Atom Position Coding in a Matching Pursuit Based Video Coder

by

Pierre Garrigues

Ing. Ecole Polytechnique 2003

A thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science
in

Engineering - Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Fall 2005

Atom Position Coding in a Matching Pursuit Based Video Coder

Abstract

Atom Position Coding in a Matching Pursuit Based Video Coder

by

Pierre Garrigues

Master of Science in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Avideh Zakhor, Chair

In this thesis, we propose a new scheme for position coding of the atoms within the Matching Pursuit algorithm as applied to the displaced frame difference (DFD) in a hybrid video encoder. Atom position coding represents up to 40% of the bit rate in an encoded video sequence if it is not rapidly changing. Thus, optimizing the position coding is likely to have a significant impact on the overall encoding bit rate of the video sequence. The positions of the atoms are not identically distributed within the DFD. As such, we have to capture statistics of the atom positions to be able to compress them efficiently. To do so, we exploit the spatial and temporal coherence of the atom positions in the DFD: the atoms tend to cluster in regions where the motion prediction fails. The location of these regions is correlated over time. Thus, we propose a block-based technique in which the number of atoms inside each block is coded differentially with respect to the same block in the previous frame, and the atom positions inside each block are coded using the previously proposed NumberSplit algorithm. We demonstrate the effectiveness of our proposed approach on a number of video sequences, and show bit rate saving of up to 11%.

Professor Avideh Zakhor
Thesis Committee Chair

Contents

Contents i

1 Introduction 1

2 Matching Pursuit Based Video Coding 3

2.1 Matching Pursuit Theory . . . 3

2.2 Application of Matching Pursuit to Video Coding . . . 4

2.3 Parameters of the Overcomplete Expansion . . . 6

3 Atom Position Coding 9

3.1 Entropy for IID data . . . 9

3.2 Frame based position coding . . . 11

3.3 Block-based position coding . . . 12

3.4 NumberSplit . . . 13

4 Proposed Position Coding Scheme 15

4.1 The temporal coherence . . . 15

4.2 The proposed algorithm . . . 16

4.3 The choice of the block size . . . 17

4.4 Atom position coding within blocks . . . 17

5 Results 20

6 Conclusion 28

Bibliography 29

References . . . 29

Chapter 1

Introduction

There has been a tremendous development of multimedia applications recently. The increased capacity of cellular networks and of the internet connections make it possible for the end users to receive videos on their cell phones and to watch videos on their personal computer streamed from website servers. There is now a huge demand for these applications, and the expectations of the users in terms of quality of service are high. It is thus important to develop efficient video compression algorithms at medium to low bit rates.

The hybrid motion-compensated Discrete Cosine Transform (DCT) structure or a DCT-like integer transform structure are a part of nearly all existing video coding standards such as H.263, H.264/AVC and MPEG-1,2,4, and commercially available video compression systems. In each of these systems, a block-based motion model is used to predict each frame from a previously reconstructed frame, and the residual energy in each motion block is coded using the DCT. One of the problems with DCT is the blocking artifacts typically associated with all block-based transforms. To circumvent blocking artifacts, a number of non-block-based schemes based on expansion of signals on overcomplete set of basis functions have been proposed. In this thesis, we focus on Matching Pursuit (MP), originally proposed in the statistical literature [4], and later reintroduced to the signal processing community by Mallat and Zhang [8]. Application of MP to residual video coding was originally proposed in [11], and further developed in [1] [2] [10] [7] [9] [5]. MP-based codecs have been found to result in significant PSNR and visual quality improvement over DCT-based schemes especially at low bit rates. This is because they contain no blocking artifacts.

In analyzing the coding statistics of MP-based video codec in [1], we have empirically determined that a significant portion of the bits are devoted to atom position coding. Other sources of bit consumption include motion vectors, atom modulus coding and indices of the elements in the dictionary. Specifically, at higher bit rates, position coding represents up to 40% of the bit rate. Since the performance gap between MP and block-based techniques decreases at higher bit rates, it is conceivable to reduce this gap by more efficient position coding techniques, and to increase its performance at low bit rate.

In this thesis, we propose a new position coding technique based on the exploitation of the spatial and temporal coherence of the atom positions. The outline of the paper is as follows. In Chapter 2 and 3, we review the Matching Pursuit theory and existing position coding techniques respectively. Our new position technique and its application to position coding are discussed in Chapter 4. Finally, experimental results are included in Chapter 5.

