4

Efficient Scalable Video Coding Based on Matching Pursuits

Jian-Liang Lin[1] and Wen-Liang Hwang[2]
[1]*MediaTek Inc.*
[2]*Institute of Information Science, Academia Sinica, Taiwan and the Department of Information Management at Kainan University, Taiwan*

1. Introduction

Efficient encoding of motion residuals is essential for low-delay video applications in which videos are encoded by hybrid motion compensation and a residual encoding structure. Current standard video coding systems use hybrid motion compensation and the discrete cosine transform (DCT), where the number of bases needed to encode a residual block is the same as the number of the pixels in the block. An encoded frame is predicted from its previous reconstructed frame, and a residual image is then encoded by a non-redundant transformation, such as the DCT, or an approximation of the DCT using integer coefficients.
As well as nonredundant transformation, a frame-based technique called matching pursuit (MP) has been proposed to encode motion residual images. Mallat and Zhang (Mallat & Zhang, 1993) were the first to propose a matching pursuit algorithm that decomposes a signal into a linear combination of bases within an overcomplete dictionary. Vetterli and Kalker have translated motion compensation and DCT hybrid video coding into a matching pursuit technique, and encoded frames by the matching pursuit algorithm and a dictionary composed of motion blocks and DCT bases (Vetterli & Kalker, 1994). In (Neff & Zakhor, 1997), Neff and Zakhor show that using a matching pursuit algorithm to encode motion residual images achieves a better performance than a DCT-based algorithm in terms of PSNR and perceptual quality at very low bit rates. The results in (Lin et al., 2005) also demonstrate that the matching pursuit FGS coding scheme performs better than MPEG-4 FGS at very low bit rates. Unlike a transform-based decoder, a matching pursuit decoder does not require an inverse transform; therefore, it is less complex. In a transform-based decoder, loop filtering and post processing are usually applied at very low bit rates to remove blocking and ringing artifacts, whereas a matching pursuit decoder can achieve comparable quality without such filtering and processing (Neff et al., 1998). Because the matching pursuit algorithm is a data-dependent frame-based representation, a matching pursuit video coding technique cannot be directly translated from conventional transform-based approaches. We must therefore develop a new matching pursuit video coding technique that can deal with quantization noise in the matching pursuit algorithm (Neff & Zakhor, Sept. 2000; Vleeschouwer & Zakhor, 2002), multiple description coding for reliable transmission (Tang & Zakhor, 2002), scalable bit-stream generation (Al-Shaykh et al., 1999; Vleeschouwer & Macq, 2000; Rose & Regunathan, 2001), and dictionary learning and adaptation (Engan et al., 2000).

In real video communication applications over wire and wireless networks, channel capacity varies constantly, so scalability at the bit-stream level is an important feature of multimedia communications. It is necessary, therefore, to develop a video coding and transmission technique that encodes a video sequence once, and allows different clients to decode the video by receiving only part of the bit-stream, according to their available bandwidth. Depending on the scalability specification, a scalable video coding scheme can support scalability either in the frame rate, video resolution, SNR quality, or a hybrid of these. Based on a hybrid motion compensation and bit-plane DCT coding scheme, MPEG-4 scalable video coding is proposed as a means of achieving fine-grain scalability (Li, 2001). The drawback of this approach is that, to avoid the drifting problem, it only uses the information of the base layer to predict the next frame; therefore, it yields lower coding efficiency than non-scalable video coding schemes at the same bit-rate. Recently, several approaches based on motion-compensated temporal filtering (MCTF) have been proposed to improve coding efficiency and solve the drifting problem encountered in closed-loop hybrid coding systems (Ohm et al., 2004; M10569/S03, 2004). Although, the hybrid motion-compensation and residual encoding scheme may not be the best solution for video scalability, it is simple and only incurs a small delay in performing motion-compensation compared to a scalable video coding scheme based on MCTF. Even so, current approaches based on hybrid motion compensation and residual encoding schemes using the DCT are still inefficient, especially at low bit-rates.
Scalable video coding schemes based on matching pursuit have been proposed as a means of achieving scalability (Al-Shaykh et al., 1999; Vleeschouwer & Macq, 2000). In (Al-Shaykh et al., 1999), scalability is supported by successively en-coding groups of atoms; thus, it is constricted by the number of atoms determined by the encoder. Generally, providing video coding with scalability degrades the coding performance. A two-layer video coding scheme achieves better coding efficiency at the expense of coarser scalability. In a two-layer system, the base layer gives a lower quality of coded video, while the enhancement layer gives a higher quality; moreover, the information of the enhancement layer is used in motion-compensation to achieve better coding efficiency.

2. Atom search

As mentioned in the previous section, matching pursuit (MP) is a greedy algorithm that decomposes a signal into a linear combination of bases within an overcomplete dictionary. The matching pursuit algorithm is usually only approximated due to its massive computational complexity.
An MP-based codec yields a better PSNR and perceptual quality than a transform-based codec, and its decoder is simpler (Neff et al., 1998). However, it cannot be used in applications that require real time bi-directional communications, because the encoder consumes a massive amount of computation time. A matching pursuit encoder does not obtain all the coefficients in one step, but iteratively finds the frame coefficient that has the largest absolute inner product value between a residual and all the bases. The inner product value and the base from which the value is obtained are called an atom. Many approaches have been proposed to simplify the complex encoding stage. One approach approximates the codewords of a dictionary with a linear combination of simpler codewords so that the computation is easier (Redmill et al., 1998; Czerepi´nski et al., 2000; Neff & Zakhor, 2002; Vleeschouwer and Macq, 1999; Jeon & Oh, 2003). This technique can be further developed by combining the inner product calculation and the atom finding components (Lin et al., 2007).

Another method pre-calculates and stores all the inner products between bases so that the encoder can update the inner products with the pre-calculated values of the bases, in-stead of re-calculating the inner products between a residual and the bases at each iteration (Mallat & Zhang, 1993). This is an efficient way to decompose a one-dimensional signal. However, it is totally unsuitable for video coding, because there are too many inner products between the bases. In the popular Gabor dictionary used in matching pursuit video encoding, there are 400 codewords, each of which is at most 35 by 35 pixels. Consequently, the inner products between the bases need at least a 30 giga-byte memory (assuming four bytes for a real value).
This difficulty prevents the matching pursuit algorithm achieving its best performance. The most popular approach for finding an atom is that proposed by Neff and Zakhor (Neff & Zakhor, 1997)], whereby a residual frame is divided into blocks and, at each iteration, an atom is found within the block with the highest energy. This approach is modified in (Al-Shaykh et al., 1999), which gives an energy weight to each block so that the more atoms chosen from a block, the smaller the energy weight of that block will be. Therefore, the block is less likely to be chosen in later iterations. The energy-weight approach reduces the likelihood that the majority of atoms will be selected from a few blocks, and improves the PSNR performance of Neff and Zakhor's algorithm.

2.1 Matching pursuit algorithm and atom extraction

There are many ways to decompose a signal into an overcomplete base set (dictionary). However, obtaining the best linear combination of bases is an NP-hard problem (Davis, 1994). The matching pursuit algorithm is a frame-based approach that represents a signal by a succession of greedy steps (Mallat & Zhang, 1993). At each iteration, the signal is projected onto the base that approximates the signal most efficiently. Let D be a dictionary of over-complete image bases $\{g_\gamma(x)\}$, where γ is the index. The algorithm decomposes an image into a linear expansion of the bases in the dictionary by a succession of greedy steps as follows.
The image f(x) is first decomposed into

$$f(x) =< f(x), g_{\gamma_0}(x) > g_{\gamma_0}(x) + Rf(x), \tag{2.1}$$

where $g_{\gamma_0}(x) = \arg_{g_\gamma(x)\in D}\max\{| < f(x), g_\gamma(x) > |\}$ and Rf(x) is the residual image after approximating f(x) in the direction of $g_{\gamma_0}(x)$. The dictionary element $g_{\gamma_0}(x)$ together with the inner product value $< f(x), g_{\gamma_0}(x) >$ is called an *atom*. The matching pursuit algorithm then decomposes the residual image Rf(x) by projecting it onto a basis function of D, which is the same as the process for f(x). After M iterations, an approximation of the image f(x) can be obtained from the M atoms by

$$\tilde{f}_M(x) = \sum_{k=0}^{M-1} < R^k f(x), g_{\gamma_k}(x) > g_{\gamma_k}(x) \tag{2.2}$$

and $\tilde{f}_M(x)$ converges strongly to f(x) as $M \to \infty$.
The matching pursuit algorithm decomposes the signal structure according to the order of importance; therefore, the most significant structures are likely to be extracted first. As the dictionary is redundant, the reconstruct signal will converge to signal f(x) after sufficient iterations. The rate of convergence is dependent on the statistics of the signal and the choice of dictionary. Neff and Zakhor apply the algorithm to encode the motion residual images in video coding by using a 2-D separable Gabor dictionary to decompose the residual image.

