Chapter 2

The Matching Pursuit Video Encoder

In this chapter, we describe the basic matching pursuit video encoding system upon which the remaining chapters will be built. The system is similar to the one described in [45], but various changes and improvements have been added since then. Consequently, the system description and results have been updated as needed.

2.1 Introduction

The hybrid motion-compensated DCT structure is a part of nearly all existing video coding standards [26], [1], [34]. 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 both hardware and optimized software which efficiently implement DCT-based coders. Unfortunately, block-based DCT systems have trouble coding sequences at very low bit rates. At 24 kbit/s and below, 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 has better coding efficiency at low rates. Rather than 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, it is 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 [41], and is related to earlier works in statistics [21] and multistage vector quantization [68]. The matching pursuit technique is general in that it places virtually no restrictions on the choice of basis dictionary. We will use an overcomplete set of separable Gabor functions which do not contain artificial block edges. Our design will be spatially invariant, 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.

We begin in Section 2.2 by reviewing the theory and some of the mathematics behind the matching pursuit technique. Section 2.3 provides a detailed description of the resulting video codec. This includes subsections relating to motion compensation, matching pursuit encoding of the motion residual, rate control, and the method used to encode intra frames. Coding results are presented in Section 2.4, and conclusions can be found in Section 2.5.

2.2 Matching Pursuit Theory

The Matching Pursuit algorithm, as proposed by Mallat and Zhang [41], expands a signal using an overcomplete dictionary of functions. Before discussing the adaptations made for video compression, we begin with a generic description of the algorithm. Suppose we want to represent a signal vector $\vec{f}$ using basis functions from a dictionary set $\mathcal{G}$. Individual dictionary functions have unit norm and are denoted as:

$$\vec{g}_\gamma \in \mathcal{G} \tag{2.1}$$

Here γ is an indexing parameter associated with a particular element of the dictionary. The decomposition is done over multiple stages. At each stage i, we choose γ in order to maximize the absolute value of the following inner product:

$$p_i = \left\langle \vec{f}_{i-1}, \vec{g}_\gamma \right\rangle \tag{2.2}$$

Here $\vec{f}_i$ is the signal energy remaining after stage i, with $\vec{f}_0 = \vec{f}$. We then say p_i is an expansion coefficient for the signal onto the chosen dictionary function, which we label g_{γ_i}. A residual signal is computed as:

$$\vec{f}_i = \vec{f}_{i-1} - p_i\, \vec{g}_{\gamma_i} \tag{2.3}$$

The current residual $\vec{f}_i$ is then expanded in the same way as the previous residual. The procedure continues iteratively until some stopping criteria is reached. For example, a set number of expansion coefficients may be generated, or some energy threshold for the residual may be reached. Each stage i yields a dictionary structure specified by γ_i, an expansion coefficient p_i, and a residual $\vec{f_i}$ which is passed on to the next stage. After a total of N stages, the signal may be approximated as a weighted summation of the dictionary elements:

$$\vec{f} \approx \vec{f'} = \sum_{i=1}^{N} p_i \, \vec{g}_{\gamma_i} \tag{2.4}$$

Note that the above equations are suitable for signal decomposition and analysis, which were the original applications explored in [41]. Some modifications are necessary in order to use the technique for encoding. Perhaps the most obvious is that some quantizer $Q(\cdot)$ must be applied within the atom finding loop to quantize each modulus value p_i. The quantized values are then used to compute the reconstruction of Equation (2.4). Modulus quantization will be explored in greater detail in Chapter 4.

The matching pursuit 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 $\vec{f}$ and the coded signal $\vec{f'}$. 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 most efficiently coded by designing the dictionary functions to match the shape, scale, or frequency of the desired features.

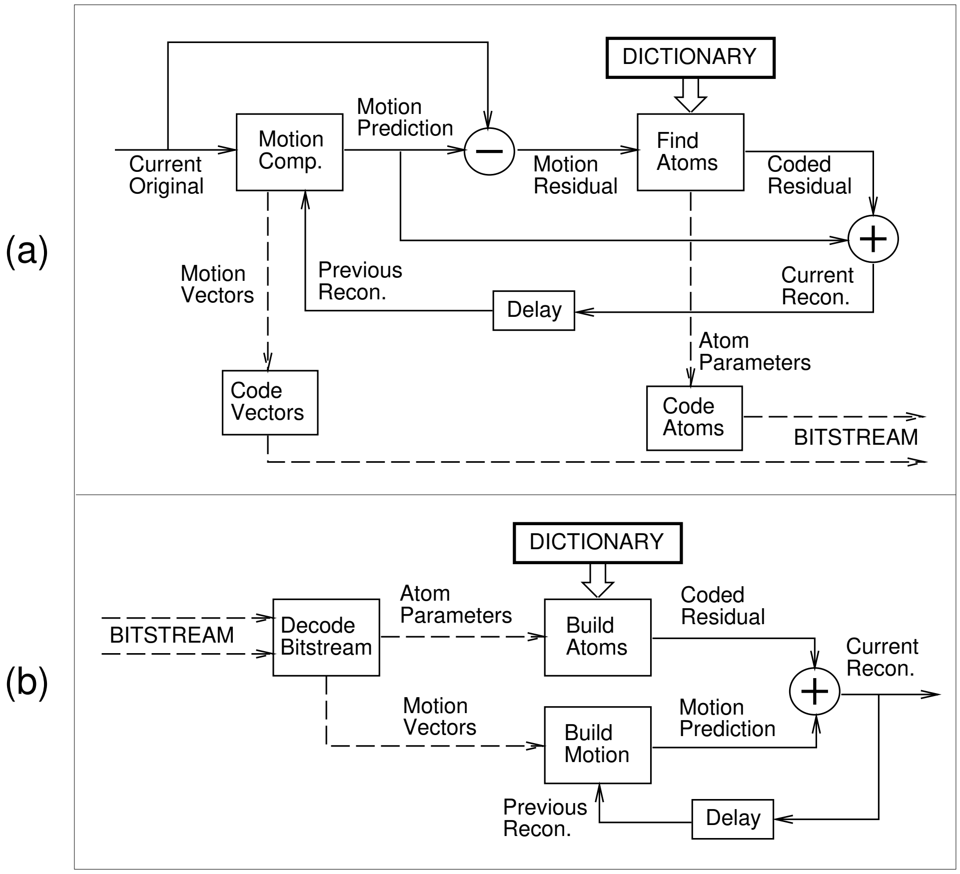
An interesting property of the matching pursuit technique is that it places very few restrictions on the dictionary set. The original Mallat and Zhang paper considers both Gabor and wavelet packet dictionaries, but such regular structure is not required by the algorithm itself. Mallat and Zhang showed that if the dictionary set is at least complete, then $\vec{f}'$ will eventually converge to $\vec{f}$, although the rate of convergence is not guaranteed [41]. Convergence speed and thus coding efficiency are strongly related to the design of the dictionary. However, true dictionary optimization can be difficult since there are so few restrictions. Any collection of arbitrarily sized and shaped functions may be used with matching pursuit, as long as completeness is satisfied.

We now show how the matching pursuit algorithm may be adapted to video compression.

2.3 System Description

We apply the matching pursuit decomposition technique to encode motion residual signals in a hybrid video codec. Simplified block diagrams of the encoder and decoder are shown in Figure 2.1. As can be seen in Figure 2.1a, original images are first motion compensated using the previous reconstructed image. The motion model is taken from the MPEG-4 standard, as described in Section 2.3.1. The matching pursuit algorithm is then used to decompose the motion residual signal into coded dictionary functions which are called atoms. The process of finding and coding atoms is described in Section 2.3.2. Once the motion vectors and atom parameters are found, they can be efficiently coded and sent to the decoder, shown in Figure 2.1b. The decoder then reconstructs both the motion prediction image and the matching-pursuit coded residual, which are then combined to form

Figure 2.1: (a) Block diagram of encoder. (b) Decoder.



the reconstructed image.

2.3.1 Motion Compensation

The motion model which will be used throughout the thesis for our matching pursuit system is similar to that used in the MPEG-4 standard [35]. The main difference is that intra-block coding, applied when motion compensation fails, is implemented differently for matching pursuit. This is done mainly because matching pursuit is not a block based method. We now describe the motion model in detail.

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. A decision criterion 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. A block splitting criterion is then 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 an overlapping motion window.

DCT-based systems such as H.263 and MPEG-4 video normally encode intra blocks using DCT without motion prediction. In order to avoid DCT artifacts in the matching pursuit encoder, 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 overlapping motion block window is used to smooth the edges of intra blocks and thus reduce blocking effects in the prediction image. Additional detail coding is left to the matching pursuit residual encoder. Specifically, if the intra prediction for a particular block has significant residual energy, then atoms will be attracted to the intra block during the residual coding stage. More on this stage is described in the next subsection.

2.3.2 Matching-Pursuit Residual Coding

After the motion prediction image is formed, it is subtracted from the original image to produce the motion residual signal, $\vec{f}$. This residual image is coded using matching pursuits. We now describe the details of this procedure.