Chapter 2

Matching Pursuit Based Video Coding

2.1 Matching Pursuit Theory

The Matching Pursuit algorithm, as proposed by Mallat and Zhang [8], expands a signal onto an overcomplete dictionary of functions. For simplicity, the procedure can be illustrated for a 1-D time signal. Suppose we want to represent a signal $f(t)$ using basis functions from a dictionary set Γ. Individual dictionary functions are denoted as:

$$g_\gamma(t) \in \Gamma$$

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

$$p = < f, g_\gamma >$$

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

$$R(t) = f(t) - p g_\gamma(t)$$

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

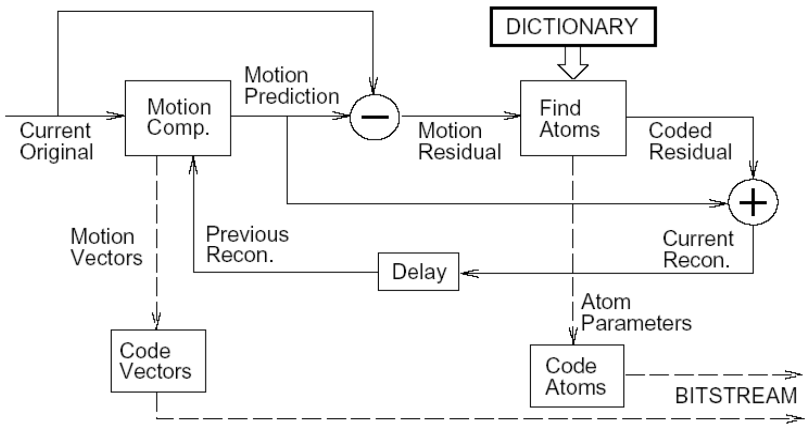
$$\hat{f}(t) = \sum_{n=1}^{M} p_n g_{\gamma_n}(t)$$

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

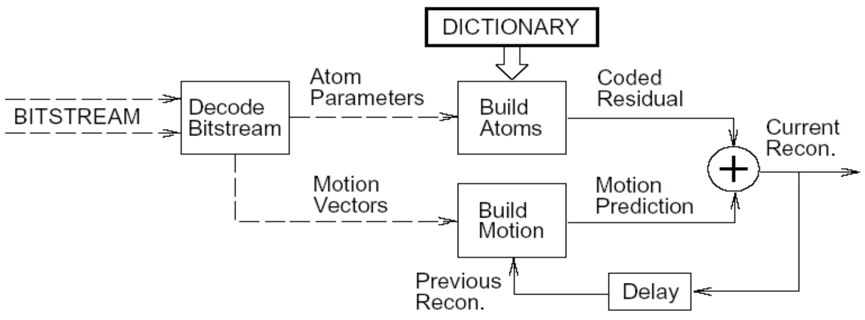
2.2 Application of Matching Pursuit to Video Coding

Most video compression systems in use today are built on a hybrid motion-compensated transform structure. Such systems predict the current frame using motion compensation, and then transmit the prediction error residual using a transform, usually the discrete cosine transform (DCT). Neff and Zakhor demonstrated that improved coding efficiency may be achieved by replacing the DCT with an overcomplete transform. The block diagram for the encoder and the decoder are shown in Figure 2.1. The gain in performance is due to the fact that the prediction error residual or displaced frame difference (DFD) is usually a sparse signal. It has a few pockets of energy where the motion prediction fails. Figure 2.2 shows a typical DFD from the sequence Hall. By decomposing the DFD onto an overcomplete dictionary, we only need a few elements from the dictionary, and thus few iterations of the

MP algorithm to obtain a reasonable approximation of the DFD. This means that there is only a small number of elements from the dictionary to be coded, typically in the order of a hundred for a QCIF resolution video at 10 frames per second. We now describe in details the coding of the overcomplete expansion.