For each iteration, a basis is selected from the dictionary to match the residual image. The corresponding inner product, basis index, and location are defined as an atom.
Instead of recalculating the inner products at each iteration, Mallat and Zhang (Mallat & Zhang, 1993) developed the matching pursuit update algorithm. At the kth iteration, let

$$g_{\gamma_k} = \max_{\gamma} | < R^k f(x), \ g_\gamma > | \tag{2.3}$$

be the base of the largest absolute inner product value. The new residual signal $R^{k+1}f$ is

$$R^{k+1}f(x) = R^k f(x) - < R^k f(x), g_{\gamma_k} > \ g_{\gamma_k}. \tag{2.4}$$

The inner products between $R^{k+1}f(x)$ and the bases $\{g_\gamma\}$ are represented by

$$< R^{k+1}f(x), g_\gamma > = < R^k f(x), g_\gamma > - < R^k f(x), g_{\gamma_k} > < g_{\gamma_k}, g_\gamma >. \tag{2.5}$$

Because $< R^k f(x), g_\gamma >$ and $< R^k f(x), g_{\gamma_k} >$ were calculated in the previous iteration, and if $< g_{\gamma_k}, g_\gamma >$ is pre-calculated, this update operation only needs one addition and one multiplication. However, the algorithm needs a huge amount of space to store all non-zero $< g_{\gamma_k}, g_\gamma >$ in an image and it is only practical for one-dimensional signal decomposition. Thus, the matching pursuit update algorithm cannot be used in video encoding. Instead, the proposed approach in (Neff & Zakhor, 1997; Al-Shaykh et al., 1999) divides a residual into blocks and, at each iteration, the matching pursuit algorithm is applied to the block with the highest energy. This approach is both simple and efficient, and has been implemented in many MP-based video codecs.
As the above algorithms find an atom from the largest (weighted) energy block, we call them one-block algorithms. These approaches are simple and efficient, but their coding performance may be unsatisfactory. Although the performance can be improved by finding an atom from more than one block, there is still the issue of the massive number of inner products between a residual and the bases in the blocks. To solve this problem, we approximate a residual in a subspace, spanned by a small number of bases within a few blocks (Lin et al., March 2006). The bases and the blocks are selected according to the content of the residual, while the coding performance and efficiency are determined by the number of bases and the number of blocks. Simulations show that the algorithm achieves better subjective and objective performances and requires less run-time than one-block algorithms for various sequences at low bit-rates.
Since the bases in the dictionary are redundant, as the number of iterations increases, the redundancy in linear combinations also increases. To solve this problem, an orthogonal matching pursuit algorithm has been proposed to reduce the redundancy between bases (Davis, 1994). At each iteration, the projection of bases on the selected orthogonal basis is removed, but it does not normalize the orthogonal basis; therefore, the basis selection unfair. To ensure that all bases have the same norm, instead of selecting the maximum absolute inner product between a signal and the orthogonal basis, we have proposed an orthonormal matching pursuit algorithm that finds the maximum absolute inner product after normalization (Lin et al., May 2006). Using the orthonormal matching pursuit algorithm, a signal can be approximated efficiently with a linear combination of selected bases so that the redundancy between the selected bases can be successfully removed.

2.2 Dictionary design

Since matching pursuit codecs are asymmetrical (as the complexity of the decoder is low, while the complexity of the encoder is extremely high), the dictionary design is a very important issue. It affects the complexity of the encoder, as well as the subjective and objective performance of matching pursuit video codec. The traditional dictionary design for reducing complexity is the 2-D separable dictionary (Neff & Zakhor, 1997). To further reduce the complexity of this dictionary, another approach has been proposed to approximate it with a low cost factorized separable dictionary in which larger basis functions in the 2-D separable dictionary are represented as a successive convolution of short basis functions (Vleeschouwer & Macq, 1999; Czerepi´nski et al., 2000). Although the complexity is drastically reduced, the disadvantage of this approach is that the dictionary is restricted within a 2-D separable set; thus, the design of the dictionary may not be optimal in terms of coding performance.

To overcome the restrictions of the 2-D separable dictionary, another dictionary design has been proposed to generate a non-separable 2-D dictionary by approximating any arbitrary basis function by a linear combination of elementary functions (Redmill et al., 1998; Neff & Zakhor, 2002). In this approach, the elementary dictionary and the order of bases are usually chosen heuristically, which affects the coding performance. Therefore, developing a systematic dictionary design approach to approximate any arbitrary dictionary is essential. In (Lin et al., 2007), we propose a systematic dictionary approximation scheme by using the most important eigenfunctions of the target dictionary to approximate each basis in the dictionary. Since the structures of eigenfunctions may be very complex, we transform these eigenfunctions into a dyadic wavelet domain and approximate each eigenfunction by the most significant dyadic wavelet transform coefficients. This framework allows a trade-off between the approximation quality and the complexity, depending on the number of eigenfunctions and the number of wavelet coefficients used to construct the approximated dictionary. Associated with a treebased search algorithm in atom selection, a combined efficient dictionary design structure and atom extraction scheme is thus proposed.

3. Scalable video coding

As the development of multimedia applications grow, video techniques have been gradually changing from one-to-one (simulcast) to one-to-many (multicast) communications. Due to channel capacity variation and disparate requirements for different receivers, it is necessary to develop video coding and transmission techniques that are efficient and scalable to Internet heterogeneity. Although representing a video with multiple redundancy in different bit rates is a simple solution for multicasting in most commercial systems, this approach cannot efficiently cope with channel capacity variation (McCanne et al., 1997; Li, 2001). In contrast, video scalability is a better solution as it generates a single bit-stream for all intended recipients, and each decoder can reconstruct a varied quality video within a specific bit rate range. Depending on the specification of receivers, a scalable system can support scalability either in frame rate (temporal scalability), frame resolution (spatial scalability), frame quality (SNR scalability) or a hybrid of these (hybrid scalability). Despite the fact that many scalable coding methods have been developed in recent years, they are still inefficient, especially at low bit rates (Girod et al., 2002). Most of the existing systems use hybrid motion-compensated DCT for video coding. Although the hybrid motion-compensation algorithm may not be the best solution for video scalability, the

hybrid scheme is simple, efficient, and has a small delay in performing frame prediction. In our study, we focus on developing SNR scalability algorithms at low bit rates in a hybrid motion-compensation video coding system in which the frame residuals are encoded using matching pursuits.

Certain matching pursuit SNR-scalable schemes have been proposed in (Al-Shaykh et al., 1999; Vleeschouwer & Macq, 2000). An FGS produces a continuous bit-stream with increasing PSNR for a wide range of bit rates. A matching pursuit FGS coding algorithm is presented in (Al-Shaykh et al., 1999) in which enhancement layer scalability is achieved by successively encoding groups of atoms in which the number of atoms in a group is the primary parameter controlling scalability. A better coding efficiency than FGS can be obtained at the expense of coarser scalability. A two-layer system is coarse scalable because the bit-stream does not provide a continuous quality improvement over a wide range of bit rates. The lower layer delivers minimal bit rates while the upper layer delivers the highest possible quality. An estimation-theoretic approach to improve the performance of a two-layer system is proposed in (Rose & Regunathan, 2001) in which prediction was made from the previous base layer and enhancement layer.

Both our SNR FGS video codec and our two-layer SNR scalable video codec are based on successive bit-plane quantization coding of the atoms selected from motion residuals using matching pursuits (Lin et al., 2005). The proposed FGS algorithm uses bit-plane coding and uses the spatial and temporal dependence between bit-planes to exploit the redundancy in the bit-planes. The efficiency of our algorithm in encoding atom positions lies in using quadtree to represent a bit-plane and to perform bit-plane prediction. Our experiments indicate that our algorithm can achieve a 1 to 2 bits reduction in encoding atom positions to that of the theoretical lower bound given in (Al-Shaykh et al., 1999). This bound is derived from the assumption that atom positions are uniformly and identically distributed random variables in each frame. The bound can be out-performed if the redundancy of atom positions is exploited. We then combine position coding and progressive refinement of atom modula in our matching pursuit FGS structure.

A two-layer scalable system is able to decide which layer to emphasize without changing bit rates. If available bandwidth is full most of the time, a bit-stream is generated by using more information from the residuals of the high quality (enhancement) layer. Likewise, if it is limited, the residuals of the low quality (base) layer are emphasized. Using a linear combination of base layer and enhancement layer residuals in a two-layer scalable system has attracted much attention since it was published in (Al-Shaykh et al., 1999). Following this combination approach, we propose the use of a combined frame obtained from a linear combination of the reconstructed base layer and enhancement layer images to estimate motion vectors. Since both base and enhancement layer reconstructed images are used in our motion vector estimation, a better performance is attained in the enhancement layer (Lin et al., 2005).

4. Progressive atom coding

4.1 Set partitioning coding of atoms

When motion residual images are encoded using matching pursuits, the coding efficiency lies in economically encoding atoms. An atom includes an inner product value as well as a basis function containing location and index. Following, we will illustrate an algorithm which can efficiently encode atoms progressively.

Following the well-known set partitioning strategy adopted in the EZW (Shapiro, 1993) and SPIHT (Said & Pearlman, 1996) algorithms, we apply the bit-plane based successive-approximation quantization (SAQ) on the inner product values to encode the atoms selected from a motion residual using matching pursuits. This algorithm will successively improve the resolution of a residual frame from many scans of the bit-planes to find new significant atoms and refine the values of existing significant atoms. Initially, all atoms are assumed insignificant. Let the N_{BP}-th bit-plane be the most significant bit-plane of all the atoms. At the i-th step, a threshold whose value is set to $2^{N_{BP}-i}$ is compared to the inner products of insignificant atoms. An insignificant atom becomes significant when its absolute inner product value is larger than the current threshold. The positions and the index of the basis function and the sign of the inner product of the atom are encoded, and then the atom is added to the significant set (also called *AtomList*). The atom will remain in the significant set and the atom's inner product value will be successively refined by the following refinement steps.

