they do call for additional functionalities such as object-based manipulation and editing. Both classes of applications are expected to function at bit rates as low as 10 or 24 kbit/s.

The current standard of comparison for real-time two way video communication is the ITU's recommendation H.263 [1]. This system is based on a hybrid motion-compensated DCT structure similar to the two existing MPEG standards. It includes additional features such as an overlapping motion model which improves performance at very low bit rates. There is currently no standard system for doing the object-based coding required by multimedia database applications. However, numerous examples of object oriented coding can be found in the literature [2] [3] [4], and several such systems were evaluated at the MPEG4 subjective tests.

The hybrid motion-compensated DCT structure is a part of nearly all existing video coding standards, and variations of this structure have been proposed for object based

coding as well [4] [5]. This popularity can be attributed to the fact that the DCT performs well in a wide variety of coding situations, and because a substantial investment has been made in developing chipsets to implement DCT-based coders. Unfortunately, block-based DCT systems have trouble coding sequences at very low bit rates. At rates below 20 kbit/s, the number of coded DCT coefficients becomes very small, and each coefficient must be represented at a very coarse level of quantization. The resulting coded images have noticeable distortion, and block edge artifacts can be seen in the reconstruction.

To overcome these problems, we propose a hybrid system in which the block-DCT residual coder is replaced with a new coding method which behaves better at low rates. Instead of expanding the motion residual signal on a complete basis such as the DCT, we propose an expansion on a larger, more flexible basis set. Since such an overcomplete basis contains a wider variety of structures than the DCT basis, we expect it to be better able to represent the residual signal using fewer coefficients. The expansion is done using a multistage technique called matching pursuits. This technique was developed for signal analysis by Mallat and Zhang [6], and is related to earlier work in statistics [7] and multistage vector quantization [8]. The matching pursuit technique is general in that it places virtually no restrictions on the choice of basis dictionary. We choose an overcomplete set of separable Gabor functions, which do not contain artificial block edges. We also remove grid positioning restrictions, allowing elements of the basis set to exist at any pixel resolution location within the image. These assumptions allow the system to avoid the artifacts most often produced by low bit rate DCT systems.

The following section reviews the theory and some of the mathematics behind the matching pursuit technique. Section III provides a detailed description of the coding system. This includes subsections relating to intraframe coding, motion compensation, matching pursuit residual coding, and rate control. Coding results are presented in Section IV, and conclusions can be found in Section V.

II. Matching Pursuit Theory

The Matching Pursuit algorithm, as proposed by Mallat and Zhang, [6] expands a signal using an overcomplete dictionary of functions. For simplicity, the procedure can be illustrated with the decomposition of a 1-D time signal. Suppose we want to represent a

signal $f(t)$ using basis functions from a dictionary set $\mathcal{G}$. Individual dictionary functions can be denoted as:

$$g_\gamma(t) \in \mathcal{G} \tag{1}$$

Here γ is an indexing parameter associated with a particular dictionary element. The decomposition begins by choosing γ to maximize the absolute value of the following inner product:

$$p = < f(t), g_\gamma(t) > \tag{2}$$

We then say that p is an expansion coefficient for the signal onto the dictionary function $g_\gamma(t)$. A residual signal is computed as:

$$R(t) = f(t) - p\, g_\gamma(t) \tag{3}$$

This residual signal is then expanded in the same way as the original signal. The procedure continues iteratively until either a set number of expansion coefficients are generated or some energy threshold for the residual is reached. Each stage n yields a dictionary structure specified by γ_n, an expansion coefficient p_n, and a residual R_n which is passed on to the next stage. After a total of M stages, the signal can be approximated by a linear function of the dictionary elements:

$$\hat{f}(t) = \sum_{n=1}^{M} p_n\, g_{\gamma_n}(t) \tag{4}$$

The above technique has some very useful signal representation properties. For example, the dictionary element chosen at each stage is the element which provides the greatest reduction in mean square error between the true signal $f(t)$ and the coded signal $\hat{f}(t)$. In this sense, the signal structures are coded in order of importance, which is desirable in situations where the bit budget is very limited. For image and video coding applications, this means that the most visible features tend to be coded first. Weaker image features are coded later, if at all. It is even possible to control which types of image features are coded well by choosing dictionary functions to match the shape, scale, or frequency of the desired features.

An interesting feature of the matching pursuit technique is that it places very few restrictions on the dictionary set. The original Mallat and Zhang paper considers both

Gabor and wavepacket function dictionaries, but such structure is not required by the algorithm itself. Mallat and Zhang showed that if the dictionary set is at least complete, then $\hat{f}(t)$ will eventually converge to $f(t)$, though the rate of convergence is not guaranteed [6]. Convergence speed and thus coding efficiency are strongly related to the choice of dictionary set. However, true dictionary optimization can be difficult since there are so few restrictions. Any collection of arbitrarily sized and shaped functions can be used with matching pursuits, as long as completeness is satisfied.

There has been much recent interest in using matching pursuits for image processing applications. Bergeaud and Mallat [9] use the technique to decompose still images on a dictionary of oriented Gabor functions. Vetterli and Kalker [10] present an interesting twist on the traditional hybrid DCT video coding algorithm. They use matching pursuits blockwise with a dictionary consisting of DCT basis functions and translated motion blocks. The matching pursuit representation can also be useful for pattern recognition, as demonstrated by Phillips [11]. For many applications, the algorithm can be computationally intensive. For this reason, several researchers have proposed speed enhancements [9], [12], [13].

The system presented in this paper is a hybrid motion compensated video codec in which the motion residual is coded using matching pursuits. This is an enhanced version of the coding system presented in [14] and [15]. The current system allows dictionary elements to have arbitrary sizes, which speeds the search for small scale functions and makes storage more efficient. Larger basis functions are added to increase coding efficiency for scenes involving higher motion or lighting changes. The complete system, including enhancements, is described in the next section.

III. Detailed System Description

This section describes the matching pursuit coding system used to generate the results presented in Section IV. Simplified block diagrams of the encoder and decoder are shown in Figure 1. As can be seen in Figure 1a, original images are first motion compensated using the previous reconstructed image. Here we use the advanced motion model from H.263, as described in Section III-A. The matching pursuit algorithm is then used to decompose the motion residual signal into coded dictionary functions which are called atoms. The