(a) Encoder



(b) Decoder

Figure 2.1. Block-diagram for the encoder and the decoder

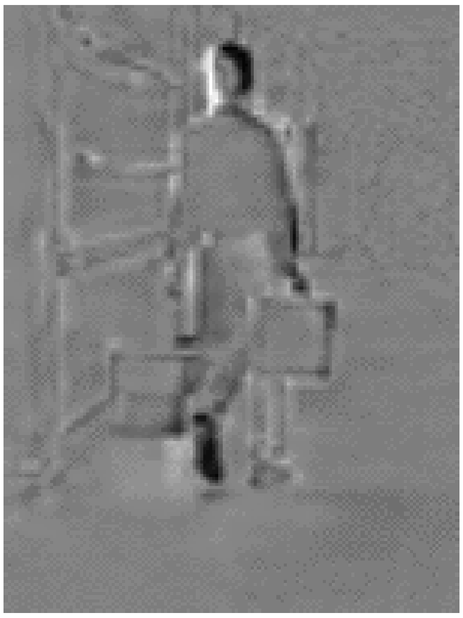


Figure 2.2. A DFD from the video sequence Hall

2.3 Parameters of the Overcomplete Expansion

Let $f(i,j)$ denote the DFD. After M iterations of the MP algorithm, we obtain the following approximation of the DFD:

$$\hat{f}(i,j) = \sum_{n=1}^{M} p_n g_{\gamma_n}(i,j).$$

The number of iterations is determined via the rate control option we are using. If we encode a video at constant quality, we iterate the MP algorithm until the distortion goes below a specified threshold. If we encode a video at a fixed bit rate, we iterate the MP algorithm until the bid budget for the coding of the DFD is exhausted.

The dictionary set we use in this thesis consists of an overcomplete collection of 2-D separable Gabor functions [11] that are shown in Figure 2.3. We define this set in terms of a prototype Gaussian window:

$$g(t) = 2^{\frac{1}{4}} e^{-\pi t^2}.$$

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

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

Here $\overrightarrow{\alpha} = (s, \xi, \phi)$ is a triplet consisting of a positive scale, a modulation frequency, and a phase shift respectively. The constant $K_{\overrightarrow{\alpha}}$ is chosen such that the resulting sequence is of unit norm. If we consider $\mathcal{B}$ to be the set of all such triplets $\overrightarrow{\alpha}$, then we can specify our 2-D separable Gabor functions to be of the following form:

$$G_{\overrightarrow{\alpha}, \overrightarrow{\beta}}(i, j) = g_{\overrightarrow{\alpha}}(i) g_{\overrightarrow{\beta}}(j) \qquad i, j \in \{0, 1, \cdots, N-1\}, \quad \overrightarrow{\alpha}, \overrightarrow{\beta} \in \mathcal{B}.$$