The Dictionary Set

We use the 2-D separable Gabor dictionary defined in [45]. The set is defined in terms of a 1-D prototype Gaussian window:

$$g(t) = \sqrt[4]{2} e^{-\pi t^2} \tag{2.5}$$

By scaling and modulating this window, we build a set of 1-D discrete Gabor functions. These are defined as:

$$g_{\vec{\alpha}}(n) = K_{\vec{\alpha}} \cdot g(\frac{n - \frac{L}{2} + 1}{s}) \cdot cos(\frac{2\pi\xi(n - \frac{L}{2} + 1)}{16} + \phi) \qquad n \in \{0, 1, ...L{-}1\}; \tag{2.6}$$

Each 1-D Gabor function is parameterized by $\vec{\alpha} = (s, \xi, \phi)$. This triple consists respectively of a positive scale, a modulation frequency, and a phase shift. The constant $K_{\vec{\alpha}}$ is chosen such that the resulting 1-D basis vector is of unit norm. Suppose $\mathcal{B}$ is some set of triples $\vec{\alpha}$. We then define our 2-D separable Gabor dictionary to consist of all separable products of the 1-D functions specified by $\mathcal{B}$. Specifically, our 2-D dictionary functions are defined as:

$$G_{\vec{\alpha},\vec{\beta}}(m, n) = g_{\vec{\alpha}}(m) g_{\vec{\beta}}(n) \qquad \begin{array}{l} m \in \{0, 1, ...L_h{-}1\} \\ n \in \{0, 1, ...L_v{-}1\} \\ \vec{\alpha}, \vec{\beta} \in \mathcal{B} \end{array} \tag{2.7}$$

where L_h and L_v are respectively the horizontal and vertical extents of the 2-D basis function $G_{\vec{\alpha},\vec{\beta}}(m,n)$.

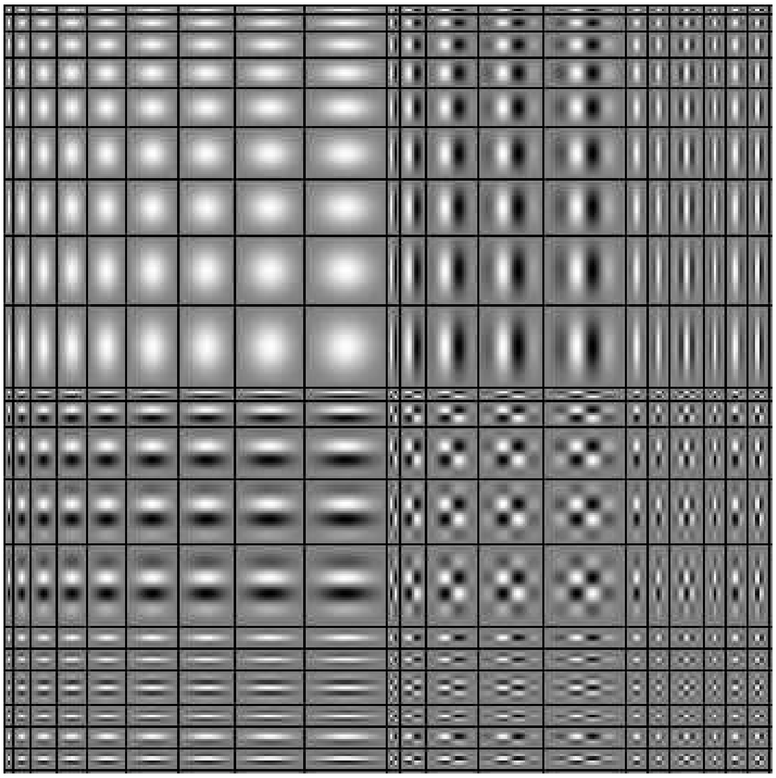
The set of triples $\mathcal{B}$ which we use in our experiments is defined in Table 2.1, with the resulting 2-D separable Gabor dictionary pictured in Figure 2.2. Note that the 2-D dictionary contains functions at many different scales and frequencies, and the extents range from 1×1 to 35×35 pixels. The 1-D functions which define the dictionary were chosen using an ad-hoc training algorithm as follows. A large initial set of 1-D Gabor functions was first formed using regular sampling of scale, frequency and phase shift parameters. The large 2-D dictionary defined by this 1-D set was then used with matching pursuit to decompose motion residual images from a training set. The 1-D components most commonly chosen by the matching pursuit algorithm were then retained for use in the final dictionary. This dictionary was first introduced in [45].

The dictionary pictured in Figure 2.2 effectively forms a large overcomplete set for coding residual images. To understand the size of the dictionary, consider that a complete basis set for a 176×144 QCIF image must contain 25344 elements. The basis used for typical DCT systems satisfies this criterion, having 64 basis functions to represent each of 396 blocks. However, in our codec each basis shape may exist at any integer-pixel location in the image. For a QCIF image, our basis set effectively contains 400 shapes 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 nonzero coefficients than the DCT. This is a tradeoff, since the cost in bits of representing each element increases with the size of the basis set. Our experiments will show that this

Table 2.1: Dictionary triples and associated extents L which define the 1-D Gabor components of our 2-D separable dictionary.

index	s_k	ξ_k	ϕ_k	L
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

Figure 2.2: The 2-D separable Gabor dictionary.



tradeoff is favorable, especially at low bit rates.

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 such as Container Ship which experience an overall change in luminance with time.

Finding Atoms

The previous section defined a dictionary of 2-D structures which are used to decompose motion residual images using matching pursuits. We now describe the details of the decomposition procedure. 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. We thus "pre-scan" the image for a location containing significant residual energy, and we use this location as an initial estimate for the inner-product search. We refer to this scanning procedure as "Find Energy".

The basic "Find Energy" procedure involves dividing the motion residual image into blocks and computing a sum-of-squares energy measure for each block. The block with the largest energy is thus selected as a starting point for the atom search. This method may be used at each atom stage to restrict the matching pursuit inner product search to a small local region.

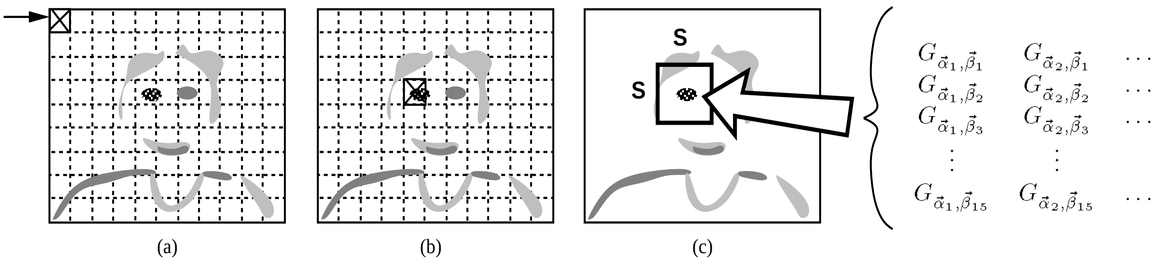
For our current experiments, we use an improved "Find Energy" algorithm introduced in [3]. Again the residual image is partitioned into blocks, but now a counter A_j is associated with each block j. The counters are initialized to zero at the first atom stage, and incremented whenever block j is selected for the local atom search. The counter A_j thus records the number of atoms previously awarded to block j. The winning energy block at each stage is again selected by comparing sum-of-squares energy measures for all blocks, but the energy measure for block j is now multiplied by a heuristic weight before the comparison is made. This weight is a function of A_j, as defined in Table 2.2. As can be seen in the table, the weights decrease with A_j. The resulting weighted "Find Energy" search is thus less likely to select blocks which have been awarded more atoms at previous stages. This has a few desirable effects. First, if the dictionary is unable to efficiently code a particular local pattern in the residual, the weighting scheme forces the encoder to move on to other areas to which the dictionary is more suited. This prevents matching pursuit from spending too many atoms on any small local area. Second, the weighting scheme tends to force a few atoms per frame to be coded in less active areas of the motion residual. This tends to improve objective quality (e.g. PSNR) for low bit rate sequences with static backgrounds, since the quality of the background improves over time. Such results are demonstrated in [3]. However, perceptual improvement due to this effect may be difficult to judge, since it effectively trades a small amount of quality in the moving foreground areas for an improved quality background.

The complete atom search procedure is outlined in Figure 2.3. The motion residual image to be coded is first divided into blocks as shown in Figure 2.3a. The weighted "Find

Table 2.2: Heuristic weights used for improved "Find Energy" search.

A_j	Weight
0	1.00
1	.590
2	.440
3	.350
4	.307
5	.268
6	.240
7	.225
8	.210
9+	.200

Figure 2.3: Inner product search procedure. (a) Partitioning of the image into energy blocks. (b) Weighted find energy determines initial estimate for atom location. (c) Exhaustive inner product search performed in $S \times S$ window in order to select the next atom.



Energy" procedure is then used to select the block which contains the largest weighted energy measure. The selected block is highlighted in Figure 2.3b. The center of this block is then used as an initial estimate for the inner product search. The 2-D Gabor dictionary is then exhaustively matched to an $S \times S$ window around the initial estimate, as shown in Figure 2.3c. In practice, we use 16×16 blocks for the "Find Energy" procedure, and a search window size of $S = 16$.

The exhaustive inner-product search may be thought of as follows. Each $L_h \times L_v$ dictionary element is centered at each location in the search window, and the inner product between that element and the corresponding $L_h \times L_v$ region of image data is computed. The

Table 2.3: Parameters which define an atom

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

image is extended by zero padding whenever a candidate atom extends beyond the image boundaries. The largest computed inner product is used to select the next atom. Inner products are computed efficiently using a separable filtering technique, as will be shown later in this section.