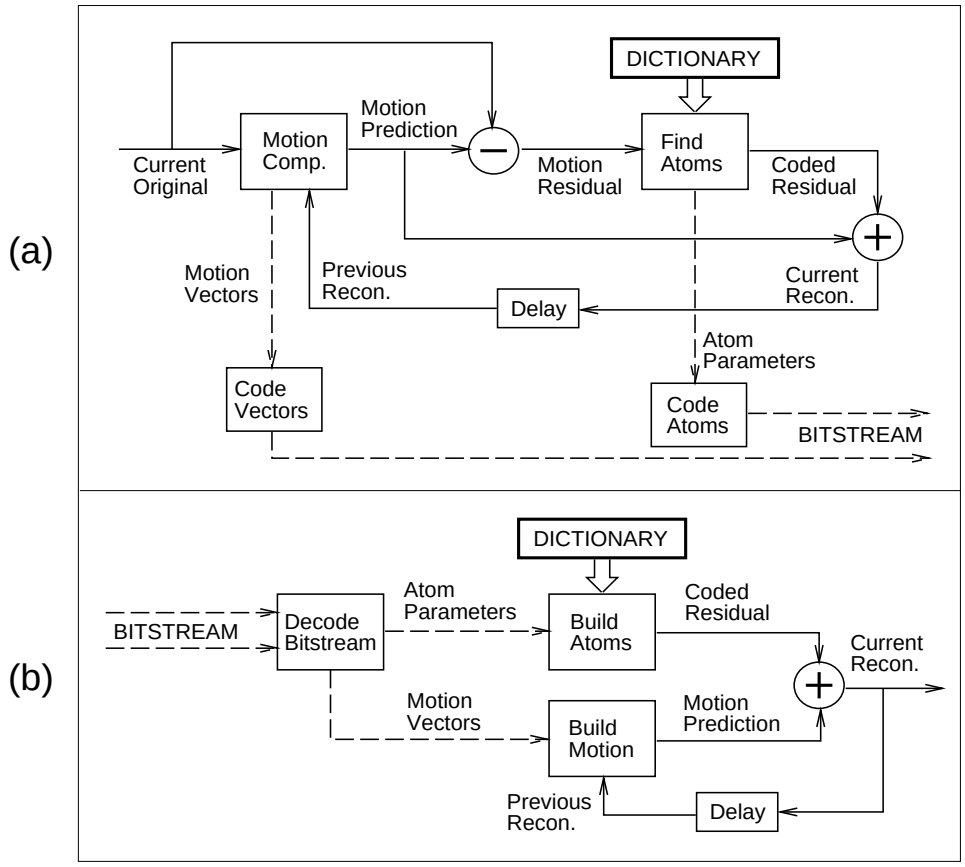

Fig. 1. (a) Block diagram of encoder. (b) Decoder.

process of finding and coding atoms is described in Section III-B. Once the motion vectors and atom parameters are found, they can be efficiently coded and sent to the decoder, shown in Figure 1b. This process is governed by a rate control system, which is discussed in Section III-C. Finally, Section III-D describes the scalable subband coder [16] which is used to code the first intra frame.

A. Motion Compensation

The motion model used by the matching pursuit system is identical to the advanced prediction model used by TMN [17], which is a specific implementation of H.263 [1]. This section provides a brief description of this model, and also describes how intra blocks are coded when motion prediction fails.

The system operates on 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 each block is assigned either a single motion vector, a set of four motion vectors, or a set of intra block parameters. The motion search for a block begins with a ± 15 pixel search at integer resolution. The inter/intra decision criterion from TMN is used to decide which prediction mode should be used. If the block is coded in inter mode, then the integer resolution search is refined with a ± 1 half pixel search at half pixel resolution. The block is then split into four 8×8 subblocks and each of these is given a ± 2 half pixel search around the original half-pixel block vector. The block splitting criterion from TMN is used to decide whether one or four vectors should be sent. In either case, vectors are coded differentially using a prediction scheme based on the median of neighboring vector values. Although motion vectors are found using rectangular block matching, the prediction image is formed using the overlapping motion window from H.263. For more details on these methods, consult the H.263 and TMN documents [1],[17].

An H.263 encoder normally codes intra blocks using the DCT. Since we wish to avoid DCT artifacts in the matching pursuit coder, we follow a different approach for coding intra blocks. If a 16×16 block is coded intra, the average pixel intensities from each of the six subblocks are computed. These include four 8×8 luminance subblocks and two 8×8 chrominance subblocks. These six DC intensity values are quantized to five bits each and transmitted as intra block information. The H.263 overlapping motion window is used to smooth the edges of intra blocks and thus reduce blocking effects in the prediction image. It is assumed that the additional detail coding of intra blocks will be done by the matching pursuit coder during the residual coding stage.

B. 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 the matching pursuit technique introduced in Section II. To use this method to code the motion residual signal, we must first extend the method to the discrete 2-D domain with the proper choice of a basis dictionary.

B.1 The Dictionary Set

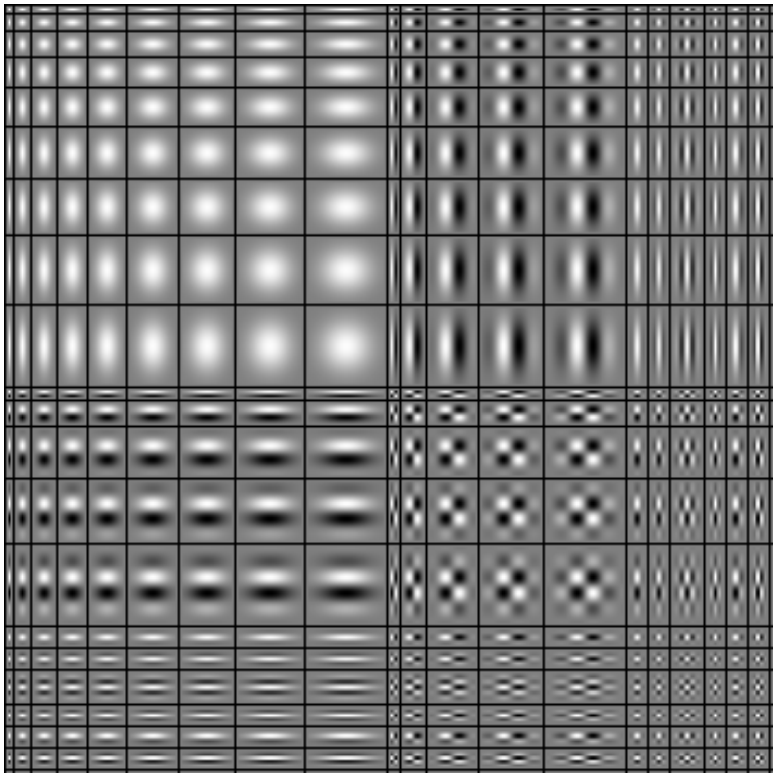

Fig. 2. The 2-D separable Gabor dictionary.

The dictionary set we use consists of an overcomplete collection of 2-D separable Gabor functions. We define this set in terms of a prototype Gaussian window:

$$g(t) = \sqrt[4]{2}e^{-\pi t^2} \tag{5}$$

We can then define 1-D discrete Gabor functions as a set of scaled, modulated Gaussian windows:

$$g_{\vec{\alpha}}(i) = K_{\vec{\alpha}} \cdot g(\frac{i - \frac{N}{2} + 1}{s}) \cdot cos(\frac{2\pi\xi(i - \frac{N}{2} + 1)}{16} + \phi) \qquad i \in \{0, 1, ...N\text{–}1\}; \tag{6}$$

Here $\vec{\alpha} = (s, \xi, \phi)$ is a triple consisting respectively of a positive scale, a modulation frequency, and a phase shift. This vector is analogous to the γ parameter of Section II. The constant $K_{\vec{\alpha}}$ is chosen such that the resulting sequence is of unit norm. If we consider $\mathcal{B}$ to be the set of all such triples $\vec{\alpha}$, then we can specify our 2-D separable gabor functions to be of the following form:

$$G_{\vec{\alpha},\vec{\beta}}(i,j) = g_{\vec{\alpha}}(i)g_{\vec{\beta}}(j) \qquad i,j \in \{0,1,...N\text{–}1\} \qquad \vec{\alpha},\vec{\beta} \in \mathcal{B} \tag{7}$$

In practice, a finite set of 1-D basis functions is chosen and all separable products of these 1-D functions are allowed to exist in the 2-D dictionary set. Table I shows how the 1-D dictionary used to generate the results of Section IV can be defined in terms of its 1-D Gabor basis parameters. To obtain this parameter set, a training set of motion residual images was decomposed using a dictionary derived from a much larger set of parameter triples. The dictionary elements which were most often matched to the training images were retained in the reduced set. The dictionary must remain reasonably small, since a large dictionary decreases the speed of the algorithm.