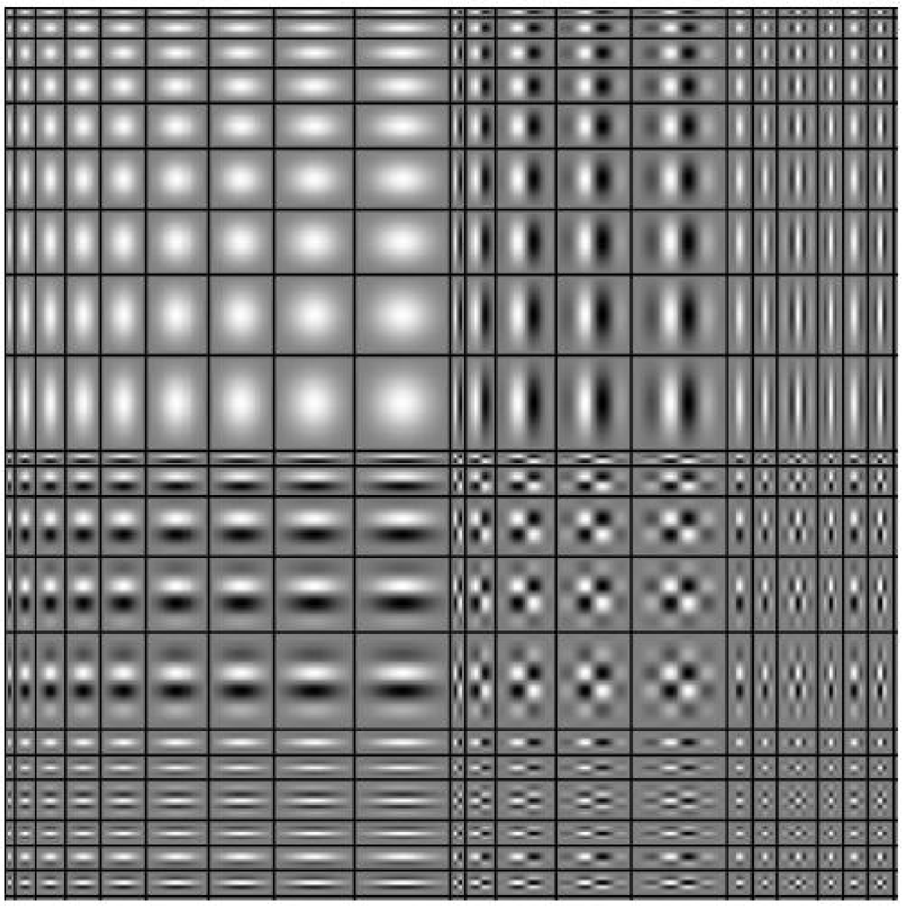


Figure 2.3. Separable two dimensional Gabor dictionary

At stage n of the MP algorithm, the largest inner product p_n, along with the indexing parameter γ_n define an atom. γ_n consists of (a) the two parameters that define the 2-D separable Gabor functions, e.g. $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$, and (b) the two position parameters that define its location in the residual image. When the atom decomposition of a single residual frame is computed, it is important to code the resulting parameters efficiently to minimize the resulting bit rate. p_n and the parameters defining the structure of the dictionary are typically coded using fixed Huffman codes. In the remainder of this thesis we investigate the coding of the position parameters. We refer to this as atom position coding.

Chapter 3

Atom Position Coding

We begin this Chapter by deriving an expression for the entropy of independent identically distributed (IID) position parameters of n atoms within a frame of size N pixels. We then provide a brief overview of existing position coding techniques.

3.1 Entropy for IID data

Let us consider the problem of coding the position parameters of n atoms in a frame of size N pixels assuming the positions to be uniformly identically distributed within the frame and independent of each other. We allow several atoms to be at the same position, as it does happen in practice. We now derive the number of possible placements of the n atoms within a frame with N pixels.
Let $\mathcal{F}$ denote the set of all possible placements of n indistinguishable balls in N boxes, where we allow several balls per box. Let $\mathcal{G}$ denote the set of all possible placements of n indistinguishable balls in $N + n - 1$ boxes, where we can have at most a one ball per box. It is easy to see that $\#\mathcal{G} = \binom{N+n-1}{n}$. We introduce a mapping ϕ from $\mathcal{F}$ to $\mathcal{G}$ defined as follows. let $x = (x_1 x_2 \cdots x_N) \in \mathcal{F}$, where x_i is defined as the number of balls in box i. Clearly, $\sum_{i=1}^{N} x_i = n$. Let $y = \phi(x)$, where y is constructed from x in the following way. Each $x_i = 0$ is left untouched, whereas each $x_i > 0$ is replaced by a sequence of x_i ones and

1 zero, except for the rightmost $x_i > 0$ in the sequence that is replaced by a sequence of x_i ones [1]. Figure 3.1 illustrates this construction for $N = 7$, $n = 6$ and $x = (0, 1, 0, 2, 0, 3, 0)$.

- $x_1 = 0$ is mapped to $y_1 = 0$
- $x_2 = 1$ is replaced by (1,0), i.e. $y_2 = 1$ and $y_3 = 0$
- $x_3 = 0$ is mapped to $y_4 = 0$
- $x_4 = 2$ is replaced by (1,1,0), i.e. $y_5 = y_6 = 1$ and $y_7 = 0$
- $x_5 = 0$ is mapped to $y_8 = 0$
- $x_6 = 3$ is replaced by (1,1,1), i.e. $y_9 = y_{10} = y_{11} = 1$, since x_6 is the rightmost nonzero element in x
- $x_6 = 0$ is mapped to $y_{12} = 0$.

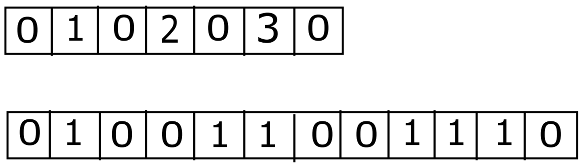


Figure 3.1. The 1-1 mapping

We now determine the length of the sequence y. From the mapping construction, we see that each $x_i = 0$ does not induce new elements for y, whereas $x_i > 0$ induces x_i new elements except for the rightmost $x_i > 0$ that induces $x_i - 1$ new elements. Thus the length of y is equal to $N + \sum_{i=1}^{N} x_i - 1 = N + n - 1$. It easy to see that $\sum_{i=1}^{N+n-1} y_i = n$, and $y_i \in \{0, 1\}$ for all i. Thus $\phi((F)) \subset \mathcal{G}$.