When a matching pursuit is used together with a bit-plane based SAQ, the positions of atoms must be carefully handled to ensure coding efficiency. The energy of a transformed-based coding method is usually concentrated in some bases. This occurs in low-low frequency bands for DCT transform, and in coarser scales for a wavelet transform. There are efficient bit-plane scanning orders for both DCT and wavelet transform. The left subfigure of Figure 1 shows one DCT method using a zig-zag order. The middle subfigure shows one wavelet transform method using a tree order. Unlike the transformed-based coding methods, the atoms of a matching pursuit can be randomly positioned in a residual frame, as shown in the rightmost subfigure. The energies of atoms are scattered over the residual frame according to the contents in the frame. Thus, neither DCT zig-zag ordering nor wavelet tree ordering can encode the atom positions efficiently. In consequence, we should develop an efficient scanning order for the atom positions in a bit-plane to attain better coding efficiency.

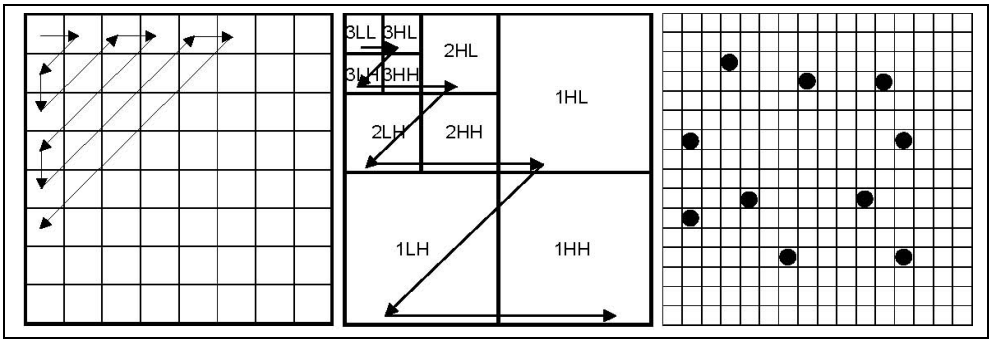


Fig. 1. Left:DCT scan-order. Middle:EZW scan-order. Right:Atom positions.

4.2 Quadtree and quadtree prediction of atom position

Position coding efficiency is dependent on the number of atoms in a bit-plane. To specify this dependency, a theoretical lower bound for atom positions is proposed in (Al-Shaykh et al., 1999) as a comparison reference for atom position encoding algorithms. The lower bound is derived by assuming that atoms are uniformly and independently distributed on

an $N_1 \times N_2$ image and that no pixel of the image has more than one atom. The latter assumption is valid at low bit rates since only a few atoms are selected in a residual, and the probability that more than one atom exists at each pixel location is low. Our simulation on sequences Akiyo and Container at low bit rates shows that the probability that more than one atom are selected at the same location is less than 1%. If there are n atoms on the image, the entropy for encoding the positions of an atom will be $\log 2 \binom{N_1 \times N_2}{n} / n$. Note that atoms usually distribute non-uniformly in residuals. This bound can be out-performed if an atom position encoding algorithm takes advantage of non-uniformity in atom distribution and removes the redundancy among them.

In (Al-Shaykh et al., 1999), the *NumberSplit* algorithm is presented to encode atom positions built on a multidimensional searching algorithm. In this algorithm, the total number of atoms of a residual frame is decided and given to a decoder. A frame is then separated into two halves, and the number of atoms in the first half is given to the decoder. The number is entropy-coded by an adaptive Huffman table which is built according to the number of atoms to be coded at each frame. The decoder can use the total number of atoms and the number of atoms in the first half to obtain the number of atoms in the other half. Each half is further divided recursively until it reaches either one pixel or a region that contains no atom. In (Al-Shaykh et al., 1999), atoms tend to cluster around regions of high residual error. By taking advantage of non-uniform atom clustering in a residual, the *NumberSplit* method spends an average of 0.25 bit per atom position less than the theoretical lower bound. Nevertheless, the *NumberSplit* algorithm does not explore temporal dependencies between residual frames, and encoding the number of atoms yields a relatively complex entropy coder which requires more computation to achieve efficiency. In contrast to this algorithm, we propose an algorithm based on a quadtree representation of a bit-plane. This algorithm predicts the quadtree for the adjacent bit-plane using the quadtree for the current bit-plane to remove spatial and temporal redundancy. Simulations show that the efficiency of our approach in encoding atom positions is improved to 1 to 2 bits below that of the theoretical low bound for uniform independent atom distribution.

We will explain how we implement quadtree, then give the details of simulation results. Quadtree is a simple image decomposition algorithm and has been used successfully in representing binary images at different resolution levels (Shusterman & Feder, 1994). A bit-plane can be represented by a quadtree. If the entire bit-plane has at least one atom, we label the root node "1". Four children, representing four quadrants of the bit-plane, are added to the root node. If the bit-plane has no atom, we label it "0". This process can be applied recursively to each of the four children until each child node represents a single pixel. Thus, if the image size is $2^{l_{max}} \times 2^{l_{max}}$, the quadtree has at most $l_{max} + 1$levels. Quadtree-based multi-resolution representation is resistant to small variations of bit patterns between bit-planes. An example is given in Figure 2 where the bit patterns at level 0 (the finest resolution of the quadtree) are different in (a) and (b). However, at upper levels of quadtrees of (a) and (b) (which correspond to coarser resolutions of the quadtrees) the same patterns occur. In other words, the small variations of 0 and 1 between the bit-planes do not propagate to upper levels. Hence, if two bit-planes have a lot of bit pattern overlap, the quadtree of one bit-plane can be used to efficiently predict the quadtree of the other bit-plane.

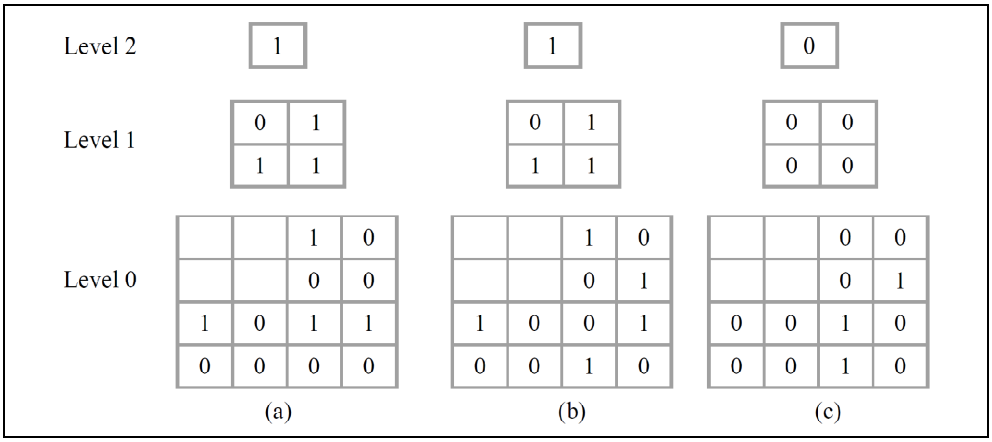


Fig. 2. Quadtree and quadtree prediction. (a) The quadtree to be predicted. (b) The quadtree from which to predict. (c) The EXCLUSIVE OR of (a) and (b). Note that one can obtain (a) by (b) EXCLUSIVE OR (c).

In video encoding at low bit rates, corresponding bit-planes in two consecutive frames and adjacent bit-planes within a frame tend to have many redundant structures. The temporal and spatial dependences of these bit-planes are exploited in the following recursive *Quadtree_Prediction* algorithm. The bit-plane b of the current frame is first represented as a quadtree Q_b^t. The quadtree is then predicted either from the quadtree of the corresponding bit-plane in the previous frame or from the quadtree of the union of all the previous bit-planes at the same frame. We use Q_b^t (k, i, j) to denote the node at the (i, j)-th position in level k of the quadtree corresponding to the b-th bit-plane in the t-th residual frame. For any P-frame in a GOP, starting from the most significant bit-plane (which is the N_{BP}-th bit-plane) toward the least significant bit-plane, our encoder enters this algorithm from the root node of Q_b^t. The nodes in the tree are then traversed from the root in the depth-first order. Note that other multi-dimensional tree search orderings can be used to traverse a quadtree (Le & Lei, 1999). We give our encoder prediction algorithm as follows. We use notation Q'^t_b to denote the differential quadtree which was obtained from predicting Q_b^t from that of its previous bit-planes. Note that our decoder uses the same algorithm described below, except that Q_b^t and Q'^t_b in the algorithm are switched.