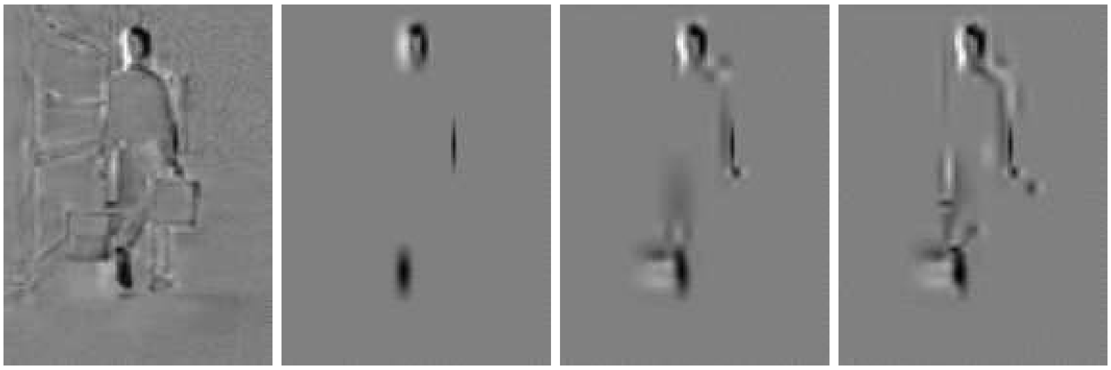
The largest inner product, along with the corresponding dictionary shape and image location, form a set of five parameters, as shown in Table 2.3. We say that these five parameters define an atom, a coded structure within the image. The atom decomposition of a motion residual image is illustrated in Figure 2.4. The motion residual from a sample frame of the Hall sequence is shown in Figure 2.4a. The first five coded atoms are shown in Figure 2.4b. Note that the most visually prominent features are coded with the first few atoms. Figures 2.4c and 2.4d 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 2.4e. A rate control system must be employed to determine the total number of atoms, as will be discussed in Section 2.3.3. For this example, the Hall 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. These remain in Figure 2.4f, which displays the final coding error. The reconstructed frame is shown in Figure 2.4g, and a comparison image coded with H.263 at the same bit rate and frame rate is shown in Figure 2.4h. This is provided for illustration only. Additional comparisons to DCT based systems will be shown in Section 2.4 and also in Chapter 3.

Color information is encoded using a simple extension of the atom search procedure. Weighted "Find Energy" is performed on each of the three color component subimages (Y, U, V), and the largest weighted energy measures from each subimage are determined. The winning values from the chroma components are then multiplied by χ, a chroma weighting constant, before comparison with the winning luminance value. If either of the χ-weighted chroma energy measures exceeds the measure computed for luminance, then a chroma atom is coded. If not, a luminance atom is coded. For the experiments presented here, χ is set to 1.0. However, by changing χ it is possible to adjust the bit rate distribution between the luma and chroma subimages.

Coding Atom Parameters

When the atom decomposition of a single residual frame is found, it is important to code the resulting parameters efficiently. In order to code positions (x, y), we associate each atom with the 16×16 image block in which it is centered. Each block is then coded separately in a raster scan order. Mode information is transmitted for each block in order to indicate what type of motion vector is coded and whether any atoms are present. Motion information is then sent. This includes coded motion vectors or intra block parameters as appropriate. If atoms are present, then the number of atoms centered in the 16×16 block are transmitted using a variable length code (VLC) table. The position of each atom within the block is then sent differentially along a modified spiral scanning pattern. Details of this block-based position coding method are given in [3].

Figure 2.4: 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.



(a) (b) (c) (d)

(e) (f) (g) (h)

The other three atom parameters are coded using VLC tables in the following manner. The basis shape is specified by separate horizontal and vertical components $\vec{\alpha}$ and $\vec{\beta}$, each of which is represented by an index as given in the first column of Table 2.1. Separate VLC tables are maintained for the horizontal and vertical shape indices. Two special escape codewords are added to the horizontal shape table. These are used to indicate when a coded atom belongs to the U or V chroma components. In all cases, the modulus p is quantized by a linear quantizer with a fixed stepsize, and the resulting quantizer index transmitted using a VLC table.

To obtain the VLC tables used in the above description, it is necessary to gather training statistics. We do so using a set of training sequences. Care is taken to avoid training on our test sequences, hence there is no overlap between the training set and the test sequence set defined in Table 1.2.

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 atom stage 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 2.1 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 2.7. The dictionary set may thus be compactly written as

$$G_{h,v}(m,n) = g_h(m)g_v(n) \qquad m \in \{0, ...L_h–1\} \tag{2.8}$$

$$n \in \{0, ...L_v–1\} \tag{2.9}$$

$$h, v \in \{0, ...B–1\} \tag{2.10}$$

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 extents 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 $I(m,n)$ may be written as

$$p = \sum_{m=0}^{L_h-1} \sum_{n=0}^{L_v-1} G_{h,v}(m,n)I(m,n) \tag{2.11}$$

Computing modulus p requires L_hL_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

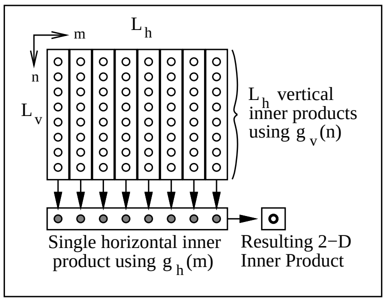
$$T_{nonsep} = S^2 \sum_{h=0}^{B-1} \sum_{v=0}^{B-1} L_hL_v$$

Using the parameters in Table 2.1 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 2.11 can be written instead as

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

Figure 2.5: Separable computation of a 2-D inner product.

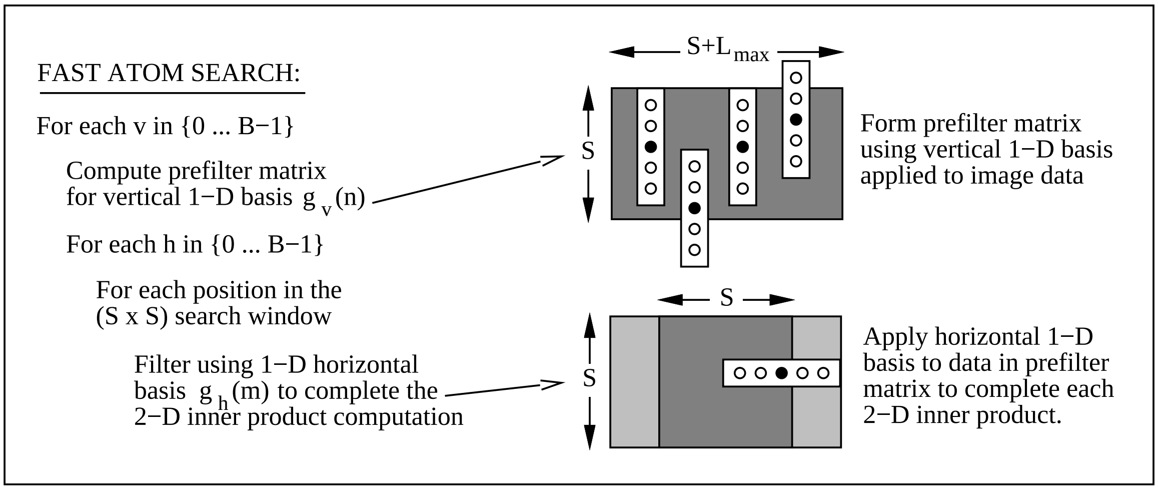


Computing a single 2-D inner product is thus equivalent to taking L_h vertical 1-D inner products, each of length L_v, and then following with a single horizontal inner product of length L_h. This concept is visualized in Figure 2.5. The atom search requires us to exhaustively compute the inner product 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 "prefiltering matrix" to store all the 1-D vertical filtering results for a particular g_v. The resulting search algorithm is shown in Figure 2.6. The operation count for finding a single atom is thus:

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

Note that L_{max} is the size of the largest 1-D basis function. Using the values from Table 2.1, we find that 1.7 million multiply-accumulate operations are required to find each atom. Separable filtering thus gives a speed improvement factor of about 13 over the general nonseparable case.

Figure 2.6: The separable inner product search algorithm.



Even when efficient 2-D separable filtering is used, the exhaustive local inner product search makes the matching pursuit encoder relatively expensive. For comparison, an 8×8 DCT may be efficiently implemented using 522 operations [19]. The luma and chroma components of a QCIF image contain a total of 594 such DCT blocks. The DCT transform complexity used to encode a QCIF motion residual image is thus 3.10×10^5 operations per frame. Using the above analysis for the 2-D separable case, the matching pursuit encoder requires 3.4 million operations to encode *each atom*, where addition and multiplication are counted as separate operations. The number of atoms per frame varies with bit rate, but our experiments typically show about 50 atoms per frame at 10 kbit/s and 100 atoms per frame at 24 kbit/s [45]. Using these values, we estimate that matching pursuit residual coding is about 500 times more complex than DCT for QCIF at 10 kbit/s, and about 1000 times more complex at 24 kbit/s.