Consider the following input $x = (2, 3)$. It maps to $y = (1, 1, 0, 1, 1, 1)$. This is because 2 is replaced with 110 and 3 with 111. In general, we do not need an extra zero for the last $x_i > 0$ since we know that the last group of k consecutive ones in y corresponds to some $x_i = k$. Thus, given a sequence $y \in \mathcal{G}$, we can decode a unique sequence $x \in \mathcal{F}$ by doing

[1] In essence, the reason we add an extra zero to the x_i ones when mapping $x_i > 0$ in x to y is to signal the end of the bitstream in y due to one nonzero element in x and the beginning of another bitstream in y corresponding to the adjacent nonzero element in x. This results in unique decodability of x sequences from y sequences.

the inverse construction. By reading y from left to right, a 0 becomes a 0, (1,1,0) becomes 2, etc... We conclude that ϕ is a bijection and $\phi((F)) = \mathcal{G}$. Thus, $\#\mathcal{F} = \#\mathcal{G} = \binom{N+n-1}{n}$.

We have shown that there are $\binom{N+n-1}{n}$ possible placements of the n atoms within a frame with N pixels, and since the atom positions are IID, they are all equi-probable. Thus the entropy per atom is given by

$$\frac{1}{n}\log_2\binom{N+n-1}{n}.$$

While in practice the atoms for coding the DFD for a video sequence are not IID, the above expression is still a useful reference for atom position coding. Indeed, achieving a lower average bits per atoms is a good indicator of how well we are able to capture the true statistical structure of the atom positions.

3.2 Frame based position coding

The position coding technique proposed by Neff and Zakhor [11] is frame based and can be described as follows: the atoms are sorted in position order from left to right and top to bottom within the residual image. A differential coding strategy employs three basic codeword tables. The first table P1 is used at the beginning of the frame line to indicate the horizontal distance from the left side of the image to the location of the first atom on the line. For additional atoms on the same line, the second table P2 is used to transmit inter-atom distances. The P2 table also contains an escape code indicating that no additional atoms exist on the current line. The escape code, when used, is always followed by a P3 code, indicating how many lines in the image may be skipped before the next line containing coded atoms. The P3 code is then followed by a P1 code, since the next atom will be the first on some line. No special codeword is needed to indicate the end of the atom field, since the number of coded atoms is transmitted as header information. We construct the Huffman tables P1, P2, and P3 using a set of training sequences. We refer to this position coding scheme as P1P2P3.

3.3 Block-based position coding

In video transmission over noisy channels, it is important for bit-streams to be robust to transmission errors. It is also important for the transmission error to affect a small region, and not to propagate to other areas. In the frame based technique, an error could potentially propagate to the entire frame. This problem can be addressed by a new position coding mechanism that limits the effect of an error to a block of 16×16 pixels and its immediate neighbors [1]. In this approach atoms that are in the same block are coded together. The idea of coding atoms on a block by block basis was initially introduced by Banham and Brailean [2]. The method presented in [1] improves their block-based position coding technique by increasing the coding efficiency without sacrificing error resilience, and works as follows. The atoms within each block are first reordered according to a spiral scanning order shown in Figure 3.2 and then coded differentially. The reason for using a spiral scan is that the motion model is block-based, and thus the model tends to fail at the boundaries of the blocks. This results in a high residual energy around the boundaries of the macro-blocks, and atoms are more likely to lie in that area. A spiral scan will then result in smaller run-lengths. Four different VLC tables are used to code the run-lengths depending on the number of atoms in it. Since the atoms are coded on a block level, the positions of the atoms can be multiplexed with the motion information for each block. For each macro-block a flag indicates the presence of atoms, and the number of atoms in the block is coded using a Huffman code. We refer to this position coding scheme as MB.

In [10], the frame is divided into blocks of equal size, and the size of the block is allowed to change from frame to frame using rate optimization. The size is signaled in the bitstream with a flag, and positions within a block are fixed-length coded. There is a trade-off between the number of bits for position coding within a block and the number of bits to code the flags. Indeed, if the size of the blocks is large, then the number of bits for position coding is high but the number of flags and thus the number of bits to code them is low, and vice versa. Adapting the size of the blocks from one frame to another optimizes that trade-off.