$Quadtree\_Prediction(Q_b^t, k, i, j)$

{

IF Q_b^{t-1} is used to predict Q_b^t

(1)Output the bit $(Q_b'^t(k,i,j) = Q_b^t(k,i,j) \oplus Q_b^{t-1}(k,i,j))$;

OTHERWISE

(2)Output the bit $(Q_b'^t(k,i,j) = Q_b^t(k,i,j) \oplus \bigcup_{p=b+1}^{N_{BP}} Q_p^t(k,i,j))$;

IF $Q_b^t(k,i,j) = 0$

The node is a leaf node;

ELSE IF ($Q_b^t(k,i,j) = 1$ and $k = 0$)

(3) The node is a leaf node and associates with one or more atoms;

(4) The index of the basis and sign of the inner product of each atom and

a symbol indicating the last atom at the position are entropy encoded;

ELSE encode its four children

(5) $Quadtree\_Prediction(Q_b^t, k-1, 2i, 2j)$;

(6) $Quadtree\_Prediction(Q_b^t, k-1, 2i, 2j+1)$;

(7) $Quadtree\_Prediction(Q_b^t, k-1, 2i+1, 2j)$;

(8) $Quadtree\_Prediction(Q_b^t, k-1, 2i+1, 2j+1)$;

}.

(1) in the above algorithm uses temporal prediction of the current bit-plane. Differences in quadtrees are obtained from using EXCLUSIVE OR ($\oplus$) on corresponding nodes in Q_b^t and Q_b^{t-1}. (2) uses spatial prediction in which the quadtree corresponding to the bit-plane obtained from the union of the bit-planes from b + 1 to N_{BP} in the current frame is used to predict, again by EXCLUSIVE OR, the quadtree of the current bit-plane. In (3) and (4), a "1" at a terminal node indicates new significant atoms whose basis indexes and signs of their inner products are then entropy-encoded. Since more than one atom can become significant at a location of a bit-plane, we introduce one symbol *last*, with a value of either 1 or 0, which indicates whether an atom is the last atom at a given position of a bit-plane. In our implementation, each atom is associated with a triplet (*index, sign, last*). If two atoms become significant at the same location of a bit-plane, then the *last* in the first atom is 0 and the *last* of the second atom is 1. Two atoms may become significant at the same location but in different bit-planes. In this case, the *last* of both atoms is 1.

If a node is not a leaf, then its four children are visited in a depth-first search order. Figure 2 is a simple example illustrating (a) the quadtree for the current bit-plane (to be predicted) and (b) the previous bit-plane (from which to predict). The differential quadtree (c) is the result of applying EXCLUSIVE OR on the corresponding nodes in (a) and (b). The blanks in the top left corner of all subfigures indicate that nodes located there are not encoded since their parent nodes have value 0. One can recover (a) from using EXCLU-SIVE OR on (b) and (c). The differential quadtree in (c) is traversed in depth-first order and yields the sequence 00000000000101100. Since there are many zeroes, the entropy of the differential quadtree is relatively low. Moreover, the symbols used in the sequence are only zeroes and ones, thus they can be encoded efficiently via the adaptive arithmetic code. When applying our algorithm to encode atoms, the index of the basis function, the sign of inner product value, and the last symbol of each atom are each entropy-encoded by adaptive arithmetic codes.

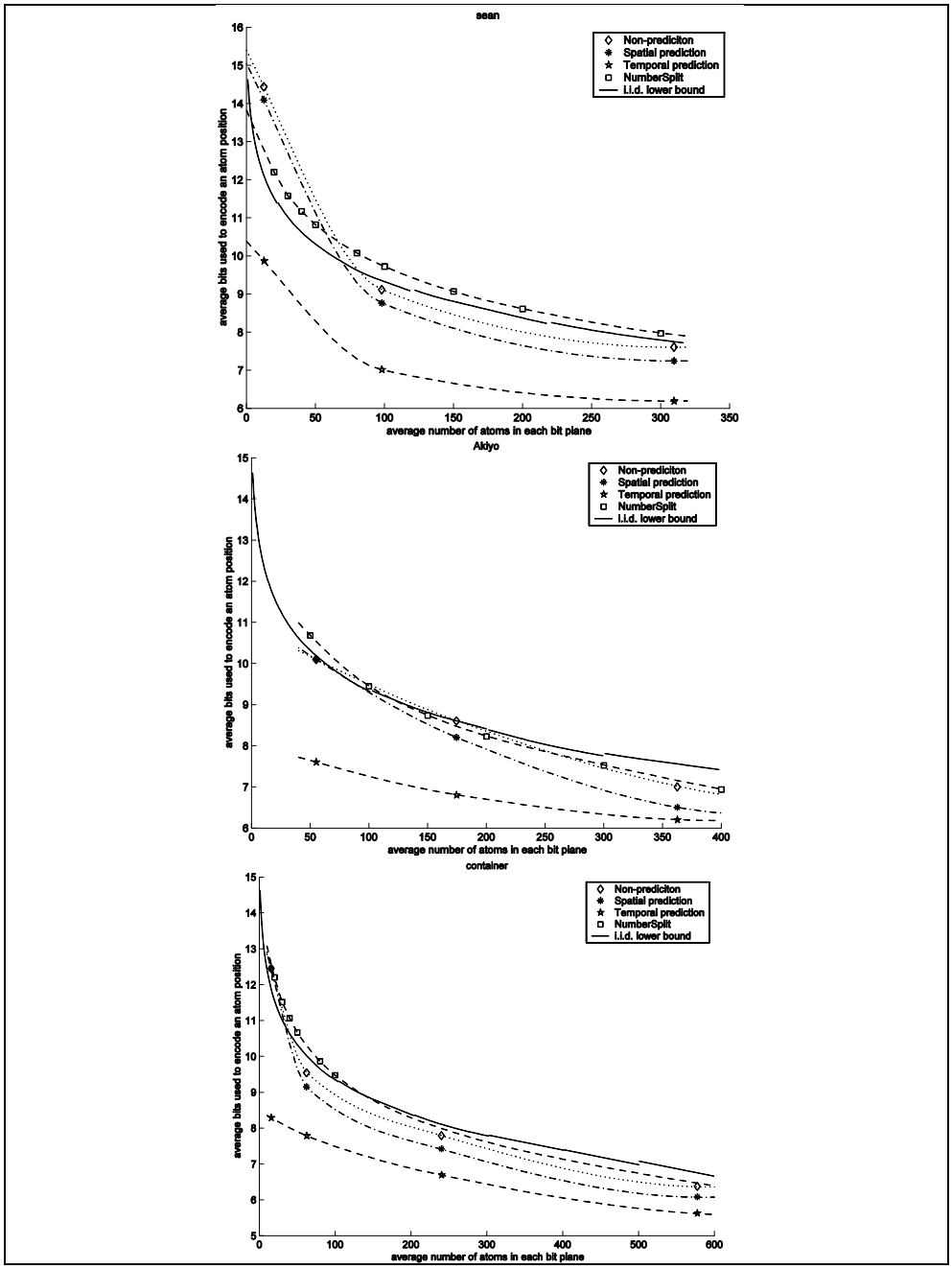


Fig. 3. Comparison of atom position encoding efficiency of quadtree, quadtree prediction and *NumberSplit*. Sequences from top to bottom are respectively Sean, Akiyo and Container.

The performance of the proposed algorithm depends on the correlation of atoms between bit-planes. A node at a higher level of a quadtree represents a larger block in the bit-plane. It is likely that corresponding higher-level nodes have the same value. This is particularly true when coding slow motion sequences in which larger motion residual errors most likely occur at the boundaries of moving objects. These errors tend to be located in almost the same region of two adjacent frames. As for coding efficiency of our algorithm, in Figure 3 we demonstrate and compare the average bits used to encode each atom position over Sean, Akiyo and Container sequences using 10 frames/sec in a QCIF format and a ten second testing time. The first three bit-planes for each frame in Sean and Akiyo are encoded while the first four bit-planes are encoded in each frame of the Container sequence. These bit-planes are encoded using either temporal bit-plane prediction, spatial bit-plane prediction, or no prediction. The X-axis in the figure is the average number of atoms for each bit-plane and the Y-axis is the average number of bits that encodes each atom position for each bit-plane. From left to right, the first mark in a curve of either quadtree or quadtree prediction corresponds to the first significant bit-plane, the second mark to the second significant bit-plane, and so on. Squares (X,Y) in the *NumberSplit* curve indicate that an average of Y bits is used to encode one atom position. All atoms are partitioned into groups of X atoms. The clustering parameter f in *NumberSplit* is set to 0.5.
The coding efficiency of temporal bit-plane prediction is evidently superior to all the others including that derived from theoretical lower bound. This bound, obtained by assuming that atoms are uniformly and identically distributed, can be out-performed if an algorithm can effectively take advantage of non-uniformity in atom distributions. The performances of quadtree without prediction and the *NumberSplit* algorithm have similar results. Quadtree results using spatial bit-plane prediction are slightly better than those without.

4.3 Progressive atom coding algorithm

This section provides the final framework that combines the set partitioning scheme for atom modula and atom positions encoding. We alternatively apply the following two phases on each bit-plane in a residual frame after initially setting the required parameter which would be the most significant bit-plane in a video. In the refinement phase, atom modula in the *AtomList* are refined by adding one extra bit of information. In the sorting phase, new significant atoms are found in the current bit-plane and are then appended to the *AtomList*. The following gives the algorithmic description of the framework.

1. **Initialization :** Set the $AtomList$ as empty. Let n be the most significant bit-plane for motion residual frames of our video.

2. **Refinement Phase :** produce a bit stream corresponding to one extra bit from the modulus of each atom in the $AtomList$.