Fortunately residual coding is not performed in isolation, and the complexity difference is not as great when motion compensation is considered. Both encoders use the same motion model, which is also relatively expensive. As computed in [45], the full search motion algorithm used in the DCT-based TMN encoder uses about 50 million operations to encode a QCIF frame. When this cost is considered, the overall matching pursuit encoder is about 4.4 times more expensive than the DCT encoder at 10 kbit/s, and about 7.8 times more expensive at 24 kbit/s. Still, complexity reduction is an important goal for matching pursuit research. Some work has already been done on fast matching pursuit algorithms [18], [25]. An alternative is to use the standard algorithm, but design the dictionary to allow fast inner product computation. The separable nature of our Gabor dictionary exploits this to some extent, but more advanced dictionary designs have been shown to provide further complexity reductions [55], [15], [12]. A detailed treatment of this topic will be given in Chapter 6.

2.3.3 Rate Control

We now show that simple and effective per-frame rate control methods exist for the matching pursuit algorithm. Recall that matching pursuit encoding selects atoms iteratively until some stopping criterion is met. Simple rate control may be achieved by defining the stopping criterion in a meaningful way. For example, one could code atoms until a target bit budget for the frame is exhausted. Alternately, atoms may be coded until some target quality is reached for the coded frame.

Suppose the encoder must meet a target bit budget for each frame. The bit rate must be progressively tracked as atoms are added to the current frame so that the encoder knows when the target budget is reached. The exact bit count may be measured at each atom stage i by VLC encoding the i atoms which have already been found. However, we have found that a simpler method is sufficient in practice. An estimate of the cumulative bit rate may be computed by determining B_{atom}, the average number of bits used to code each atom in the previous frame. If this value is relatively constant from frame to frame, then the cost to code N atoms in the current frame is approximately $N \cdot B_{atom}$ bits. For a target budget of B_{target} residual bits, we thus estimate the required number of atoms as:

$$N_{target} = \frac{B_{target}}{B_{atom}}$$

The current residual frame is then coded to N_{target} atoms, and the target bit budget is approximately met. Our experiments show that extremely fine rate granularity may be achieved, and each frame is typically encoded to within 30 bits of the given target rate.

Suppose instead that rate control is to be based on target quality. In this case, the encoder must have some way to progressively track the quality of the reconstructed frame as atoms are added. In this way, the encoder may determine when the target quality has been reached. One way to do this is to compute the remaining signal energy $\|\vec{f_i}\|^2$ after each atom stage, and stop when some target energy threshold is crossed. This method allows the current frame to be coded to a target mean square error (MSE) or to the equivalent peak signal-to-noise ratio (PSNR). Furthermore, $\|\vec{f_i}\|^2$ may be computed progressively at low cost using tools which will be developed in Chapter 4.

Several rate control possibilities are illustrated in Figure 2.7. Each pair of plots shows the bit rate and luminance PSNR (Y-PSNR) plotted against frame number for a matching pursuit encoding of Mother at about 24 kbit/s. Figure 2.7a shows the result when the matching pursuit encoder is run to meet a constant target bit budget of 2273 bits per frame. The plots show a nearly constant bit rate, with the PSNR varying to reflect the coding difficulty of the sequence. We measure the rate targeting accuracy using the average error between true bit rate and the target. For the result shown in Figure 2.7a, this average error is 28.9 bits/frame. Figure 2.7b shows the result when rate control matches a quality target equivalent to a Y-PSNR of 35.3 dB. The plots show that the resulting PSNR is extremely flat for most of the sequence, and the resulting bit rate varies with sequence activity. We again measure targeting accuracy by the average difference between the true Y-PSNR and the target. The average error for the result in Figure 2.7b is 0.042 dB.

Figure 2.7c shows that bit rate targeting may be applied in order to track an arbitrary bit rate trace. The targeting method is unchanged, but the target rate changes from frame to frame. To generate this particular result, we ran the MPEG-4 video codec with a constant quantizer stepsize to encode Mother at about 24 kbit/s. The resulting rate trace was recorded, and the number of bits used for each frame was then used as the target for the matching pursuit encoder. The left plot of Figure 2.7c shows that matching pursuit rate control faithfully tracks the MPEG-4 rate, since the two traces coincide on the plot. The average targeting error is computed to be 33.4 bits/frame. With the bit rates equalized at the frame level, a fair PSNR comparison can be made. We see from the right plot that matching pursuit has a higher Y-PSNR than the MPEG-4 encoder, with an average gain of 0.61 dB.

Figure 2.7d shows that an arbitrary quality trace may also be tracked. We use the same MPEG-4 encoding of Mother as was used in the previous case, but this time the matching pursuit encoder matches the Y-PSNR at every frame. The right plot of Figure 2.7d shows the accuracy of this PSNR targeting. The two traces in the plot coincide, and the average targeting error is computed to be 0.005 dB. In the left plot, the resulting bit rate traces show that matching pursuit uses fewer bits to achieve the same objective quality. While the MPEG-4 result uses an average of 23.5 kbit/s, the matching pursuit result requires only 20.3 kbit/s, for a bit rate savings of 13.8 percent.

Note that the per-frame rate control algorithms above are much simpler than those typically used for DCT based systems. Rate control for DCT is typically based on adjusting the DCT quantizer, either at the frame or macroblock levels. If a single quantizer step is adjusted per frame, then the bit rate may only be controlled to a coarse granularity, and so rate targeting accuracy is limited. Finer control may be achieved by changing the quantizer at the macroblock level. However, this usually requires some type of feedback control system to predict the continuously changing relationship between quantizer step size and bit rate [16], [32]. It also introduces overhead since bits must be spent to specify the changes in the stepsize.

For the majority of our comparisons, we will use the "Matched Bit Rate" scenario illustrated in Figure 2.7c. Matching pursuit rate targeting is used to target the bit rate generated previously by a DCT-based encoder, and a fair comparison of PSNR is then made. This approach is used to generate the results at the end of this chapter. An alternate comparison using "Matched PSNR" will be presented in Chapter 3. As illustrated in Figure 2.7d, Y-PSNR is equalized between the two encoders, and a comparison is made between the number of bits spent by each encoder. As a final note, it is possible to improve matching pursuit coding efficiency by using more elaborate per-frame rate control schemes. We will demonstrate this in Chapter 4 by introducing a rate control method based both on

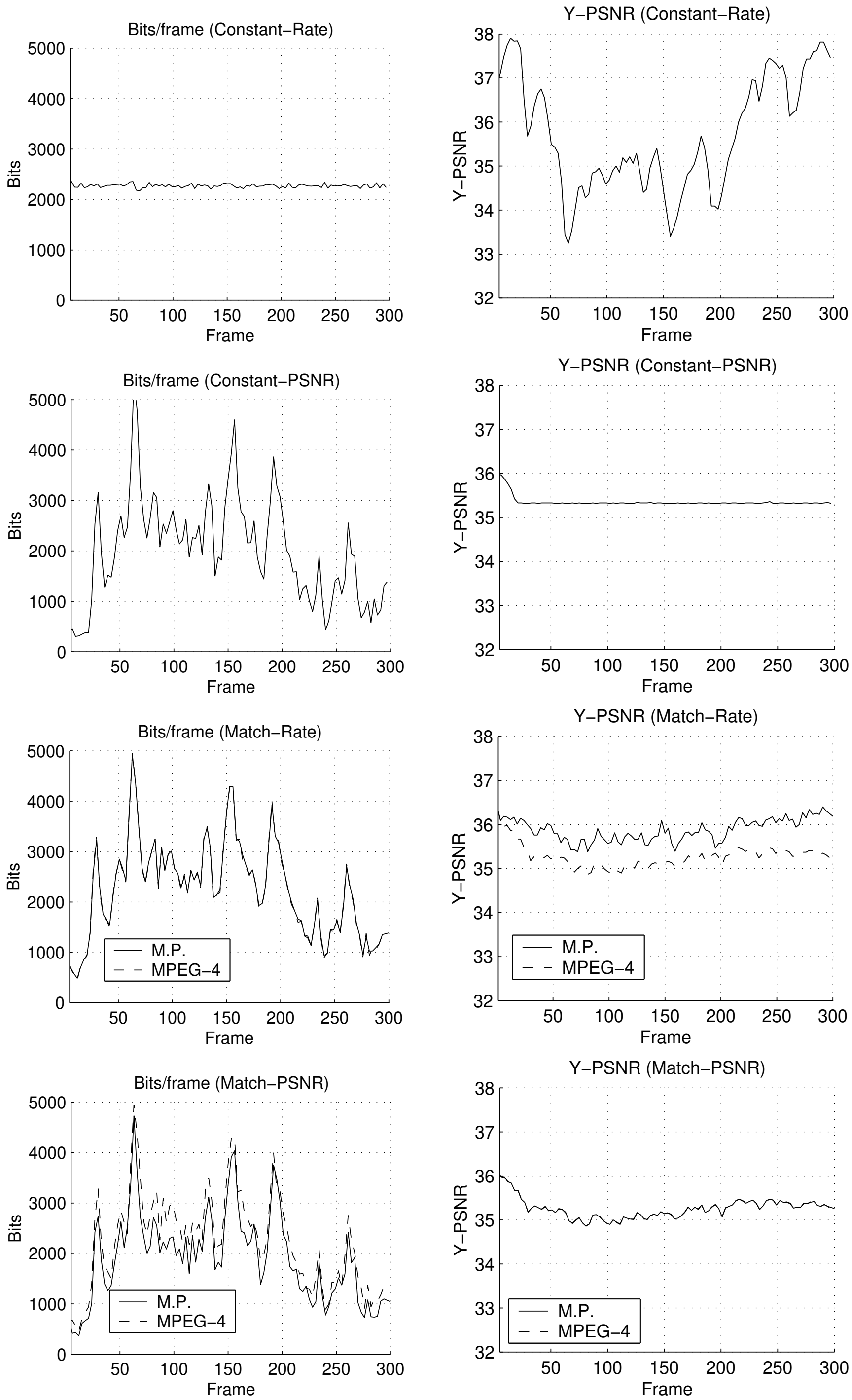
Figure 2.7: An illustration of matching pursuit rate control techniques. Each pair of plots reflects the Mom-24 sequence encoded using a different simple rate control scheme. (a) Constant bit rate per frame. (b) Constant Y-PSNR. (c) Matching an arbitrary bit rate trace. (d) Matching an arbitrary PSNR trace.