A visualization of the 2-D basis set is shown in Figure 2. In previously published experiments [14], [15] a fixed size of $N = 16$ pixels was used. In the current system, each 1-D function has an associated size, which allows larger basis functions to be used. An associated size also allows the basis table to be stored more efficiently and increases the search speed, since smaller basis functions are no longer padded with zeros to a fixed size. The size for each 1-D basis element is determined by imposing an intensity threshold on the scaled prototype gaussian window $g(\cdot)$ in Equation 6. Larger scale values generally translate to larger sizes, as can be seen in Table I.

Note that in the context of matching pursuits, the dictionary set pictured in Figure 2 forms an overcomplete basis for the residual image. This basis set consists of all of the shapes in the dictionary placed at all possible integer pixel locations in the residual image. To prove that this is an overcomplete set, consider a subset containing only the 1×1 pixel element in the upper left corner of Figure 2. Since this element may be placed at any integer pixel location, it alone forms the standard basis for the residual image. Using all 400 shapes from the Figure produces a much larger basis set. To understand the size of this set, consider that a complete basis set for a 176×144 QCIF image must contain 25344 elements. The block-DCT basis used by H.263 satisfies this criterion, having 64 basis functions to represent each of 396 blocks. Our basis set effectively contains 400 elements for each of the 25344 locations, for a total of more than 10 million basis elements. Using such a large set allows the matching pursuit residual coder to represent the residual image using fewer coefficients than the DCT. This is a tradeoff, since the cost of representing each element increases with the size of the basis set. Experiments have shown that this

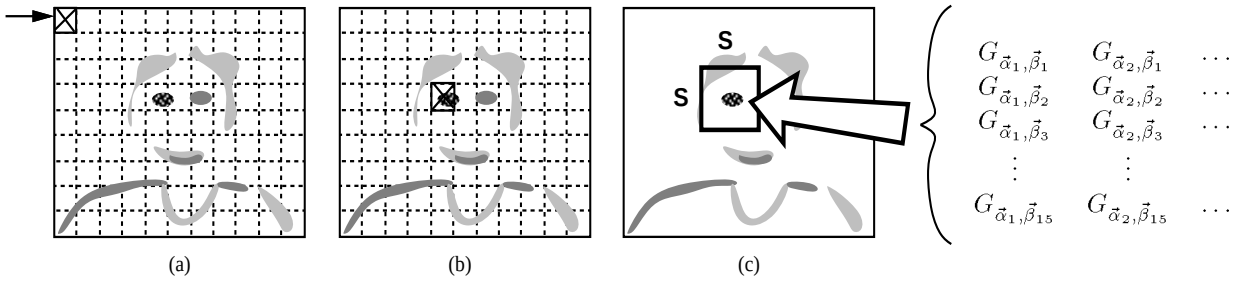

Fig. 3. Inner product search procedure. (a) Sum-of squares energy search. (b) Largest energy location becomes estimate for exhaustive search. (c) Exhaustive inner product search performed in $S \times S$ window.

tradeoff is extremely favorable at low bit rates [15].

One additional basis function is added at the frame level to allow the coder to more efficiently represent global lighting changes. This is effectively the DC component of the luminance residual frame. It is computed once per frame, and transmitted as a single 8-bit value. This increases coding efficiency on sequences which experience an overall change in luminance with time. Further research is necessary to efficiently model and represent more localized lighting changes.

B.2 Finding Atoms

The previous section defined a dictionary of 2-D structures which are used to decompose motion residual images using matching pursuits. A direct extension of the matching pursuit algorithm would require us to examine each 2-D dictionary structure at all possible integer-pixel locations in the image and compute all of the resulting inner products. In order to reduce this search to a manageable level, we make some assumptions about the residual image to be coded. Specifically, we assume that the image is sparse, containing pockets of energy at locations where the motion prediction model was inadequate. If this is true, we can "pre-scan" the image for high-energy pockets. The location of such pockets can be used as an initial estimate for the inner-product search.

The search procedure is outlined in Figure 3. The motion residual image to be coded is first divided into blocks, and the sum of the squares of all pixel intensities is computed for each block, as shown in Figure 3a. This procedure is called "Find Energy." The center of the block with the largest energy value, depicted in Figure 3b, is adopted as an initial

estimate for the inner product search. The dictionary is then exhaustively matched to an $S \times S$ window around the initial estimate, as shown in Figure 3c. In practice, we use overlapping 12×12 blocks for the "Find Energy" procedure, and a search window size of $S = 16$.

The exhaustive search can be thought of as follows. Each $N \times N$ dictionary structure is centered at each location in the search window, and the inner product between the structure and the corresponding $N \times N$ region of image data is computed. Fortunately the search can be performed quickly using separable inner product computations. This technique is shown in Section III-B.4.

The largest inner product, along with the corresponding dictionary structure and image location, form a set of five parameters, as shown in Table II. We say that these five parameters define an atom, a coded structure within the image. The atom decomposition of a motion residual image is illustrated in Figure 4. The motion residual from a sample frame of the Hall sequence is shown in Figure 4a. The first five coded atoms are shown in Figure 4b. Note that the most visually prominent features are coded with the first few atoms. Figures 4c and 4d show how the coded residual appears with fifteen and thirty coded atoms, respectively. The final coded residual, consisting of sixty-four atoms, is shown in Figure 4e. The total number of atoms is determined by the rate control system described in Section III-C. For this example, the sequence was coded at 24 kbit/s. A comparison between the coded residual and the true residual shows that the dominant features are coded but lower energy details and noise are not coded. These remain in Figure 4f, which is the final coding error image. The reconstructed frame is shown in Figure 4g, and a comparison image coded with H.263 at the same bit rate and frame rate is shown in Figure 4h.

The above atom search procedure is also used to represent color information. The "Find Energy" search is performed on each of the three component signals (Y, U, V), and the largest energy measures from the color difference signals are weighted by a constant before comparison with the largest energy block found in luminance. For the experiments presented here, a color weight of 2.5 is used. If either of the weighted color "best energy" measures exceeds the value found in luminance, then a color atom is coded.

TABLE I

DICTIONARY TRIPLES AND ASSOCIATED SIZES THAT FORM THE 1-D BASIS SET.

k	s_k	ξ_k	ϕ_k	$size$ (pixels)
0	1.0	0.0	0	1
1	3.0	0.0	0	5
2	5.0	0.0	0	9
3	7.0	0.0	0	11
4	9.0	0.0	0	15
5	12.0	0.0	0	21
6	14.0	0.0	0	23
7	17.0	0.0	0	29
8	20.0	0.0	0	35
9	1.4	1.0	$\pi/2$	3
10	5.0	1.0	$\pi/2$	9
11	12.0	1.0	$\pi/2$	21
12	16.0	1.0	$\pi/2$	27
13	20.0	1.0	$\pi/2$	35
14	4.0	2.0	0	7
15	4.0	3.0	0	7
16	8.0	3.0	0	13
17	4.0	4.0	0	7
18	4.0	2.0	$\pi/4$	7
19	4.0	4.0	$\pi/4$	7

TABLE II

PARAMETERS WHICH DEFINE AN ATOM

$\vec{\alpha}, \vec{\beta}$	Best match structure element from dictionary
x, y	Location of best match in residual image
p	Value of largest inner product; Projection of image data at (x, y) onto $G_{\vec{\alpha},\vec{\beta}}(i, j)$