3. **Sorting Phase :** use the $Quadtree\_Prediction$ algorithm to generate a set of new

atoms whose absolute values of inner products are within $[2^n, 2^{n+1})$ and then include the new atoms to the end of the $AtomList$.

3.1 If the bit-plane-shift parameter $b > 0$, then encode the values in bit-planes $n - 1, \cdots, n - b$ of the new significant atoms.

4. **Next Bit-Plane :** decrease n by 1 and go to Step 2.

Note that the above algorithm is not the same as that proposed in (Shapiro, 1993; Said & Pearlman, 1996). Compared to theirs, our algorithm allows more than one bit-plane atom modulus to be encoded at the sorting pass for each new significant atom. Parameter b in step 3.1, which gives the number of extra bit-planes from the current bit-plane, is used in encoding the modulus of a new significant atom. The position of each atom is encoded only in the pass in which the atom becomes significant and then the atom's modulus is refined by successive passes through the refinement phase. Encoding the position of a new significant atom requires more bits than in refining the atom's modulus. The purpose of encoding more than one bit-plane for a new significant atom is to increase the distortion-reduction δD of atom modulus to compensate for the relative large rate R in encoding the new significant atom.
Let the largest atom modulus be normalized to 1 and let m atoms be newly significant at the k-th significant bit-plane. The total bits spent is at most R_m+mb, where R_m is the bits for the atoms and mb is the bits for coding extra modula of the atoms. In matching pursuit video coding, $\frac{R_m}{m} \gg b$ is satisfied for new significant atoms in a bit-plane and reasonable b. Using b extra bit-planes when coding a new significant atom reduces a fraction of at most 2^b distortion over that without using an extra bit-plane (with approximately the same number of bits). This yields an increase of PSNR for encoding new significant atoms. An atom newly significant at the k-th significant bit-plane has a normalized modulus error of at most 2^{-k} and, with bit-plane-shift parameter b, the distortion becomes $2^{-(k+b)}$. Encoding the modulus of a new significant atom with more than one bit-plane is a feature in our low bit rate video coding. The efficacy of this feature in FGS at low bit rates is illustrated in the next section. Figure 4 gives the order in which bit-planes are included in the sorting and refinement phases of our algorithm. The process of sorting and refining bit-planes of the left subfigure is b =0 which is the same as that in (Shapiro, 1993; Said & Pearlman, 1996). In the right subfigure, three bit-planes of atom modula are encoded at the sorting phase, while one bit-plane is encoded at the enhancement phase.

5. FGS matching pursuit video codec

A video streaming encoder compresses and stores video streams and simultaneously transmits them on demand through a scalability-aware transport mechanism to a large amount of users (Chou & Miao, 2006). There are many video scalable encoder applications. Among them, the FGS MPEG-4 video approach has attracted the most attention as it has achieved a fine balance between coding efficiency and coding complexity for producing

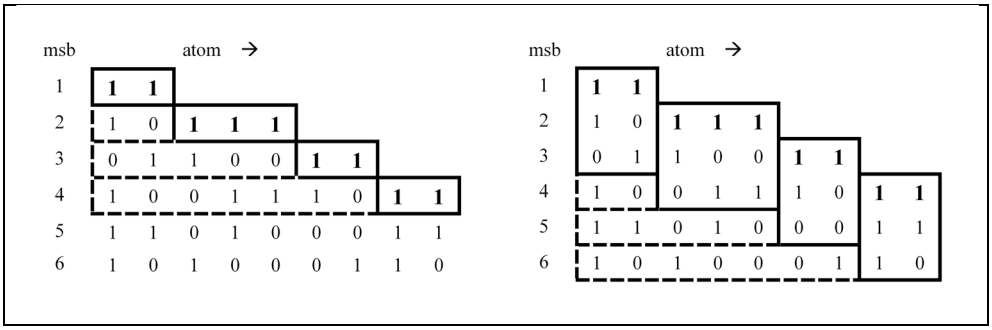
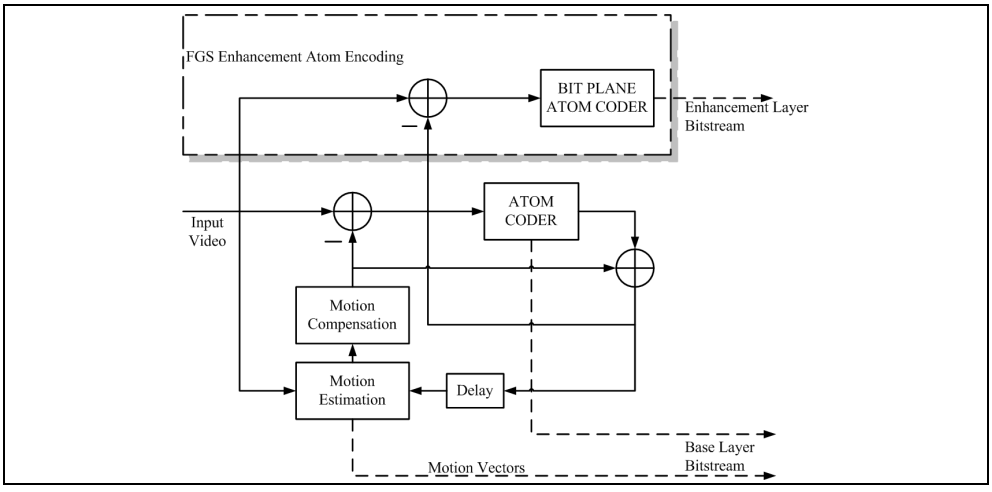


Fig. 4. Modulus sorting and refinement comparison using different b after 4 passes. Left: b = 0. Right: b = 2. Bit-plane values included in sorting and in refinement phases are indicated by solid boxes and dashed boxes respectively.

scalable bit streams. A basic FGS framework requires two layers: the base layer and the enhancement layer. The base layer includes a motion prediction and has an encoder with highly efficient coding in low bit rates. Any decoder must be able to decode the base-layer bit stream. In principle, any FGS method can be used to produce streaming bit-streams for the enhancement layer. The bit-plane-based DCT FGS encoding method is still the most widely used.

5.1 Proposed FGS

Our proposed two-layer FGS matching pursuit video codec is shown in Figure 5. Our base layer encoder, shown at the top subfigure, performs motion compensation and encodes motion residual frames using a matching pursuit. Although an MP-based video encoder is more complex, its decoder is comparably less complex than other methods. Both the base layer and the enhancement layer of our matching pursuit FGS coders use the progressive atom encoding algorithm proposed in previous section in which the atom position is encoded by the Quadtree Prediction algorithm. The position encoding algorithm must store both a quadtree



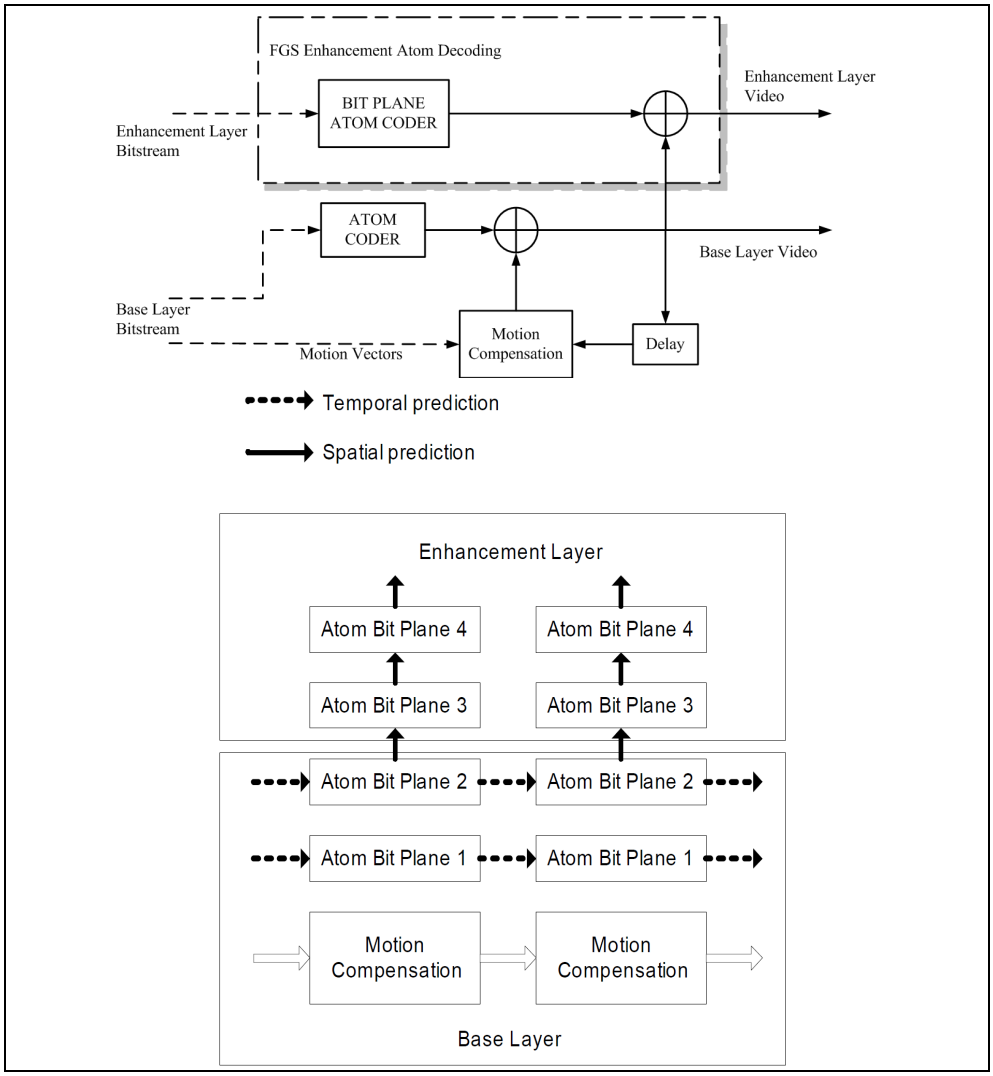


Fig. 5. Our FGS matching pursuit codec. Top: Encoder. Middle: Decoder. Bottom: Atom positions encoded by bit-plane-based predictions.