(a)

(b)

(c)

(d)



the number of coded atoms and the matching pursuit quantizer design. This will be shown to improve coding performance, especially at high bit rates.

2.3.4 Intraframe Coding

The algorithm described thus far is designed for matching pursuit encoding of motion residual images. For a complete video encoder, some method for encoding intra frames is needed. For our current experiments, we use the DCT-based intraframe encoder from MPEG-4 video [34] with in-loop deblocking and deringing filters. This method will be generally used to encode the first frame, with all remaining frames encoded using motion prediction and matching pursuit residual encoding. We have also considered encoding the intra images using the scalable subband encoder presented in [65]. Earlier results using this method were presented in [45].

Note that matching pursuit is not used to code the intra frames. The reason is that we have found matching pursuit to be much less suitable for intra and still image encoding than for motion residual encoding. Although we have attempted to adapt our basic algorithm to the intra-frame encoding task, such experiments have failed to produce better results than DCT or wavelet-encoded images. Complexity is also a problem. Because intra images cannot be modeled as sparse images with occasional pockets of energy, the "Find Energy" search method may not be applied to reduce the complexity. Intra images require many more encoded atoms than motion residual images, and so the complexity of matching pursuit intra coding may be very high.

Solving these problems is an open topic for future research. At present, the best option is to use an existing DCT or wavelet still image encoder for the intra images. For our comparison to MPEG-4 video, our treatment of intra frames is fair in the sense that identically encoded DCT intra frames are used for both MPEG-4 and matching pursuit encoders. However, it should be noted that our proposed matching pursuit system has a greater implementation cost, since separate codecs must be maintained for intra and motion residual images. In a comparable MPEG-4 system, the hardware or software used to implement the DCT is common to both intra and inter image coding.

2.4 Experimental Results

In this section, we compare matching pursuit to two standardized DCT-based encoders, H.263+ and MPEG-4 video. For H.263+, we use the TMN 3.1.1 encoder from University of British Columbia [47]. For MPEG-4, we use the reference software codec developed by the European MoMuSys consortium [53]. This was one of two software platforms used for testing during the development of the MPEG-4 video standard. We adopt the test conditions used in the MPEG-4 coding efficiency tests. These conditions are now described in detail.

We test each encoder using the 14 sequences specified in Table 1.2. As stated earlier, all VLC table training for the matching pursuit encoder was done using training sequences which are not included in the test set. Each test sequence contains 300 frames representing 10 seconds of video. The desired frame rate is achieved by subsampling original frames. For example, the 10 frame/s rate is achieved by encoding frames $\{0, 3, 6, 9, ...\}$. Frame 0 is intra-coded, with the remaining frames coded using motion prediction and residual coding. Each of the two DCT-based encoders is run with a constant quantizer stepsize.

For the MPEG-4 codec, we use the stepsizes specified for the MPEG-4 coding efficiency tests. These were originally chosen so that the average bit rate for each ten second sequence approximately matches the target rates specified in Table 1.2.

We provide a closely controlled comparison between MPEG-4 video and matching pursuit by linking the encoders in two ways. First, the matching pursuit encoder uses the same DCT-encoded intra frame as MPEG-4 video. Intra-frame artifacts are reduced by turning on the optional deblocking and deringing filters. Second, matching pursuit rate targeting is employed to match the bit rate used at each frame by the MPEG-4 encoder. Because the intra-frame encoding and bit rate are matched in this way, the comparison between MPEG-4 video and matching pursuit is fair. For MPEG-4 video, the deblocking and deringing postfilters are used throughout the sequence. These were shown during the MPEG-4 coding efficiency tests to greatly improve the visual results. The effect of these filters, as well as the decoder complexity cost, will be examined in greater detail in Chapter 3.

An alternate comparison to H.263+ [26] is provided. Because of differences between H.263+ and MPEG-4, neither the intra frame nor the bit rate may be exactly matched to the MPEG-4 and matching pursuit results. We instead choose the quantizer stepsize for each run which most closely matches the average bit rate seen in the matching pursuit and MPEG-4 runs. We also turn on the following optional H.263+ annexes. Unrestricted Motion Vector Mode (Annex D) allows motion vectors to extend beyond the image boundary. Advanced Prediction Mode (Annex F) uses overlapped block motion compensation and allows either 16×16 or 8×8 motion vectors. Note that the features enabled by Annexes D and F are intrinsic to the MPEG-4 motion model. Advanced Intra Coding mode (Annex I) improves the coding efficiency of intra macroblocks and frames by applying predictive encoding to some of the intra-DCT coeficients, and also by modifying the quantizer and VLC tables used to encode the coeficients for intra blocks. Deblocking Filter Mode (Annex J) applies an in-the-loop deblocking filter which adapts its strength based on the DCT quantizer stepsize. Our experiments suggest that this filter is less effective than the MPEG-4 postfilters at removing typical DCT artifacts. Such artifacts are more visible in the H.263+ results than in those of MPEG-4, as our visual comparison will shortly show.

A summary of DCT quantizer steps and resulting bit rates is shown in Table 2.4. Because matching pursuit targets the rate produced by MPEG-4, the average matching pursuit rate always falls within one percent of the MPEG-4 rate. This is shown in the column labelled "(C) vs. (A)." The H.263+ rate is not synchronized frame by frame to either of the other two rates. However, the "(C) vs. (B)" column shows that the average rate is fairly well matched, falling typically within four percent of the matching pursuit rate.

2.4.1 Objective Quality Comparison

For the test conditions described above, Tables 2.5 through 2.7 summarize the objective quality comparison in terms of the average peak signal-to-noise ratio (PSNR) for each coded sequence. The PSNR for each frame is defined in terms of the average mean square error (MSE) taken across all reconstructed image pixels as:

$$PSNR = 10 \log_{10}(255^2/MSE)$$

The tabulated values are then computed by averaging the PSNR per frame across all coded frames in the sequence. Values are computed separately for the luminance and two chromi-

Table 2.4: Average bit rates and DCT quantizer step sizes for each coded sequence. The final two columns show the percentage difference in rate between matching pursuit and each DCT-based encoder.

		Bit rate, kbit/s - [Q-Step]			Δrate, (%)	
		(A)	(B)	(C)	(C)	(C)
Seq	Label	MPEG-4	H.263+	M.P.	vs. (A)	vs. (B)
1	Cont-10	10.35 - [16]	10.22 - [18]	10.39	0.39	1.66
2	Hall-10	10.13 - [17]	10.74 - [17]	10.21	0.79	-4.93
3	Mom-10	10.57 - [14]	10.68 - [13]	10.64	0.66	-0.37
4	Cont-24	26.13 - [9]	26.94 - [9]	26.14	0.04	-2.97
5	Silent-24	25.34 - [13]	25.30 - [12]	25.53	0.75	0.91
6	Mom-24	23.54 - [8]	23.57 - [8]	23.61	0.30	0.17
7	Coast-48	51.58 - [13]	51.45 - [13]	51.54	-0.08	0.17
8	Fore-48	49.59 - [13]	48.98 - [12]	49.49	-0.20	1.04
9	News-48	47.93 - [19]	46.61 - [19]	48.11	0.38	3.22
10	Coast-112	131.05 - [31]	130.38 - [29]	131.10	0.04	0.55
11	Fore-112	137.67 - [31]	130.08 - [26]	137.30	-0.27	5.55
12	News-112	114.67 - [11]	117.61 - [11]	114.88	0.18	-2.32
13	Mob-1024	1019.88 - [14]	1059.48 - [14]	1018.62	-0.12	-3.86
14	Stef-1024	1064.09 - [12]	1023.61 - [11]	1063.16	-0.09	3.86

nance subimages.

Table 2.5 shows a comparison of luminance PSNR (Y-PSNR) for the three systems under consideration. From this table, we see that the matching pursuit encoder produces a higher Y-PSNR than MPEG-4 for all 14 test sequences. The largest gain is 1.64 dB, achieved for Foreman at 112 kbit/s. Gains of 1 dB or more are seen for half of the test sequences. Comparing matching pursuit to H.263+, we see that matching pursuit achieves a higher PSNR for all sequences except Stefan at 1 Mbit/s.[1] Again, a gain of at least 1 dB is seen for half the test sequences. The maximum gain over H.263+ is 2.32 dB, as seen for Container Ship at 10 kbit/s.