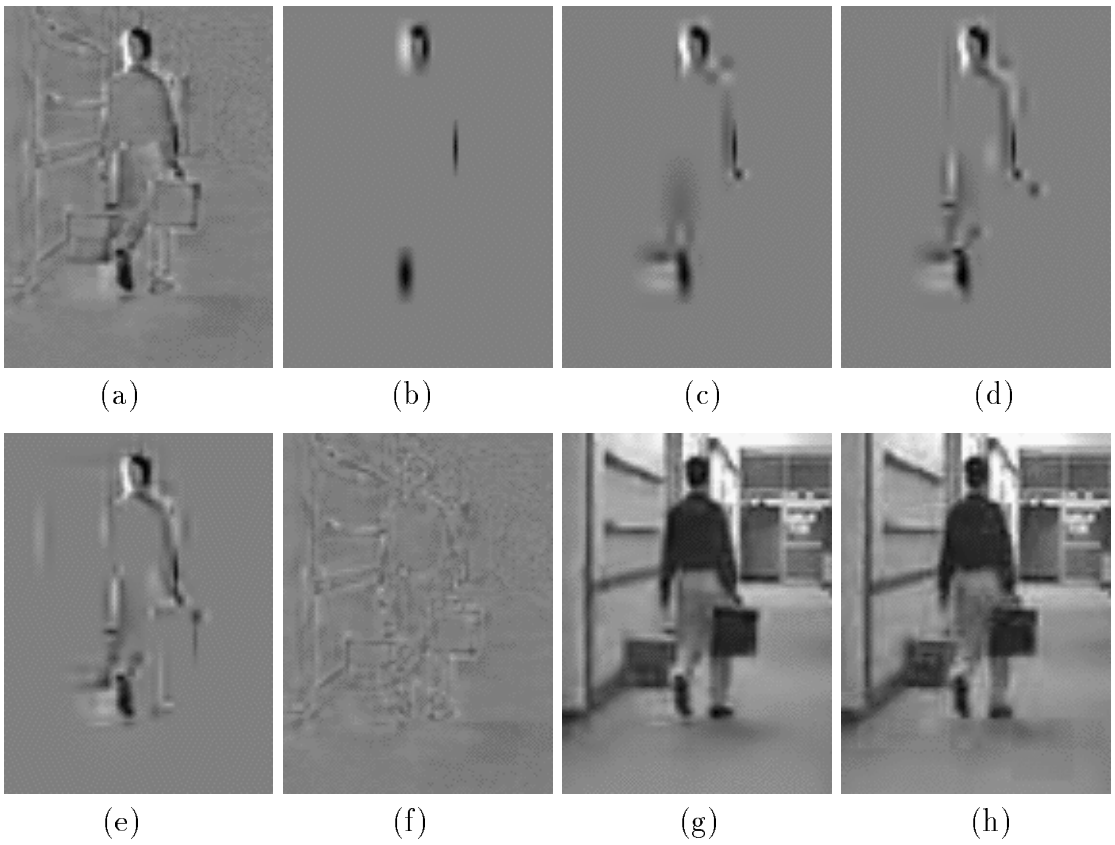

Fig. 4. Atom decomposition of Hall, frame 50. (a) Motion residual image. (b) First 5 coded atoms. (c) First 15 coded atoms. (d) First 30 coded atoms. (e) All 64 coded atoms. (f) Final coding error. (g) Reconstructed image. (h) Same frame coded by H.263.

B.3 Coding Atom Parameters

When the atom decomposition of a single residual frame is found, it is important to code the resulting parameters efficiently. The atoms for each frame are grouped together and coded in position order, from left to right and top to bottom. The positions (x, y) are specified with adaptive Huffman codes derived from the previous ten frames worth of position data. Since position data from previous frames is available at the decoder, no additional bits need be sent to describe the adaptation. One group of codewords specifies the horizontal displacement of the first atom on a position line. A second group of codes indicates the horizontal distance between atoms on the same position line, and also contains 'end-of-line' codes which specify the vertical distance to the next active position line.

The other three atom parameters are coded using fixed Huffman codes. The basis shape is specified by horizontal and vertical components $\vec{\alpha}$ and $\vec{\beta}$, each of which is represented by an index equivalent to k in Table I. Separate code tables are maintained for the horizontal and vertical shape indices. Two special escape codewords in the horizontal Y shape table are used to indicate when a decoded atom belongs to the U or V color difference signals. In all cases, the projection value p is quantized by a linear quantizer with a fixed stepsize, and transmitted using variable length codes.

To obtain the fixed huffman codes described above, it is necessary to gather statistics from previous encoder runs. For this purpose, our training set consists of the five MPEG4 Class A sequences which are listed in Section IV. In each case, we omit the sequence to be coded from the training set. When encoding the Akiyo sequence, for example, the huffman code statistics are summed from previous runs of Container, Hall, Mother and Sean. This technique allows us to avoid the case where an encoded sequence depends on its own training data.

B.4 Fast Inner Product Search

The separable nature of the dictionary structures can be used to reduce the inner product search time significantly. Recall that each stage in which an atom is found requires the algorithm to compute the inner product between each 2-D dictionary shape and the underlying image patch at each location in an $S \times S$ search window. To compute the number of operations needed to find an atom, it will first be necessary to introduce some notation describing the basis functions and their associated sizes. Suppose h and v are scalar indices into Table I representing the 1-D components of a single 2-D separable basis function. More specifically, h and v are the indices which correspond respectively to $\vec{\alpha}$ and $\vec{\beta}$ in Equation 7. The dictionary set can thus be compactly written as

$$G_{h,v}(i,j) = g_h(i)g_v(j) \qquad i \in \{0,...L_h\text{–}1\} \tag{8}$$

$$j \in \{0,...L_v\text{–}1\} \tag{9}$$

$$h,v \in \{0,...B\text{–}1\} \tag{10}$$

where B is the number of 1-D basis elements used to generate the 2-D set, and L_h and L_v

are the associated sizes of the horizontal and vertical basis components, respectively.

Suppose we ignore separability. In this case, the inner product between a given 2-D basis element and the underlying image patch can be written as

$$p = \sum_{i=0}^{L_h-1} \sum_{j=0}^{L_v-1} G_{h,v}(i,j) I(i,j) \tag{11}$$

Computing p requires $L_h L_v$ multiply-accumulate operations. Finding a single atom requires this inner product to be computed for each combination of h and v at each location in the $S \times S$ search window. The total number of operations is thus

$$T_{nonsep} = S^2 \sum_{h=0}^{B-1} \sum_{v=0}^{B-1} L_h L_v$$

Using the parameters in Table I with a search size of $S = 16$, we get 21.8 million multiply-accumulate operations.