for each base layer bit-plane in the previous frame (in order to perform temporal prediction), and a quadtree for the union of previous bit-planes at the current frame (to perform spatial prediction). Representing the quadtree of an N pixels bit-plane takes at most $\frac{4N}{3}$ bits. If we operate at a bit rate with two bit-planes as base layer, we would need at most 4N bits for storage. Although slightly more storage is required, the entropy-coder is a two symbol adaptive arithmetic code which is easily implemented at a decoder site. The bottom subfigure of Figure 5 illustrates an example of our bit-plane-based FGS to encode atom position. Temporal prediction is carried out only in base layer bit-planes while spatial prediction is

carried out in enhancement layer bit-planes. A bit-plane in the enhancement layer is predicted from all the previous bit-planes from the same frame. The spatial or temporal bit-plane predictions are based on operations on quadtrees. The PSNR can be lifted by finely quantizing atom modula of new significant atoms by setting parameter b. Figure 6 shows experimental results of PSNR performance at low bit rates with different b.

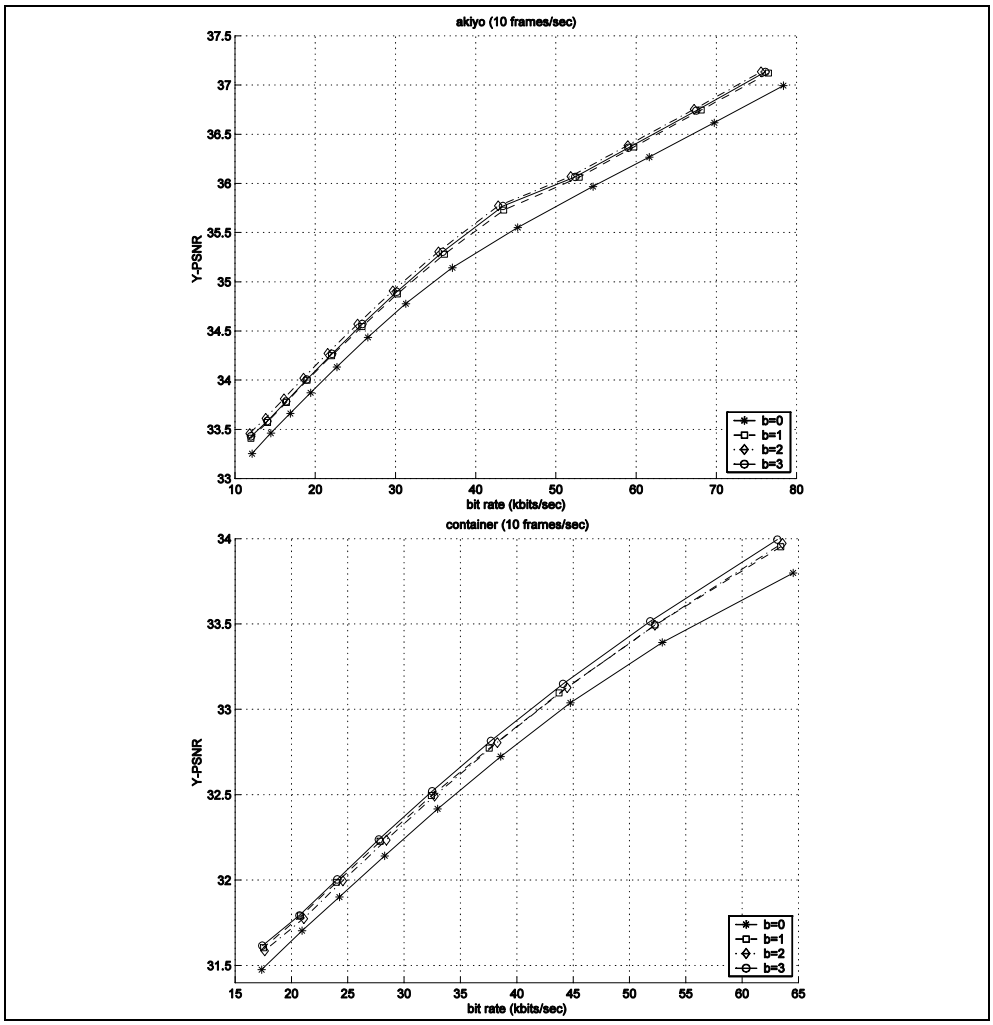


Fig. 6. PSNR at low bit rates by setting different b. Top: Akiyo. Bottom: Container. There is an average 0.2 dB improvement if b is set to either 2 or 3.

5.2 Performance evaluations and comparisons

The performance of our matching pursuit FGS at various bit rates in terms of luminance PSNR (Y-PSNR) are compared with those of DCT FGS. Our FGS DCT-based codec follows

Amendment 2 of the MPEG-4 FGS DCT-based video codec in encoding the enhancement layer of a residual frame (ISO/IEC 14496-2, 2001). In all the following experiments, we divide a frame into blocks and select an atom from only one block at each iteration (Neff & Zakhor, 1997). We use the weighted energy block search algorithm, with weights provided in (Al-Shaykh et al., 1999), to find atoms. The first two parts of this section compare performances of our bit-plane-based matching pursuit FGS to bit-plane-based DCT FGS at different low bit rates. The third part provides the evaluation of our FGS to a bit-plane-based non-scalable matching pursuit video coding.

5.2.1 Comparison of MP and DCT FGS using the same residual image

For a fair comparison of coding efficiency of residual images between bit-plane-based MP FGS and DCT FGS, different codecs should encode the same residual images using the same number of bits. We set up the experiments so that the base layer of the current frame is obtained from the base layer of the previous frame using only motion compensation. In other words, a frame is estimated from the previous base layer using motion compensation and the residual is the enhancement layer of the frame. Accordingly, the differences in encoding residual images by the codecs will not be involved in predicting the following frames, and the codecs will encode the same residual images.

Figure 7 shows Y-PSNR comparisons of our matching pursuit FGS codec with that of the DCT FGS codec using various sequences in QCIF at 10 frames/sec. Because both codecs encode the same residuals in the enhancement layer, as indicated in the Figure 7, the performance of bit-plane-based matching pursuit FGS is better than that of DCT FGS in encoding residuals at low bit rates. The slope corresponding to the matching pursuit bit-plane encoder is higher than that of the DCT bit-plane at bit rates close to the base layer bit rate in each sequence, and it yields that the curve of matching pursuit is above that of DCT in each sequence. This may be because matching pursuit first picked up a few locally energy-concentrated patterns, causing a high reduction in distortions, so that the matching pursuit slope is initially higher than that of the DCT slope. The slopes of the two curves become approximately the same as the bit rate increases. For the Akiyo sequence, at 100 kbps, the largest PSNR gain of the matching pursuit bit-plane encoder over the DCT bit-plane encoder is about 0.8 dB.

5.2.2 Comparisons at the base layer and the enhancement layer

We evaluated the coding efficiency of both layers in matching pursuit FGS and DCT FGS by comparing their Y-PSNR at various low bit rates. Both codecs have the same intra-frame (I-frame) encoded by the DCT method. The other frames are all inter-frame (P-frame). For all comparisons, the base layer of the FGS MP-based codec includes either one or two most significant bit-planes. The base layer bit rates of the FGS DCT-based codec is determined by using a fixed quantization step (QP) for all frames in the sequence. This assures that the base layer's bit rates of the two codecs are as close as possible. The frame rate of all sequences is 10 frames/sec and the format for all sequences is QCIF.

The average Y-PSNR versus the bit rates is plotted in Figure 8. Base layer performance corresponds to the beginning points whose X-axes are at the lowest bit rates in the curves. In all experiments, the Y-PSNR of our MP-based codec is better than that of the DCT-based codec, both in the base layer and enhancement layer. The average base layer improvement is about 0.7 dB. Also, the Y-PSNR of the matching pursuit FGS increases faster than that of the DCT FGS as bit rates increase.