Tables 2.6 and 2.7 show the corresponding PSNR results for the U and V chromi-

[1] A possible reason for the exception is that the H.263+ encoder implementation used a greater motion vector search range than the motion encoder used for matching pursuit and MPEG-4. This provides an advantage for high motion sequences like Stefan.

Table 2.5: Luminance PSNR (Y-PSNR) comparison between MPEG-4 video, H.263+, and matching pursuit.

Seq	Label	(A) MPEG-4	(B) H.263+	(C) M.P.	(C)-(A)	(C)-(B)
1	Cont-10	29.88	29.13	31.45	1.57	2.32
2	Hall-10	30.30	30.37	31.79	1.50	1.42
3	Mom-10	32.64	32.92	33.25	0.61	0.33
4	Cont-24	33.38	33.11	34.37	0.99	1.26
5	Silent-24	31.04	31.49	32.28	1.24	0.79
6	Mom-24	35.27	35.20	35.88	0.61	0.68
7	Coast-48	29.42	29.40	30.23	0.81	0.83
8	Fore-48	31.14	31.19	31.51	0.37	0.31
9	News-48	31.14	31.15	32.50	1.36	1.35
10	Coast-112	26.07	26.52	27.68	1.60	1.15
11	Fore-112	28.66	29.07	30.30	1.64	1.22
12	News-112	34.23	34.06	35.55	1.32	1.49
13	Mob-1024	26.45	26.08	27.01	0.56	0.92
14	Stef-1024	29.49	30.25	29.86	0.37	-0.39

nance subimages. The results are similar to those of luminance in that the matching pursuit encoder consistently achieves a higher chrominance PSNR than either DCT based encoder. The maximum gains in chrominance are seen to be above 3 dB.

A more detailed look at the objective improvement is shown in Figure 2.8 which plots Y-PSNR against frame number for the three tested codecs. Sample sequences were chosen to represent the complete range of tested bit rates. In each plot, the matching pursuit codec achieves a higher Y-PSNR than either DCT-based codec for the majority of the ten second sequence. There is no clear winner between the DCT based codecs. MPEG-4 exceeds H.263+ in Y-PSNR for Cont-24 and Mobile-1024, but the reverse is true for Silent-24 and Fore-112.

Note that matching pursuit and MPEG-4 share the same coded intra frame. These two systems thus have an identical Y-PSNR for frame 0. The MPEG-4 quantizer stepsizes

Table 2.6: U-PSNR comparison between MPEG-4 video, H.263+, and matching pursuit.

Seq	Label	(A) MPEG-4	(B) H.263+	(C) M.P.	(C)-(A)	(C)-(B)
1	Cont-10	37.00	36.69	37.36	0.37	0.67
2	Hall-10	36.52	36.20	36.59	0.07	0.39
3	Mom-10	38.73	38.68	38.91	0.18	0.23
4	Cont-24	39.45	39.13	40.42	0.97	1.29
5	Silent-24	35.26	36.08	36.79	1.53	0.71
6	Mom-24	40.13	40.83	40.99	0.86	0.15
7	Coast-48	40.00	39.58	40.63	0.63	1.05
8	Fore-48	37.22	37.46	38.20	0.98	0.74
9	News-48	35.93	35.93	37.69	1.76	1.76
10	Coast-112	38.05	37.50	38.20	0.16	0.70
11	Fore-112	35.41	35.13	36.67	1.27	1.55
12	News-112	38.27	38.25	40.19	1.91	1.93
13	Mob-1024	30.82	30.58	33.80	2.98	3.22
14	Stef-1024	34.85	35.61	36.84	1.99	1.22

Table 2.7: V-PSNR comparison between MPEG-4 video, H.263+, and matching pursuit.

Seq	Label	(A) MPEG-4	(B) H.263+	(C) M.P.	(C)-(A)	(C)-(B)
1	Cont-10	36.53	35.79	37.64	1.12	1.86
2	Hall-10	39.63	38.90	39.75	0.13	0.85
3	Mom-10	39.65	39.47	40.18	0.53	0.71
4	Cont-24	38.63	38.52	40.53	1.90	2.01
5	Silent-24	36.93	37.53	37.85	0.91	0.31
6	Mom-24	40.91	41.32	41.76	0.85	0.44
7	Coast-48	41.90	41.28	42.91	1.01	1.63
8	Fore-48	37.39	37.56	38.83	1.44	1.27
9	News-48	37.39	37.29	38.35	0.96	1.07
10	Coast-112	40.27	39.14	41.05	0.78	1.91
11	Fore-112	35.90	35.62	37.83	1.93	2.21
12	News-112	39.08	39.31	40.69	1.61	1.38
13	Mob-1024	30.35	30.06	33.42	3.07	3.35
14	Stef-1024	34.48	35.32	36.75	2.27	1.43

which produced these intra frames were given in column (A) of Table 2.4. MPEG-4 and H.263+ do not share the same intra coding method, and so the initial frame for H.263+ differs. For Hall-10 and Silent-24, the plots show that the H.263+ encoder produces a higher quality intra frame. Despite this advantage, we see that the matching pursuit encoder catches up quickly and is able to outperform the H.263+ result later in the sequence.

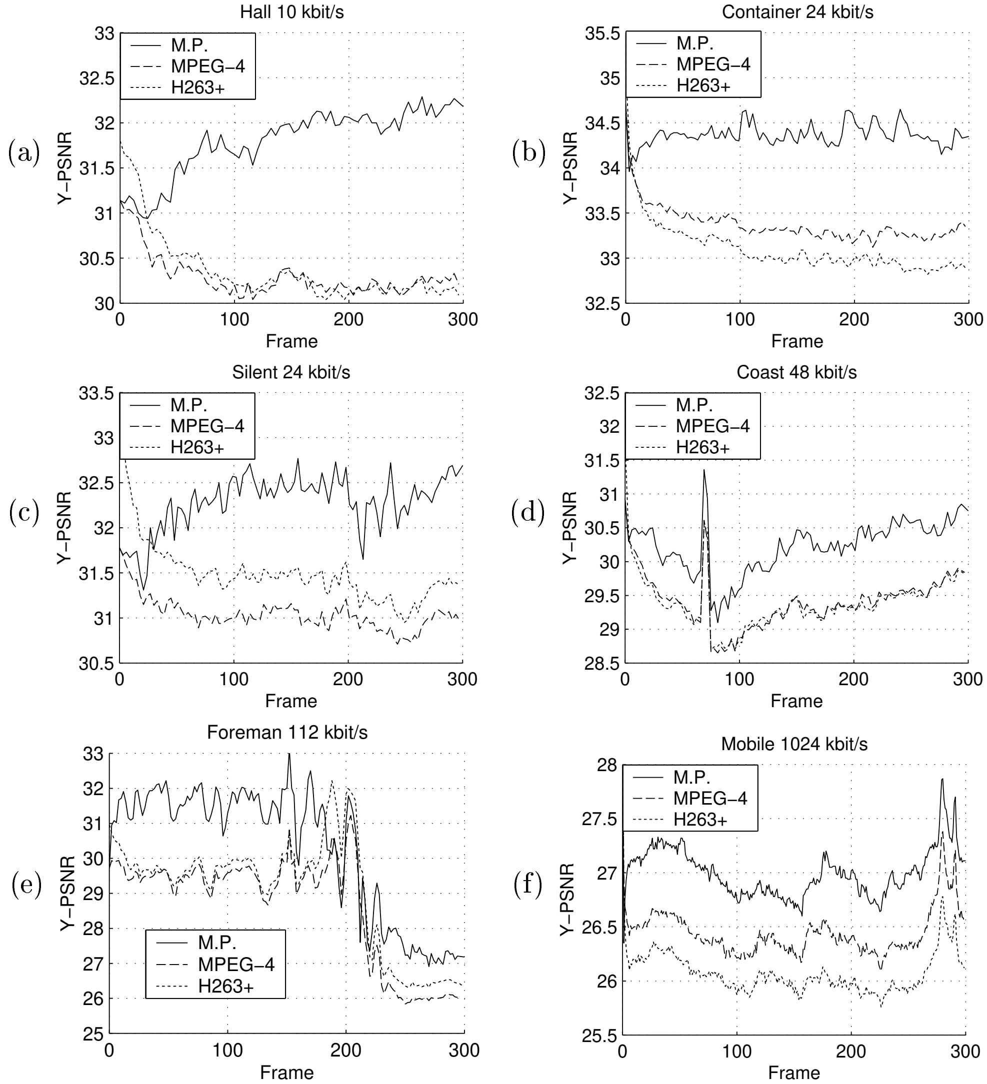
2.4.2 Visual Comparison

Figures 2.9 through 2.11 show visual comparisons of encoded frames. In each figure, the original image is shown in the upper-left position, the H.263+ and MPEG-4 video images are shown in the upper-right and lower left positions, respectively, and the matching pursuit image is shown at the lower right.