If we now assume a separable dictionary, the inner product of Equation 11 can be written instead as

$$p = \sum_{i=0}^{L_h-1} \sum_{j=0}^{L_v-1} g_h(i) g_v(j) I(i,j) = \sum_{i=0}^{L_h-1} g_h(i) \sum_{j=0}^{L_v-1} g_v(j) I(i,j)$$

Computing a single 2-D inner product is thus equivalent to taking L_h vertical 1-D inner products, each of length L_v, and then following with a single horizontal inner product of length L_h. This concept is visualized in Figure 5. The atom search requires us to exhaustively compute the inner product at each location using all combinations of h and v. It is thus natural to pre-compute the necessary vertical 1-D inner products with a particular g_v, and then cycle through the horizontal 1-D inner products using all possible g_h. Furthermore, the results from the 1-D vertical pre-filtering also apply to the inner products at adjoining locations in the search window. This motivates the use of a large "prefiltering matrix" to store all the 1-D vertical filtering results for a particular g_v. The resulting search algorithm is shown in Figure 6. The operation count for finding a single atom is thus:

$$T_{sep} = \sum_{v=0}^{B-1} (L_v S(S + L_{max}) + S^2 \sum_{h=0}^{B-1} L_h)$$

TABLE III

COMPLEXITY OF THE H.263 MOTION SEARCH

Search Resolution	Block Size	Search Range	Motion Ops Per Block	Total Blocks
Integer	16×16	± 15	246016	99
Half Pixel	16×16	± 1	2304	99
Half Pixel	8×8	± 2	1600	396

Note that L_{max} is the size of the largest 1-D basis function. Using the values from Table I, 1.7 million multiply-accumulate operations are required to find each atom. This gives a speed improvement factor of about 13 over the general nonseparable case.

An encoding speed comparison between the matching pursuit coder and TMN [17] can be made by counting the operations required to code a single inter-frame using each method. We start with a few simplifying assumptions. First, assume that the TMN encoder is dominated by the motion prediction search, and that the DCT computation can be ignored. In this case coding a TMN frame has a complexity of Z_m, the number of operations needed to perform the motion search for a single frame. The TMN motion search can be broken down into three levels, as shown in Table III. Integer resolution and half-pixel resolution searches are first performed using the full 16×16 pixel block size. A further search using a reduced 8×8 block size is also performed at half-pixel resolution. Each search involves the repetition of a basic operation in which the difference of two pixel intensities is computed and the absolute value of the result is added to an accumulator. Consider this to be a "motion operation." The number of motion operations per block is computed in the fourth column of Table III, and the total number of blocks per frame is shown in the fifth column. From these numbers it is easy to compute $Z_m = 25.22$ million motion operations per frame. Since we are ignoring the DCT computation, this will also be the number of operations needed to encode a single TMN frame.

To model the complexity of the matching pursuit coder, we note that the same motion search is used, and we add the complexity of the atom search. The "Find Energy" search is ignored, since this search can be implemented very efficiently. With this in mind, searching

TABLE IV

THEORETICAL ENCODER COMPLEXITY COMPARISON

Target Bit Rate	Avg. Coded Atoms	H.263 Complexity*	M.P. Complexity*	Complexity Ratio
10 kbit/s	51.18	25.22	112.23	4.45
24 kbit/s	95.00	25.22	186.72	7.40

* Complexity is in millions of operations per frame, where each motion operation or multiply-accumulate is counted as a single operation.

for a single atom requires 1.7 million multiply-accumulate operations as discussed above. The number of coded atoms per frame depends on the number of bits available, as set by the target bit rate. For the comparison which follows, we average the number of atoms coded per frame in the matching pursuit experiments presented in Section IV. By multiplying the average number of coded atoms by 1.7 million operations per atom, we can measure the complexity of the atom search. This value is then added to Z_m to get the total matching pursuit algorithm complexity. For simplicity, we have assumed that a single motion operation takes the same amount of processor time as a single multiply-accumulate. The validity of this assumption depends on the hardware platform, but we have found it to hold reasonably well for the platforms we tested. Table IV shows the theoretical complexity comparison. From this table, it is evident that matching pursuit encoding is several times more complex than H.263, but the complexity difference is reduced at lower bit rates.

To show that this theoretical analysis is reasonable, we provide preliminary software encoding times for some sample runs of the TMN and matching pursuit systems. These are presented in Table V. The times are given as the average number of seconds needed to encode each frame on the given hardware platforms. Times are compared across two platforms. The first is an HP-755 workstation running HP-UX 9.05 at a clock speed of 99 MHz. The second is a Pentium machine running FreeBSD 2.2-SNAP with a clock speed of 200 MHz. The GNU compiler was used in all cases. It can be seen from the final column that the software complexity ratios are fairly close to the theoretical values seen

TABLE V

SOFTWARE ENCODER COMPLEXITY COMPARISON

		HP-755 99MHz			Pentium 200MHz		
Sequence	Rate (kbit/s)	H.263 Time*	M.P. Time*	Ratio	H.263 Time*	M.P. Time*	Ratio
Container	10	0.83	4.04	4.86	0.50	1.96	3.94
Mother	10	0.81	2.95	3.64	0.52	1.39	2.69
Container	24	0.79	7.59	9.61	0.52	3.65	7.01
Mother	24	0.81	6.58	8.12	0.52	3.13	6.01

*Time in seconds per coded frame.

in Table IV.

Note that both software encoders employ the full search TMN motion model described in Section III-A, and that the matching pursuit coder uses the exhaustive local atom search described earlier in this section. Neither encoder has been optimized for speed, and further speed improvements are certainly possible. For example, the exhaustive motion search could be replaced with a reduced hierarchical search. A similar idea could be employed to reduce the atom search time as well. Also, changes in the number of dictionary elements or in the elements themselves can have a large impact on coding speed. Further research will be necessary to exploit these ideas while still maintaining the coding efficiency of the current system.

C. Buffer Regulation

For the purpose of comparing the matching pursuit encoder to the TMN H.263 encoder, a very simple mechanism is applied to synchronize rate control between the two systems. The TMN encoder is first run using the target bit rates and frame rates listed in Section IV. The simple rate control mechanism of TMN is used, which adaptively drops frames and adjusts the DCT quantization stepsize in order to meet the target rates [17]. A record is kept of which frames were coded by the TMN encoder, and how many bits were used to represent each frame. The matching pursuit system then uses this record, encoding the same subset of original frames with approximately the same number of bits for each

frame. This method eliminates rate control as a variable and allows a very close comparison between the two systems.

The matching pursuit system is thus given a target number of bits to use for each coded frame. It begins by coding the frame header and motion vectors, subtracting the bits necessary to code this information from the target total. The remaining bits become the target residual bit budget, B_r. The encoder also computes the average number of bits each coded atom cost in the previous frame, B_{atom}. An approximation of the number of atoms to code in the current frame is computed as:

$$N_{curr} = \frac{B_r}{B_{atom}}$$

The encoder proceeds with the residual coding process, stopping after N_{curr} atoms have been coded. This approximately consumes B_r bits, and closely matches the target bit rate for the frame. In practice, the number of bits used by the matching pursuit system falls within about one percent of the H.263 bit rate.

D. Intraframe Coding

To code the first intra frame, we use the scalable subband coder of Taubman and Zakhor [19]. The source code and documentation for this subband coding system were released to the public in 1994, and can be obtained from an anonymous ftp site [16]. The package is a complete image and video codec which uses spatial and temporal subband filtering along with progressive quantization and arithmetic coding. The resulting system is scalable in both resolution and bit rate. We chose this coder because it produces high quality still images at low bit rates, and because it is free of the blocking artifacts produced by DCT-based schemes. Specifically, it produces much better visual quality than the simple intra-DCT coder of H.263 at low bit rates. For this reason, we use the subband system to code the first frame on both the matching pursuit and H.263 runs presented in the next section.

An illustration of the scalable subband coder is presented in Figure 7, which shows coded versions of Frame 0 of the Hall sequence using both the scalable subband coder and H.263 in intraframe mode. Both images are coded at approximately 15 kbits. The subband version shows more detail and is free of blocking artifacts. The H.263 coded image has

TABLE VI

NUMERICAL RESULTS FROM CLASS A SEQUENCES.