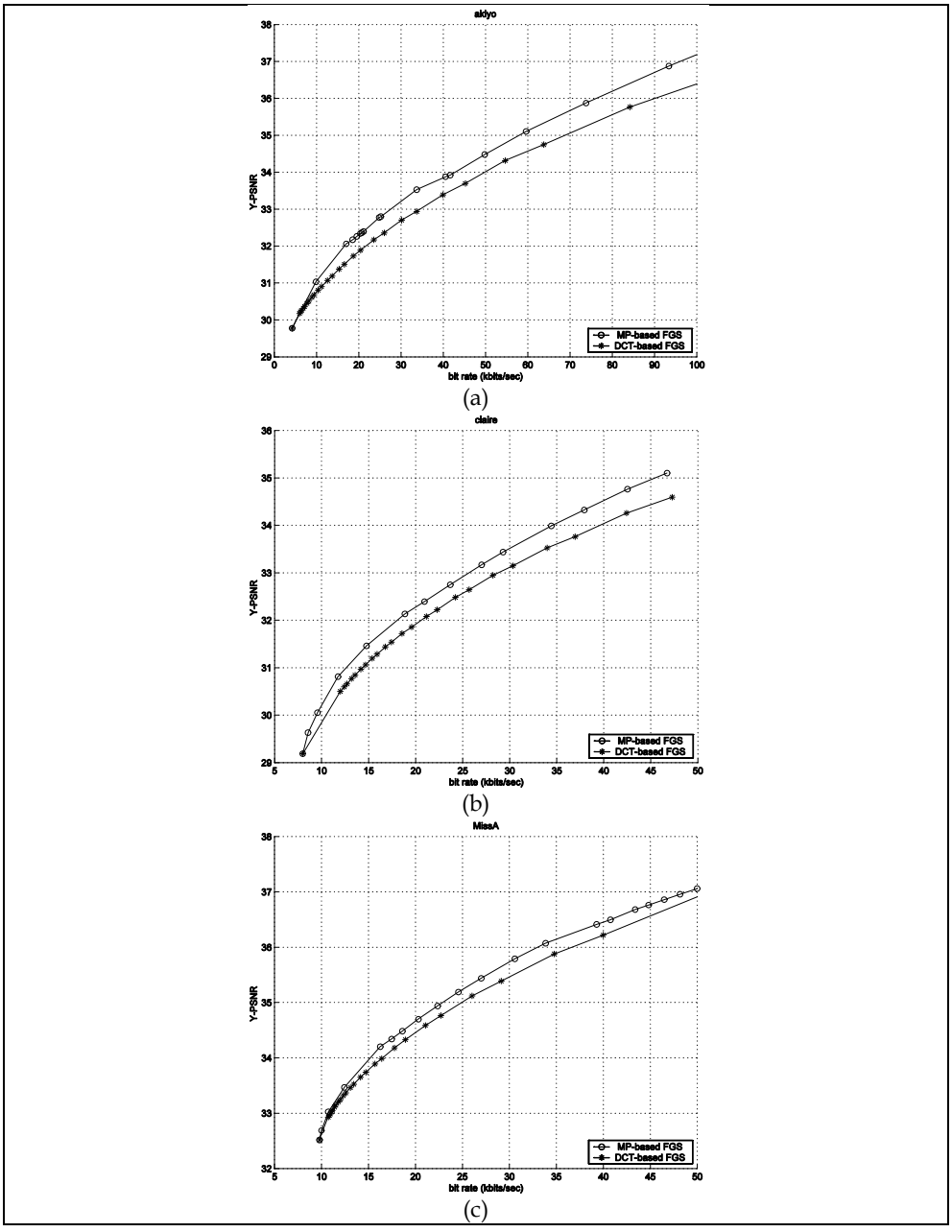


Fig. 7. Comparison of the same residual images encoded by bit-plane-based matching pursuit and bit-plane-based DCT method for (a) 10 seconds Akiyo, (b) 5 seconds Claire, and (c) 3 seconds Miss America sequences. The frame rate is 10 frame/sec and bit-plane-shift parameter b =3.

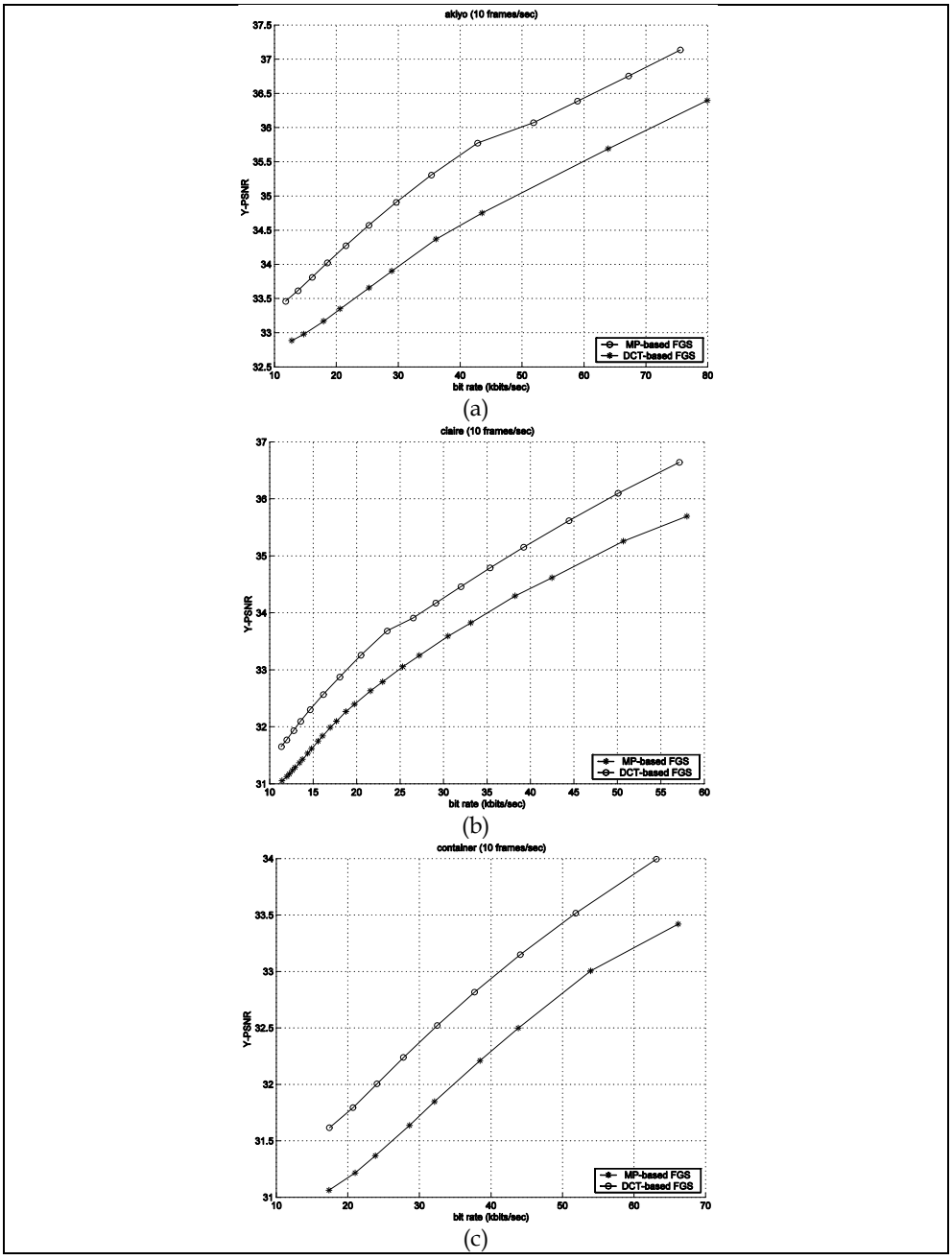


Fig. 8. Comparison of MP-based FGS with DCT-based FGS for (a) Akiyo, (b) Claire, and (c) Container. All sequences are 10 frames/sec, in QCIF, for 3.3 second. The bit-plane shift parameter b = 3.

5.2.3 Evaluating bit-plane representation and prediction of FGS

Here we use different bit-plane numbers as our base layer and compare performances with a bit-plane-based non-scalable codec at low bit rates. Note that our bit-plane-based non-scalable codec does not optimize at a particular bit rate and it is not the best non-scalable matching pursuit codec. Nevertheless, it provides a reference we can use to evaluate important aspects of our FGS.

Figure 9 gives the PSNR performance versus bit rates of the Akiyo and Container sequences. The curve corresponding to the top envelope of each subfigure is the performance of our non-scalable codec in which all the bit-planes are temporally predicted. The previous reconstructed frame from all the bits is used to perform motion compensation for the current frame. The rest of the curves, from the bottom to the top, correspond to performance using one bit-plane, two bit-planes and three bit-planes as our base layer.