Figure 2.9 shows frame 80 of Mother encoded at 24 kbit/s. The matching pursuit image is clearer and preserves more of the original detail than either of the two DCT-coded images. This can be seen by looking at the mouth, ear and hair regions of the mother, or by looking at the hair and eyes of the child. Serious ringing artifacts can be seen in the H.263+ image, particularly where the mouth, nose, and hair meet the background. These artifacts are reduced in the MPEG-4 video image through the use of a de-ringing filter which will be discussed in Chapter 3. The MPEG-4 image is thus more visually pleasing than that of H.263+, but it appears flat and lacks detail compared to the matching pursuit image. One reason for this is that the postfilters which are used to remove DCT artifacts also remove some of the coded detail.

Figure 2.10 shows frame 120 of Foreman encoded at 112 kbit/s. Again, the matching pursuit encoding is seen to preserve much more coded detail than either of the two DCT

Figure 2.8: Plots of Y-PSNR vs. frame for sample test sequences. Matching pursuit is compared to both MPEG-4 video and H.263+.



codecs. A striking example of this can be seen by comparing the Siemens logo in the upper left corner of each coded images. The logo is perfectly clear and readable in the matching pursuit image. The logo is surrounded by artifacts in the H.263+ image, and is is distorted and illegible in the MPEG-4 image. Matching pursuit also preserves more detail in the face and clothing of foreman. This is particularly evident in the ear, nose and mouth regions of the face, and also in the collar of foreman's shirt. Finally, matching pursuit preserves more details on the background building. While the matching pursuit image shows some of the concrete texture present in the original image, this texture is largely absent from the two DCT-coded images.

Figure 2.9 shows frame 200 of Mobile encoded to 1 Mbit/s. The biggest difference between matching pursuit and the two DCT-based codecs is again the sharpness of edges and the preservation of details. For example, the matching pursuit image preserves the eyes of the blue donkeys and the nail holes on the yellow fence in the center of the image. Such details are faded or absent on the two DCT-encoded images. The DCT images also blur the details on the train and the tracks. These details are better preserved in the matching pursuit image. Finally, both DCT codecs introduce artifacts on the red spotted ball which rolls along in front of the train. In the matching pursuit image, the ball appears brighter, clearer, and more closely matched to that of the original image.

Although we believe the matching pursuit encoder produces significant visual improvement over DCT-based encoders, it is important to understand the types of artifacts that may be seen in matching pursuit coded sequences. The most obvious is a gradual loss of sharpness and detail as bit rate is reduced. This occurs because fewer atoms are available

Figure 2.9: Frame 80 of Mother at 24 kbit/s. (a) Original image. (b) DCT-coded with H.263+. (c) DCT-coded with MPEG-4 Video. (d) Matching pursuit coded.

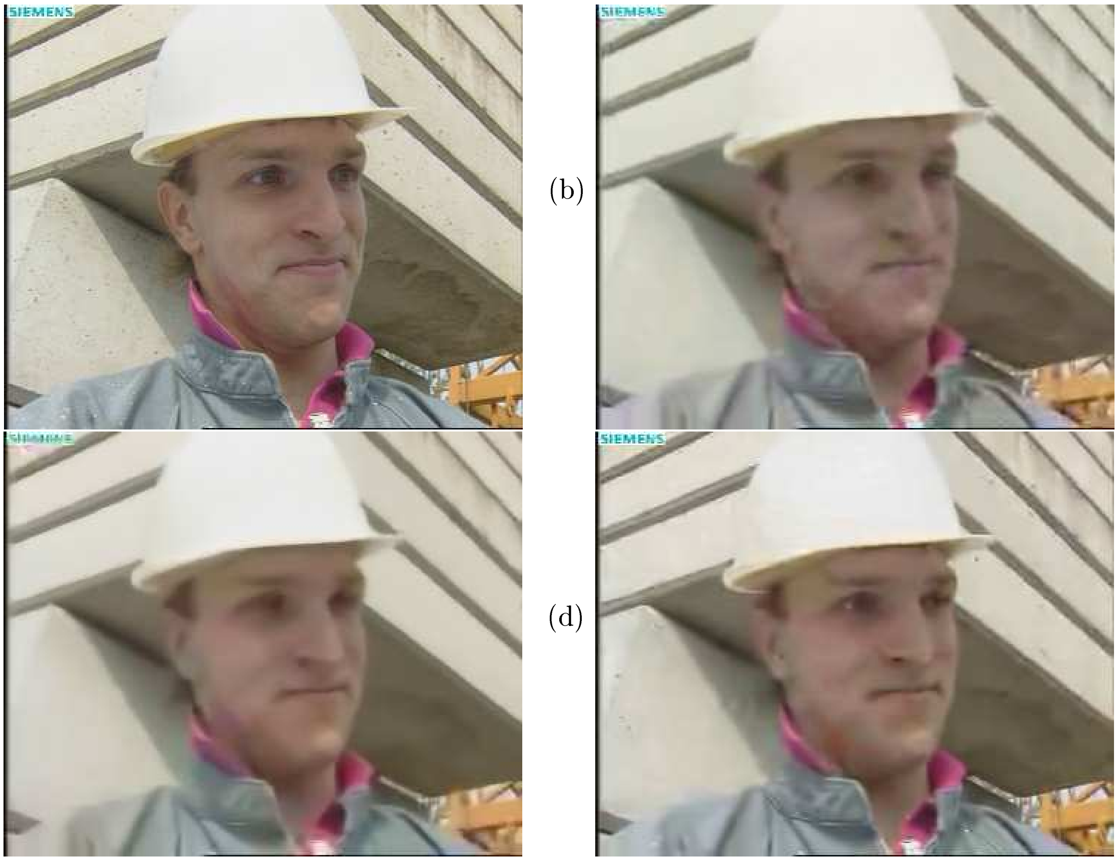
(a)

(b)

(c)

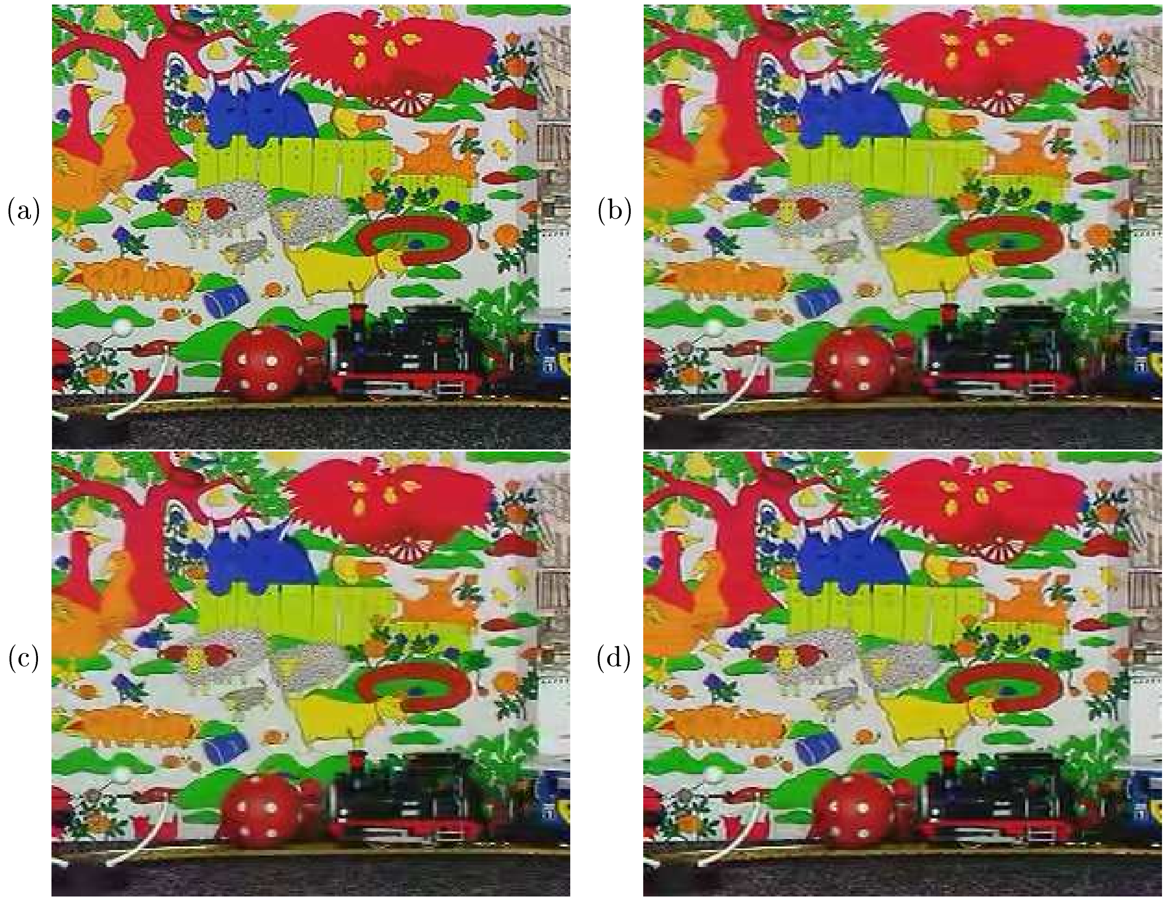
(d)

Figure 2.10: Frame 120 of Foreman at 112 kbit/s. (a) Original image. (b) DCT-coded with H.263+. (c) DCT-coded with MPEG-4 Video. (d) Matching pursuit coded.