Sequence	Bit Rate kbit/s	Frame Rate frame/s	Bits to First Frame	M.P. Total Bits*	M.P. PSNR (dB)	H.263 PSNR (dB)	PSNR Diff. (dB)
Akiyo	10	7.5	7808	100291	34.05	33.82	0.23
Akiyo	24	10	20072	238668	38.79	37.94	0.85
Container	10	7.5	14096	99112	30.36	30.04	0.32
Container	24	10	19880	239203	33.35	33.00	0.35
Hall	10	7.5	12096	99639	30.84	30.63	0.21
Hall	24	10	20096	238062	35.11	34.71	0.40
Mother	10	7.5	8046	99600	32.40	32.22	0.18
Mother	24	10	19264	239602	35.76	35.59	0.17
Sean	10	7.5	9544	99529	30.24	29.68	0.56
Sean	24	10	14088	240454	34.03	33.29	0.74

* Total bits for 10 seconds of video, including first frame.

very noticeable blocking artifacts. While these could be reduced using post-filtering, such an operation generally leads to a reduction in detail.

IV. RESULTS

This section presents results which compare the matching pursuit coding system to H.263. The H.263 results were produced with the publicly available TMN encoder software [17]. As described earlier, the motion model used by the matching pursuit coder is identical to the model used in the H.263 runs, and the rate control of the two systems is synchronized. Both the matching pursuit and H.263 runs use identical subband-coded first frames. The five MPEG4 Class A test sequences are coded at 10 and 24 kbit/s. The Akiyo, Mother and Sean sequences are suitable test material for video telephone systems. The other two sequences, Container and Hall, are more suited to remote monitoring applications.

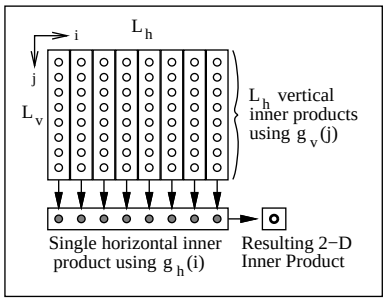

Fig. 5. Separable computation of a 2-D inner product.

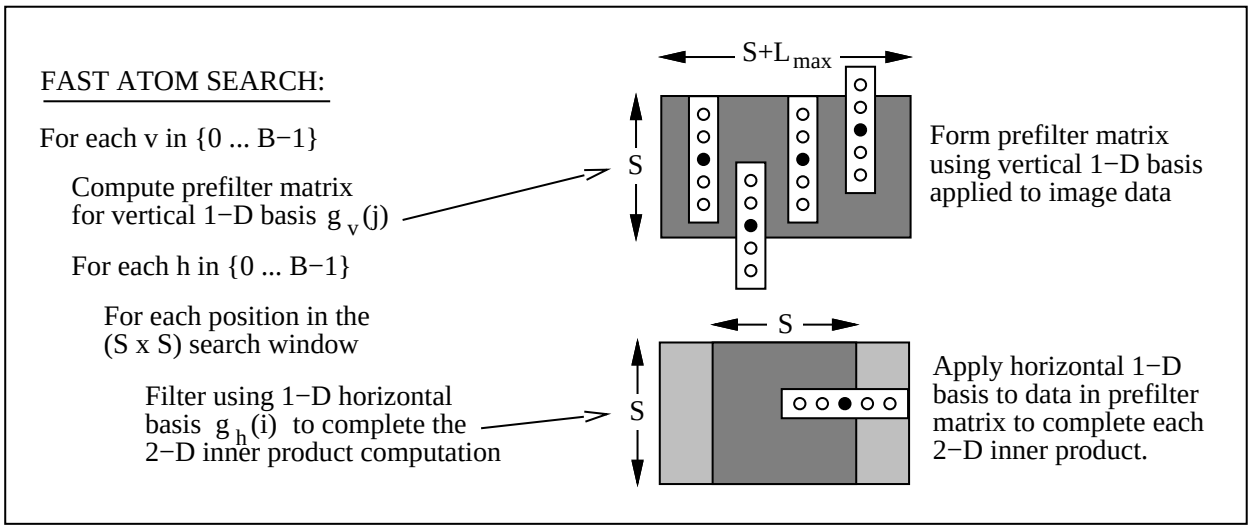

Fig. 6. The separable inner product search algorithm.

(a) (b)

Fig. 7. (a) Frame 0 of Hall coded with the H.263 intra mode using 15272 bits. (b) The same frame coded using the subband coder with 14984 bits.

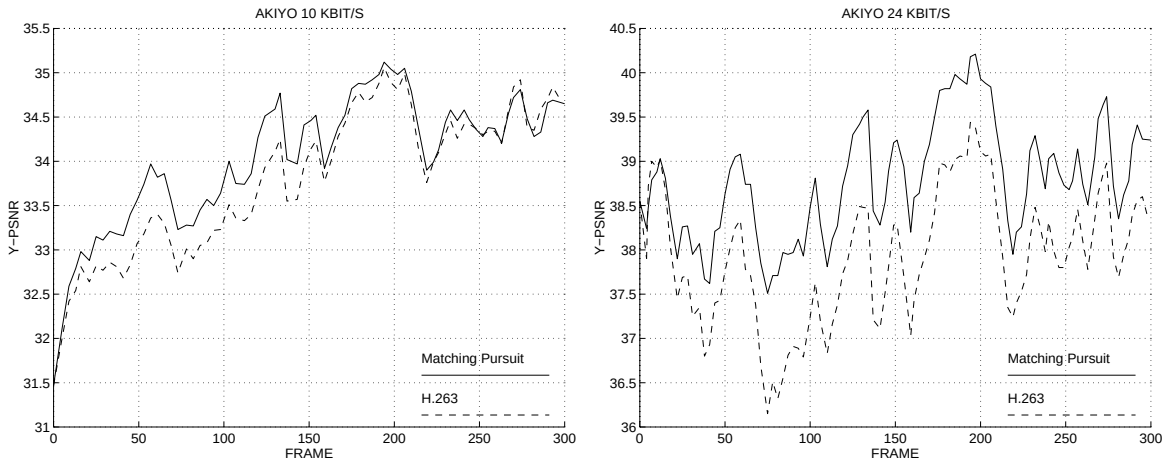

Fig. 8. PSNR comparison for Akiyo at 10 and 24 kbit/s.

Table VI shows some of the numerical results from the ten coding runs. Included are statistics on bit rate, frame rate, and the number of bits used to code the first frame in each case. Average luminance PSNR values for each experiment are also included in the table. It can be seen that the matching pursuit system has a higher average PSNR than H.263 in all the cases. The largest improvement is seen in the 24 kbit/s runs of Akiyo and Sean, which show average gains of .85 dB and .74 dB, respectively. More detail is provided in Figures 8-12, which plot the luminance PSNR against frame number for each experiment. In each case, solid lines are used to represent the matching pursuit runs and dotted lines are used for the H.263 runs. In most of the plots, the matching pursuit system shows a consistent PSNR gain over H.263 for the majority of coded frames. Performance is particularly good for Container, Sean, and the 24 kbit/s runs of Akiyo and Hall. The least improvement is seen on the Mother sequence, which shows a more modest PSNR gain and includes periods where the performance of the two coders is approximately equal. One possible explanation is that the Mother sequence contains intervals of high motion in which both coders are left with very few bits to encode the residual. Since the residual encoding method is the main difference between the two encoders, a near equal PSNR performance can be expected.