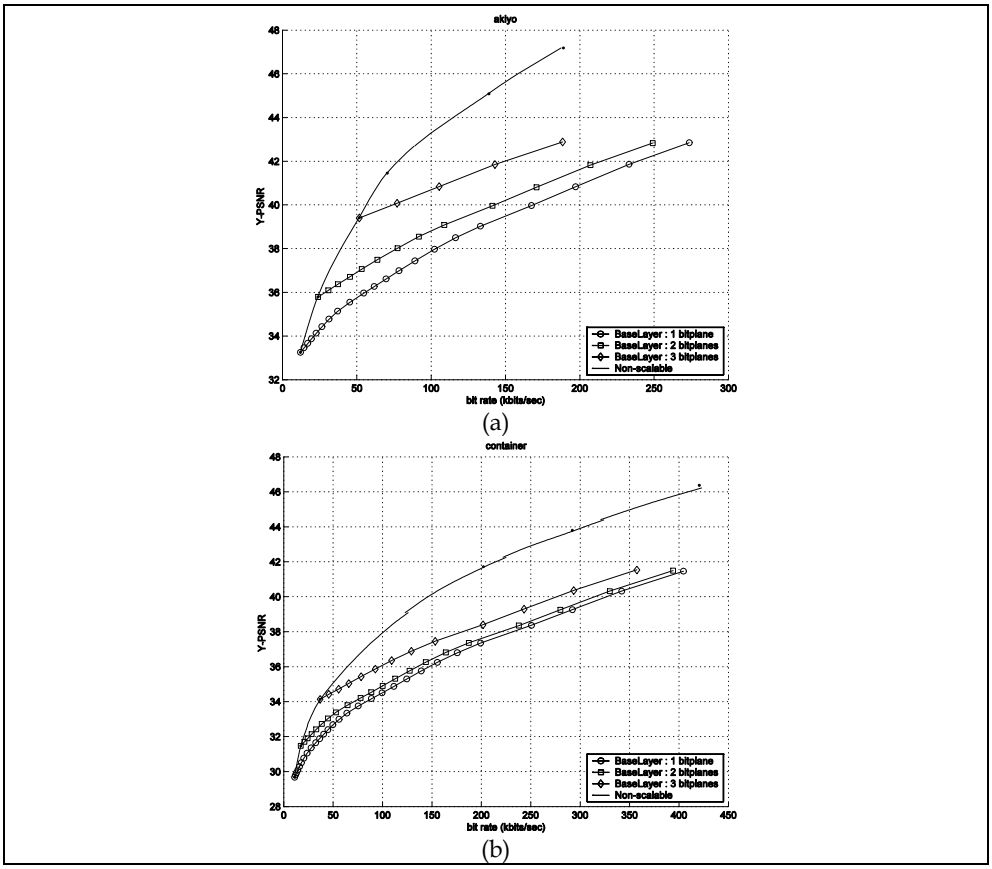


Fig. 9. Comparison of our FGS with various bit-plane numbers as base layers to a bit-planed-based non-scalable codec. Sequences: (a) Akiyo and (b) Container, are 10 frames/sec, in QCIF, for 3.3 second.

The efficacy of using bit-plane prediction and motion compensation manifests at relatively lower bit rates when the PSNR gap between consecutive curves is comparably large. The

curve using one bit-plane as base layer shows the greatest PSNR increase when bit rates are close to the base layer. This is because matching pursuit selected a few atoms with energy concentrated in the first two significant bit-planes and, therefore, caused a higher slope increase at lower bit rates. The gap between curves continues to decrease until higher bit rates are reached at which point the curves are almost parallel.

6. Conclusions

Because channel capacity varies according to network traffic and the capacity of each receiver, fine granular scalability of video coding has emerged as an important area in multimedia streaming research. We propose an FGS video codec based on matching pursuits and bit-plane coding. The proposed FGS algorithm uses the spatial and temporal dependence between bit-planes to exploit the redundancy in the bit-planes. The efficiency of our algorithm in encoding atom positions lies in using a quadtree to represent a bit-plane and to perform bit-plane prediction. The experiment results demonstrate that this algorithm can reduce the number of bits required to encode the atom positions by 1 to 2 bits. The PSNR of the proposed algorithm is compared to and out-performs that of the DCT-based FGS algorithm.

7. References

O. Al-Shaykh, E. Miloslavsky, T. Nomura, R. Neff, and A. Zakhor. (Feb. 1999) Video compression using matching pursuits. IEEE Trans. on Circuits and Systems for Video Technology, 9(1):123–143.

P. A. Chou and Z. Miao. (2006) Rate-distortion optimized streaming of packetized media. IEEE trans. on Multimedia, Vol 8, No.2, pages 390-404.

P. Czerepi´nski, C. Davies, N. Canagarajah, and D. Bull. Matching pursuits video coding: dictionaries and fast implentation. IEEE Trans. on Circuits and Systems for Video Technology, 10(7):1103–1115, Oct. 2000.

G. Davis. (1994) Adaptive Nonlinear Approximations. PhD thesis, Department of Mathematics, New York University.

K. Engan, S.O. Aase, and J.H. Hus. (Oct. 2000) Multi-frame compression: Theory and design. EURASIP Signal Processing, 80(10):2121–2140.

B. Girod, M. Kalman, Y. J. Liang, and R. Zhang. (Sept. 2002) Advances in channel-adaptive video streaming. Journal of Wirelesss Communications on Mobile Computing, 2(6):573– 584.

ISO/IEC 14496-2:2001. Streaming Video Profile.

ISO/IEC JTC1/SC29/WG11, Doc. M10569/S03.(March 2004) Scalable extension of H.264/AVC,

B. Jeon and S. Oh. (April 2003)Fast matching pursuit with vector norm comparison. IEEE Trans. on Circuits and Systems for Video Technology, 13(4):338–342.

J. Le and S. Lei. (July 1999) An embedded still image coder with rate-distortion optimization. 8(7):913–924.

W. Li. (March 2001) Overview of fine granularity scalability in MPEG-4 video standard. IEEE Trans. on Circuits and Systems for Video Technology, 11(3):301–317.

J.L. Lin, W.L. Hwang, and S.C. Pei. (Jan. 2005) SNR scalability based on bitplane coding of matching pursuit atoms at low bit rates: fine-grained and two-layer. IEEE Trans. on Circuits and Systems for Video Technology, 15(1):3–14.

J.L. Lin, W.L. Hwang, and S.C. Pei. (May 2006) Video compression based on orthonormal matching pursuits. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS).

J.L. Lin, W.L. Hwang, and S.C. Pei. (March 2006) Multiple blocks matching pursuit update for low bit rate video coding. IEEE Trans. on Circuits and Systems for Video Technology, Vol.16, No. 3, pp.331-337.

J.L. Lin, W.L. Hwang, and S.C. Pei. (2007) A combined dictionary approximation and maximum atom extraction design for matching pursuit speed-up. IEEE Trans. on Circuits and Systems for Video Technology, Vol.17, No. 12, pp.1679-1689.

G. Mallat and Z. Zhang. (Dec. 1993) Matching pursuits with time-frequency dictionaries. IEEE trans. on Signal Processing, 41:3397–3415.

S. McCanne, M. Vetterli, and V. Jacobson. (Aug. 1997) Low-complexity video coding for receiver-driven layered multicast. IEEE Journal of Selected Areas in Communications, 15(6):982–1001.

R. Neff, T. Nomura, and A. Zakhor. (1998) Decoder complexity and performance comparison of matching pursuit and DCT-based MEPG-4 video codecs. In Proceedings of the IEEE International Conference on Image Processing (ICIP), pages 783–787.

R. Neff and A. Zakhor. (Feb. 1997) Very low bit-rate video coding based on matching pursuits. IEEE Trans. on Circuits and Systems for Video Technology, 7:158–171.

R. Neff and A. Zakhor. (Sept. 2000) Modulus quantization for matching pursuit video coding. IEEE Trans. on Circuits and Systems for Video Technology, 10(6):895–912.

R. Neff and A. Zakhor. (Jan. 2002) Matching pursuit video coding–part I: dictioanry approximation. IEEE Trans. on Circuits and Systems for Video Technology, 12(1):13–26.

J.-R. Ohm, M. van der Schaar, and J. W. Woods. (Oct. 2004) Interframe wavelet coding-motion picture representation for universal scalability. Signal Processing: Image Communication, 19(9):877–908.

D.W. Redmill, D.R. Bull, and P. Czerepi´nski. Video coding using a fast non-separable matching pursuits algorithm. In Proceedings of the IEEE International Conference on Image Processing (ICIP), pages 769–773, 1998.

K. Rose and S. L. Regunathan. (July 2001) Toward optimality in scalable predictive coding. 10(7):965–976.

A. Said and W. A. Pearlman. (June 1996) A new fast and efficient image codec based on set partitioning in hierarchical trees. IEEE Trans. on Circuits and Systems for Video Technology, 6(3):243–250.

J. M. Shapiro. (Dec. 1993)Embedded image coding using zerotrees of wavelet coefficients. IEEE trans. on Signal Processing, 41(12):3445–3462.

E. Shusterman and M. Feder. (March 1994) Image compression via improved quadtree decomposi-tion algorithms. 3(2):207–215.

X. Tang and A. Zakhor. (June 2002) Matching pursuits multiple description coding for wireless video. IEEE Trans. on Circuits and Systems for Video Technology, 12(6):566–575.

M. Vetterli and T. Kalker. (1994) Matching pursuit for compression and application to motion compensated video coding. In Proceedings of the IEEE International Conference on Image Processing (ICIP), pages 725–729.

C. De Vleeschouwer and B. Macq. (Oct. 1999) Subband dictionaries for low-cost matching pursuits of video residues. IEEE Trans. on Circuits and Systems for Video Technology, 9(7):984–993.

C. De Vleeschouwer and B. Macq. (Dec. 2000) SNR scalability based on matching pursuits. IEEE trans. on Multimedia, 2(4):198–208.

C. De Vleeschouwer and A. Zakhor. (June 2002) Atom modulus quantization for matching pursuit video coding. In Proceedings of the IEEE International Conference on Image Processing (ICIP), volume 3, pages 681–684.