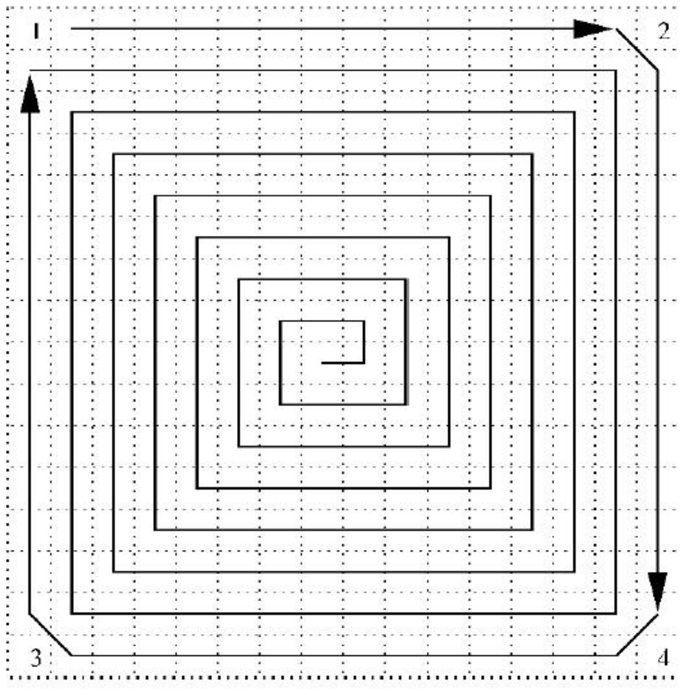


Figure 3.2. Spiral scan

3.4 NumberSplit

The NumberSplit algorithm introduced in [1] is based on the divide-and-conquer idea. First, the total number of atoms T coded on a given residual image is transmitted in the header. Then the image is divided into two halves along a larger dimension, and the number of atoms in the left or top half (depending on how the image was split) is coded. Note that if we assume that each atom falls uniformly and independently of other atoms onto either half, then the number of atoms in the first half is binomially distributed on $\{0, \cdots, T\}$ with $p = 0.5$. Since we know the total number of atoms T, assuming binomial distribution we can construct a Huffman table to encode the number of atoms in the first half. The total number of atoms and the number of atoms in the first half allow the decoder to calculate the number of atoms in the second half. This algorithm is then applied recursively to the halves of the image until there are no more atoms in a given half image or until the size of the half image is one pixel. The Huffman tables used in encoding are built dynamically assuming binomial distribution, depending on the number of atoms to be coded, allowing the NumberSplit method to avoid inefficiencies of fixed tables at a cost of more computation.

In real residual images, atoms are placed at the locations where motion estimation is ineffective. For this reason, atoms are not distributed uniformly and independently on an image. They tend to cluster around the regions of high residual error. One heuristic way to tune the NumberSplit algorithm to the real-life images is to modify the binomial distribution to account for clustering. It is easy to see that if atoms tend to cluster the probability of many atoms being in the same half of the image is higher than if atoms are independent and identically distributed. To account for this, it was proposed in [1] to emphasize the tails of the binomial distribution according to a clustering parameter f. All probabilities of splits that are smaller than a fraction f of the maximum probability in the distribution are set to f $\times$ maximum probability, followed by renormalization of the distribution. It was empirically shown in [1] that the optimal clustering parameter is $f = 0.5$, and that the NumberSplit method spends fewer bits per atom position than the theoretical lower bound for the uniform independent atom distribution described in 3.1. Hence the NumberSplit algorithm exploits the spatial coherence of the atoms positions. It is important to note that by coding all atoms together, NumberSplit achieves good compression efficiency, but also suffers from poor error resilience since a single transmission error may affect all atoms. We refer to this position coding scheme as NS.

Chapter 4

Proposed Position Coding Scheme

4.1 The temporal coherence

The aim of our proposed scheme is to take advantage of both spatial and temporal coherence of the atom positions. We know that the atoms tend to cluster in regions where the motion prediction fails. The atom positions are thus spatially highly non-stationary; as such, we take this into account in our coding scheme. Furthermore, from one frame to the next, these regions of high motion are correlated. If there is a large number of atoms in a given region of the DFD, it is likely that there was also a large number of atoms in the same region in the previous frame. We refer to this as the temporal coherence of the atom positions, and our coding scheme will make use of that property as well.