The matching pursuit coder shows a visual improvement over H.263 in all the test cases. To illustrate the visual improvement, some sample reconstructed frames are shown in Figures 13-16. In each case, the H.263 coded frame is shown on the left, and the

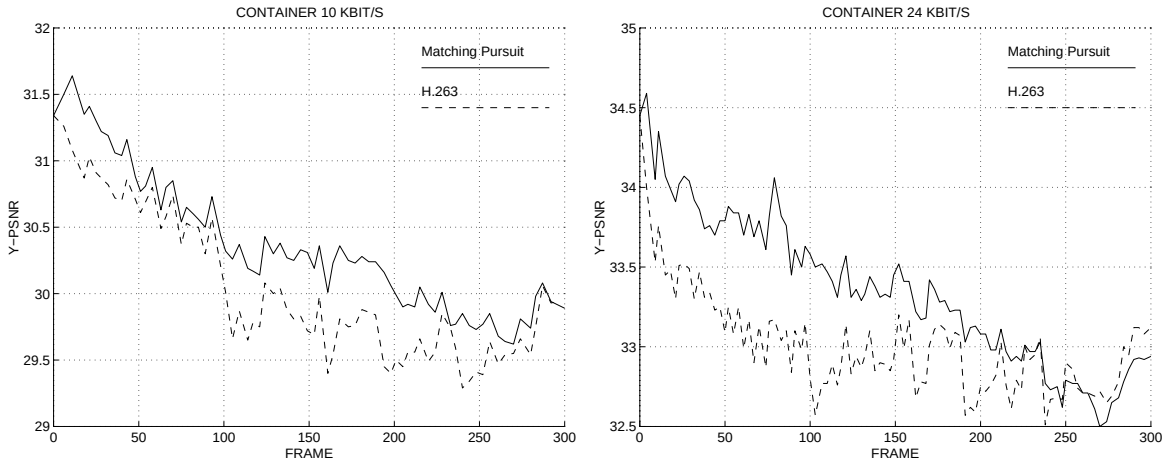

Fig. 9. PSNR comparison for Container at 10 and 24 kbit/s.

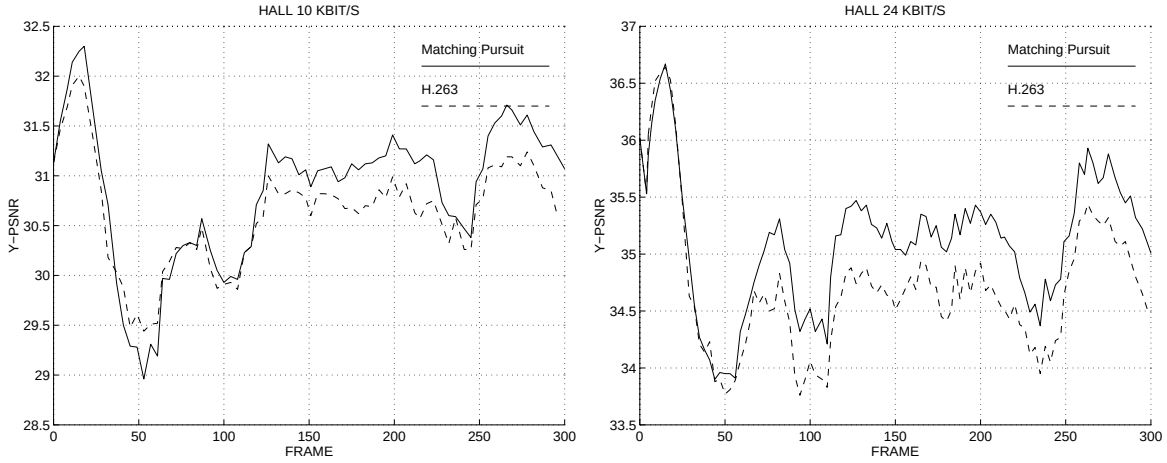

Fig. 10. PSNR comparison for Hall at 10 and 24 kbit/s.

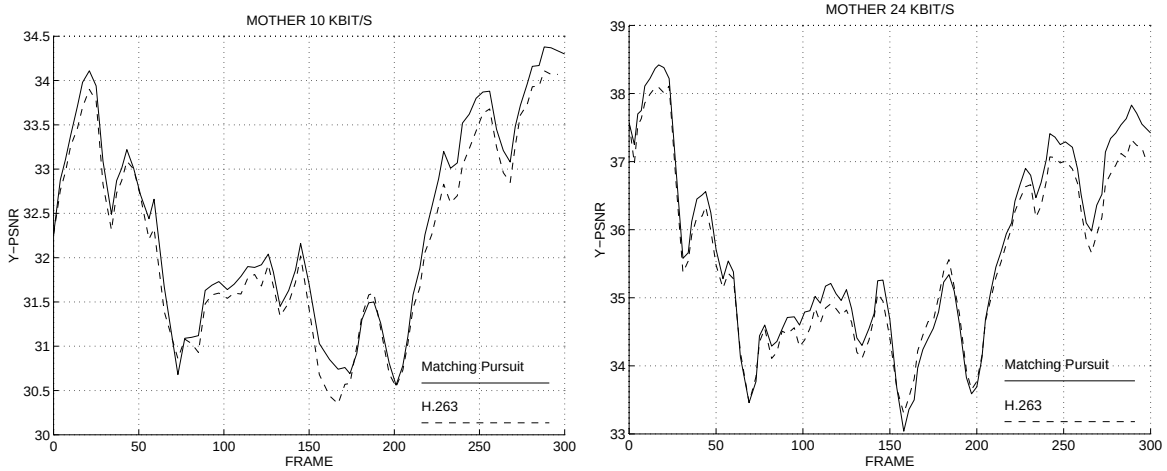

Fig. 11. PSNR comparison for Mother at 10 and 24 kbit/s.

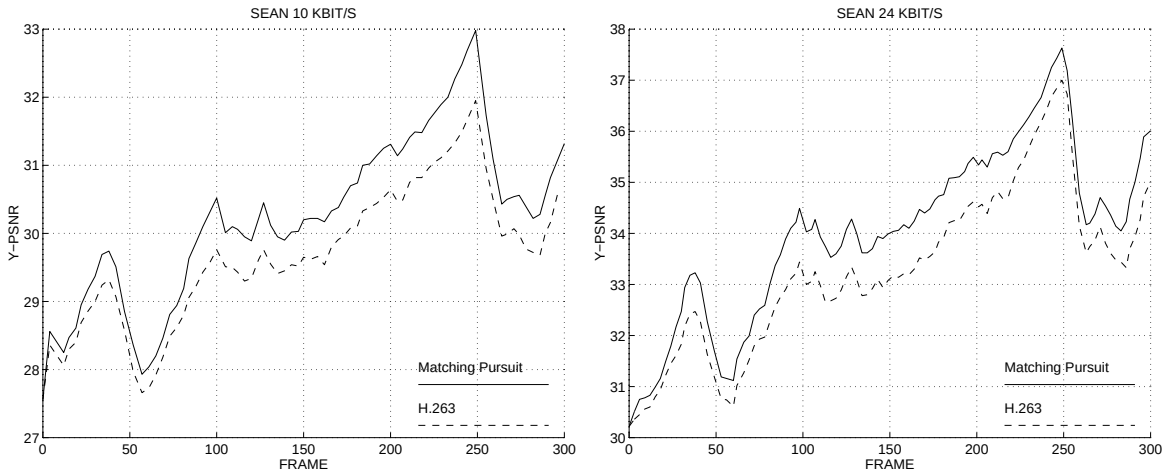

Fig. 12. PSNR comparison for Sean at 10 and 24 kbit/s.