(a) (b) (c) (d)

Figure 2.11: Frame 200 of Mobile at 1 Mbit/s. (a) Original image. (b) DCT-coded with H.263+. (b) DCT-coded with MPEG-4 Video. (c) Matching pursuit coded.



to fill in the details in the residual error image. A second type of artifact is a temporal flickering in fine details with motion. One explanation is that the overlapping block motion model has a low-pass filtering effect which smooths over some detail as it predicts motion. Each coded frame thus loses some detail in the motion prediction, and tries to rebuild that detail with residual coding. However, the number of matching pursuit atoms assigned to a local region varies from frame to frame based on the relative need for atoms in other areas. This means that in some frames the detail is rebuilt in a satisfactory way, and in other frames it is not.

A third type of artifact is a ghosting effect which may occur in sequences with low bit rate and frame rate. For example, in the container ship sequence a bird flies quickly across the screen near the end of the sequence. When this scene is coded at 10 kbit/s and 7.5 frame/s, the moving bird leaves a noticeable trail of artifacts in the water as it passes. These artifacts occur because of a combination of effects. First, the block translational motion model is not sufficient to capture the large motion of the bird between frames. This leaves matching pursuit much of the task of building the bird in its current location, and at the same time erasing traces of the bird at its previous location. At low bit rates the number of atoms is insufficient for these two tasks, and so distortion is introduced at both the current and previous locations. However, the distortion at the previous location is more noticeable since it takes the form of spurious edges in the relatively smooth water. The result is a visible artifact trail that persists for some time.

2.4.3 Discussion

Given the results of the previous two subsections, we now highlight a few sequences and attempt to explain qualitatively how they are encoded by matching pursuit. In this way, we may gain insight into the encoding process, and perhaps explain why matching pursuit encoding is more effective in some situations than others.

In the low bit rate container sequence, there are two principal moving objects. A large container ship moves slowly from right to left, and a small boat moves slowly from left to right. The block translational motion model tends to represent these objects well, and so matching pursuit residual atoms are mainly used to correct deforming motion of the small boat and its wake, and to continually correct and enhance details on the container ship. Two additional features must be encoded. First, the brightness of the scene gradually changes with time. This effect is coded efficiently using the global luminance function described along with the dictionary set in Section 2.3.2. This allows a single value to be sent as an update to overall luminance, and avoids the need to cover the scene with matching pursuit atoms in order to update the brightness. A second feature, the bird which flies across the water near the end of the scene, was discussed in detail at the end of the previous subsection. We may conclude that the matching pursuit algorithm is suitable for coding both ships and the overall luminance changes, but lacks tools to effectively code the fast-moving bird at such low rates.

In the low bit rate hall sequence, there is a static hallway background with two walking figures superimposed. The walking movement is non-translational, and so the motion model continually leaves pockets of residual energy in the vicinity of each walker.

There are thus two distinct local areas which constantly attract matching pursuit atoms. Because many atoms are clustered into these relatively small areas, the differential atom position coding method allows the positions to be coded efficiently. This case differs from a more spatially distributed scene like Coast Guard. In this scene, the scrolling background and continually changing water surface require residual atoms to be assigned more randomly and over a larger area. For this case, we expect a higher per-atom position cost than that of hall.

In the higher rate Foreman sequence, the majority of the scene involves Foreman talking in front of a background building. Motion is extensive and comes from two sources. First, the Foreman's head and face are extremely animated as he talks. Second, the camera makes jerky random motions which result in movement of both the foreground head and the background building. During this part of the sequence, matching pursuit is constantly dividing the atom budget between foreground and background areas in the residual. When the Foreman makes a particularly large deforming facial movement, the motion model fails and some blocks may even be coded in intra mode. In such cases, many atoms are attracted to needy local areas on the face, and few if any background atoms are coded. When the Foreman pauses to breathe, the encoder is able to assign more atoms to the background building and the detail in this area is improved. The encoding of Foreman may thus be thought of as a frame-by-frame competition for atoms between foreground and background areas. Further exploration of this idea will be carried out in the next subsection.

2.4.4 Spatial Bit Allocation

We close this section with a look at the way matching pursuit allocates bits spatially within the coded frame. Recall that matching pursuit considers the entire frame at once and encodes atoms in order of decreasing energy importance, regardless of where they appear spatially. This is a fundamental difference from block-based DCT systems in which each image block is coded and transmitted separately, usually in some predetermined coding order. The DCT encoder typically must predict the coding difficulty of future macroblocks in order to properly allocate bits to the current block.

The use of a "Find Energy" technique changes the way bits are allocated by the matching pursuit encoder. The simple "Find Energy" method introduced in [45] is essentially greedy, awarding each atom to the local area with the highest energy measure. The weighted "Find Energy" method introduced in [3] is modified to be less greedy, selecting against local areas which have received atoms at previous stages. When this method is used, spatial bit allocation changes to assign more bits to the less energetic areas of the frame. We now show how more direct control of spatial bit allocation may be easily added to the existing algorithm.

Suppose a sequence is to be encoded in which the relative importance of each local area is known. To use the simple example of a video telephone sequence, one might desire the facial region to be coded to a higher quality than the background. In this case, a region of interest (ROI) is identified, and the encoder is somehow instructed to assign additional bits to this region. For the matching pursuit encoder, redistribution of bits to a known region of interest may be achieved with very little modification and no additional

Figure 2.12: (a) Standard initialization of block counters A_j for weighted energy search. (b) Modified counter initialization allows region of interest coding.

(a)

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

(b)

a	a	a	a	a	a	a	a	a	a	a
a	a	a	a	a	a	a	a	a	a	a
a	a	a	0	0	0	0	0	a	a	a
a	a	a	0	0	0	0	0	a	a	a
a	a	a	0	0	0	0	0	a	a	a
a	a	a	0	0	0	0	0	a	a	a
a	a	a	0	0	0	0	0	a	a	a
a	a	a	0	0	0	0	0	a	a	a
a	a	a	a	a	a	a	a	a	a	a

complexity. Recall that for the weighted "Find Energy" search, a counter A_j is used to track the number of atoms previously awarded to each energy block j. The counters are initialized to zero before the first atom stage, as illustrated in Figure 2.12a. However, by initializing some of the counters to a larger value a, fewer atoms will be awarded to the corresponding blocks. One application of this idea is shown in Figure 2.12b. A central ROI is identified, and the counters outside this ROI are initialized to a instead of zero. Spatial bit allocation is thus changed to deflect more bits into the central region.

A coding result based on this technique is shown in Figure 2.13. The Fore-112 sequence is coded using both standard and modified counter initialization according to the patterns shown in the previous figure. Standard counter initialization is shown in Figure 2.13a, while the modified method with $a = 9$ is shown in Figure 2.13b. The modification greatly improves the facial detail, as can be seen by comparing the teeth, eye, and nose regions. However, this improvement comes at the expense of background quality, and so a reduction in detail is seen on the building in Figure 2.13b.

This example illustrates how naturally spatial bit allocation is handled in the existing matching pursuit encoder. It also shows that dramatic effects may be achieved

Figure 2.13: Sample result for matching pursuit region of interest (ROI) coding using frame 26 of Foreman encoded at 112 kbit/s. (a) Standard counter initialization. (b) Counters for blocks outside ROI initialized using $a = 9$.

(a) (b)

with very minor modifications. However, more general application of this technique requires sequence analysis in order to identify and prioritize regions of interest. While we will not explore this topic here, we do note that region of interest segmentation has appeared extensively in the literature (e.g. [59], [6], [60]).

Also note that a similar scheme was shown previously for a matching pursuit video codec in [5]. In that work, the basic encoder was modified to assign atoms based on spatial proximity to the image center. However, no visual demonstration of the improvement was provided.

2.5 Conclusions

In this chapter, we have described how the matching pursuit technique may be used to encode motion residual images in a hybrid video codec. The resulting video compression system was shown to outperform both MPEG-4 Video and H.263+ in terms of both PSNR and visual quality. In the objective quality comparison, matching pursuit was shown to achieve higher PSNR than either DCT based codec. Y-PSNR gains between 1 and 2 dB were shown for half the tested sequences. In the visual comparison, matching pursuit was shown to preserve more coded detail than either DCT-based codec. Matching pursuit results were also relatively free from systematic artifacts like ringing or blocking. Such artifacts are often produced by DCT codecs, which then require postfiltering for artifact removal. A closer look at postfiltering will be given in the next chapter.

In the course of our description, we introduced a few other useful features of the matching pursuit codec. First, we showed that frame level rate control for matching pursuit is both simple and effective. We demonstrated ways in which the matching pursuit encoder could accurately target either bit rate or coded quality at the frame level. Such targeting was then used to synchronize the matching pursuit bit rate to that of MPEG-4 and thus provide a fair PSNR comparison. We could even argue that such a comparison is biased toward the MPEG-4 system, since the matching pursuit encoder is simply adopting the rates chosen by that system. Secondly, we showed that the existing matching pursuit algorithm handles spatial bit allocation in a very logical way. We then demonstrated that the bit allocation could be manipulated to useful effect, and such manipulation requires only minor modification at the matching pursuit encoder.

Amid these interesting features and comparisons to existing standards, the main purpose of this chapter was really to provide a solid system description on which to build the remaining chapters of this work. These chapters will further explore the behavior of matching pursuit based video compression, and will also develop improvements in terms of coding efficiency and complexity.