In the context of a scalable video coder, a scheme for atom position coding that takes advantage of the temporal coherence was introduced in [7]. The principle is to represent the atoms' positions by a quadtree [12]. The temporal coherence of atom positions results in similarities between the quadtrees of two consecutive frames, and thus the XOR of those two quadtrees is coded and transmitted. We refer to this quadtree as the differential quadtree. The entropy of the differential quadtree is lower than the entropy of the original quadtree if they are correlated, and thus there will be coding gain. At the receiver, if the quadtree of the previous frame has been successfully decoded, then we recover the quadtree of the actual frame by doing a XOR with the transmitted differential quadtree. We applied this scheme in our non-scalable video coder, and found its performance to be inferior to the previous schemes P1P2P3, MB and NS since we observed similarities in two consecutive quadtrees only at their upper levels, where the number of nodes is much lower than in lower levels of the tree. At the lowest level corresponding to 1 by 1 pixel, the positions between two consecutive DFD are not correlated any more, and thus the entropy of the nodes at that level in the differential quadtree is not lower than in the original quadtree.

4.2 The proposed algorithm

Our proposed algorithm takes advantage of both temporal and spatial coherence. Since atoms are not distributed identically and independently in the DFD, we propose a block-based scheme that allows us to discriminate regions of the DFD having high and low density of atoms. The number of atoms in each block is coded differentially with respect to the same block in the previous frame, and the NumberSplit algorithm is applied to each block.

Let $N_{b,n}$ be the number of atoms in block b in frame n. Since $N_{b,n}$ and $N_{b,n-1}$ are correlated, we code the "innovation" $N_{b,n} - N_{b,n-1}$ instead of $N_{b,n}$ directly as in the scheme MB described in Section 3.3. Thus, at the decoder, if frame $n-1$ has been correctly decoded we have access to $N_{b,n-1}$, and then we can recover $N_{b,n}$. For the first P frame of the GOP, the number of atoms in each block $N_{b,1}$ is coded directly, and we use an adaptive arithmetic encoder to code the symbols corresponding to $N_{b,n} - N_{b,n-1}$ for the following P frames. We have found empirically that using 16×16 blocks is optimal for the video sequences we tested. See Section 4.3 for the details. In doing so, we take advantage of the temporal coherence.

Once we have transmitted the number of atoms, we need to code the atom positions inside each of the 16×16 blocks. We have empirically found that NumberSplit as applied to the 16×16 blocks gives better results than traditional run-length coding such as Golomb coding or using VLC tables like in MB. See Section 4.4 for the details. We refer to this new position coding scheme as NS+TC.

4.3 The choice of the block size

The block size has to be chosen accordingly. Indeed, if the block size is too small, the correlation between $N_{b,n-1}$ and $N_{b,n}$ will be low and it is more efficient to code $N_{b,n}$ directly instead of $N_{b,n} - N_{b,n-1}$. Conversely, if the block size is too large, there are fewer blocks and we have less gain since the number of symbols to code becomes small, and most of the bits go to the coding of the positions inside the blocks.

On several video sequences, we collected statistics of the number of atoms inside the blocks $N_{b,n}$ and the difference $N_{b,n} - N_{b,n-1}$ for the block sizes 4×4, 8×8, 16×16 and 32×32. We computed probability tables for each of these two symbols and computed their entropy. We observe in Figure 4.1 that for the blocks sizes 4×4 and 8×8, it is more efficient to code $N_{b,n}$ instead of $N_{b,n} - N_{b,n-1}$ since it has a lower entropy. We can say that there is no temporal coherence at that level of granularity. On the other hand, for a block size of 16×16, the entropy of $N_{b,n} - N_{b,n-1}$ is more than 1 bit lower than the entropy of $N_{b,n}$, and since there are 99 16×16 blocks in a QCIF frame, we can expect significant gains. For a 32×32 block, the entropy of $N_{b,n} - N_{b,n-1}$ is 1.5 bit lower than the entropy of $N_{b,n}$. However, there are 4 times less 32×32 blocks than 16×16 blocks, so it is more efficient to use 16×16 blocks.

4.4 Atom position coding within blocks

Once we know the number of atoms inside each block, we need to specify the positions of the atoms inside each block. If there are n atoms in a 16×16 block, then the number of possible placements of these n atoms is $\binom{255+n}{n}$. If the atoms were uniformly and independently distributed within the block, then a coding scheme such as enumerative source coding [3] would have a performance close to the theoretical limit. However, the atoms are not uniformly and independently distributed inside 16×16 blocks. They tend to cluster around the boundaries of the block where the motion estimation is the most inefficient. The block-based technique MB codes n run-lengths where the atoms have been reordered