corresponding matching pursuit frame is on the right. Note that all of the photographs except for Container show enlarged detail regions. Figure 13 shows a sample frame of Akiyo coded at 10 kbit/s. Note that the facial feature details are more clearly preserved in the matching-pursuit coded frame. The H.263 frame is less detailed, and contains block edge artifacts on the mouth and cheeks, as well as noise around the facial feature outlines. Figure 14 shows a frame of Container coded at 10 kbit/s. Both algorithms show a lack of detail at this bit rate. However, the H.263 reconstruction shows noticeable DCT artifacts, including block outlines in the water, and moving high frequency noise patterns around the edges of the container ship and the small white boat. Moving objects in the matching pursuit reconstruction appear less noisy and more natural, and the water and sky are much smoother. Figure 15 shows a frame of Hall coded at 24 kbit/s. Again, the matching pursuit image is more natural looking, and free of high frequency noise. Such noise patterns surround the two walking figures in the H.263 frame, particularly near the legs and briefcase of the foreground figure and around the feet of the background figure. This noise is particularly noticeable when the sequence is viewed in motion. Finally, Figure 16 shows a frame of the Mother sequence coded at 24 kbit/s. The extensive hand motion in this sequence causes the H.263 coded sequence to degrade into a harsh pattern of blocks. The matching pursuit sequence develops a less noticeable blur. Notice also that the matching pursuit coder does not introduce high frequency noise around the moving outline of the mother's face, and the face of the child also appears more natural.

(a) (b)

Fig. 13. Akiyo frame 248 encoded at 10 kbit/s by (a) H.263, (b) Matching Pursuits.

(a) (b)

Fig. 14. Container frame 202 encoded at 10 kbit/s by (a) H.263, (b) Matching Pursuits.

(a) (b)

Fig. 15. Hall frame 115 encoded at 24 kbit/s by (a) H.263, (b) Matching Pursuits.

(a) (b)

Fig. 16. Mother frame 82 encoded at 24 kbit/s by (a) H.263, (b) Matching Pursuits.

V. Conclusions

When coding motion residuals at very low bit rates, the choice of basis set is extremely important. This is because the coder must represent the residual using only a few coarsely quantized coefficients. If individual elements of the basis set are not well matched to structures in the signal, then the basis functions become visible in the reconstructed images. This explains why DCT based coders produce block edges in smoothly varying image regions and high-frequency noise patterns around the edges of moving objects. Wavelet based video coders can have similar problems at low bit rates. In these coders, the wavelet filter shapes can be seen around moving objects and edges. In both cases, limiting the basis to a complete set has the undesirable effect that individual basis elements do not match the variety of structures found in motion residual images.

The matching pursuit residual coder presented in this paper lifts this restriction by providing an expansion onto an overcomplete basis set. This allows greater freedom in basis design. For example, the separable Gabor basis set used to produce the results of this paper contains functions at a wide variety of scales and arbitrary image locations. This flexibility insures that individual atoms are only coded when they are well matched to image structures. The basis set is "invisible" in the sense that it does not impose its own artificial structure on the image representation. When bit rate is reduced, objects coded using matching pursuits tend to lose detail, but the basis structures are not generally seen

in the reconstructions. This type of degradation is much more graceful than that produced by hybrid-DCT systems.

Our results demonstrate that a matching-pursuit based residual coder can outperform an H.263 encoder at low bit rates in both PSNR and visual quality. Future research will include increasing the speed of the algorithm and optimizing the basis set to improve coding efficiency.

References

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

[2] H. G. Musmann, M. Hotter and J. Ostermann, "Object-Oriented Analysis Synthesis Coding of Moving Images," *Signal Processing: Image Communication,* October 1989, Vol. 1, No. 2, pp. 117-138.

[3] M. Gilge, T. Engelhardt and R. Mehlan, "Coding of Arbitrarily Shaped Image Segments Based on a Generalized Orthogonal Transform," *Signal Processing: Image Communication,* October 1989, Vol. 1, No. 2, pp. 153-180.

[4] S. F. Chang and D. Messerschmitt, "Transform Coding of Arbitrarily Shaped Image Segments," Proceeedings of 1st ACM International Conference on Multimedia, Anaheim, CA, 1993, pp.83-90.

[5] T. Sikora, "Low Complexity Shape-Adaptive DCT for Coding of Arbitrarily-Shaped Image Segments," *Signal Processing: Image Communication,* Vol. 7, No. 4, November 1995, pp. 381-395.

[6] S. Mallat and Z. Zhang, "Matching Pursuits With Time-Frequency Dictionaries," *IEEE Transactions in Signal Processing,* Vol. 41, No. 12, December 1993, pp. 3397-3415.

[7] J. H. Friedman and W. Stuetzle, "Projection Pursuit Regression," *Journal of the American Statistical Association,* Vol. 76, No. 376, December 1981, pp. 817-823.

[8] L. Wang and M. Goldberg, "Lossless Progressive Image Transmission by Residual Error Vector Quantization," *IEE Proceedings,* Vol. 135, Pt. F, No. 5, October 1988, pp. 421-430.

[9] F. Bergeaud and S. Mallat, "Matching Pursuit of Images," Proceedings of IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis, October 1994, pp.330-333.

[10] M. Vetterli and T. Kalker, "Matching Pursuit for Compression and Application to Motion Compensated Video Coding," Proceedings of ICIP, November 1994, pp. 725-729.

[11] P. J. Phillips, "Matching Pursuit Filters Applied to Face Identification," Proceedings of SPIE, Vol. 2277, July 1994, pp. 2-11.

[12] H. G. Feichtinger, A. Türk and T. Strohmer, "Hierarchical Parallel Matching Pursuit," Proceedings of SPIE, Vol. 2302, July 1994, pp. 222-232.

[13] T. -H. Chao, B. Lau and W. J. Miceli, "Optical Implementation of a Matching Pursuit for Image Representation," *Optical Engineering,* July 1994, Vol. 33, No. 2, pp. 2303-2309.

[14] R. Neff, A. Zakhor, and M. Vetterli, "Very Low Bit Rate Video Coding Using Matching Pursuits," Proceedings of SPIE VCIP, Vol. 2308, No. 1, September 1994, pp. 47-60.

[15] R. Neff and A. Zakhor, "Matching Pursuit Video Coding at Very Low Bit Rates," IEEE Data Compression Conference, Snowbird, Utah, March 1995, pp. 411-420.

[16] D. Taubman, "Fully Scalable Image and Video Codec," Public software release, available via anonymous ftp from tehran.eecs.berkeley.edu under /pub/taubman/scalable.tar.Z.

[17] ITU Telecommunication Standardization Sector LBC-95, "Video Codec Test Model TMN5." Available from Telenor Research at http://www.nta.no/brukere/DVC/

[18] C. Auyeung, J. Kosmach, M. Orchard and T. Kalafatis, "Overlapped Block Motion Compensation," Proceedings of SPIE VCIP, November 1992, Vol. 1818, No. 2, pp.561-572.

[19] D. Taubman and A. Zakhor, "Multirate 3-D Subband Coding of Video," *IEEE Transactions on Image Processing,* September 1994, Vol. 3, No. 5, pp. 572-88.