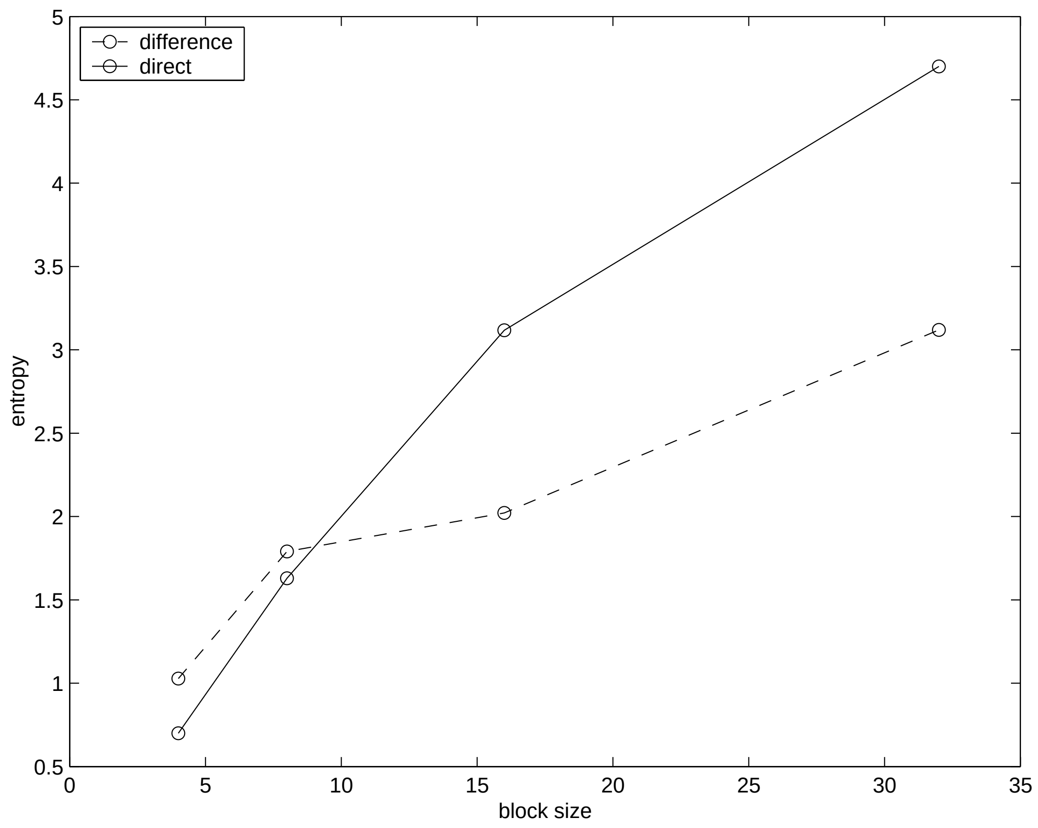


Figure 4.1. Comparison of the entropies for direct coding and differential coding of the number of atoms

according to a spiral scan in order to take advantage of the repartition of the atoms inside the block. That scheme is not very robust to the possibility of having a large distance in the spiral scan between two consecutive atoms. We thus use a scheme that takes into account the spatial coherence within the block based on the NumberSplit algorithm and offers better performance.

In our video coder the chroma channels are subsampled by a factor of 2 in the horizontal and vertical directions. Thus, the UV atoms have their positions subsampled by a factor of two. In existing schemes such as [1] the UV atom positions are doubled, and then all the YUV atoms are coded together. This is inefficient since two bits are lost for every UV atom. Our new scheme first subsamples the Y atom positions in the 16×16 block by two, so all the atoms inside that block are defined on an 8×8 grid. We first send the chroma channel information about the atoms, and we choose a raster scan order for the atoms. We then code the positions using the NumberSplit algorithm as applied to the 8×8 block, and then for each Y atom we add two bits that specify its position in the 16×16 block. The decoder knows which atom position to refine since it knows where the Y atoms are on the 8×8 grid. This is because all the atom information was sent with the atoms ordered according to the raster scan. By doing this, we do not modify the temporal and spatial coherence, and take into account the fact that not all atoms have the same spatial resolution.

Figure 4.2 presents the number of bits per atoms for the coding of the atoms' positions inside the 16 by 16 blocks as a function of the clustering parameter f used for the NumberSplit algorithm. We can see that the optimal choice is $f = 0.2$. However, there is very little sensitivity to the clustering parameter. When the NumberSplit algorithm as applied to the entire frame, the optimal clustering parameter is 0.5. The atom positions are indeed more clustered at the frame level than at the macro-block level where they are almost IID, which explains why we use a clustering parameter with a smaller value.

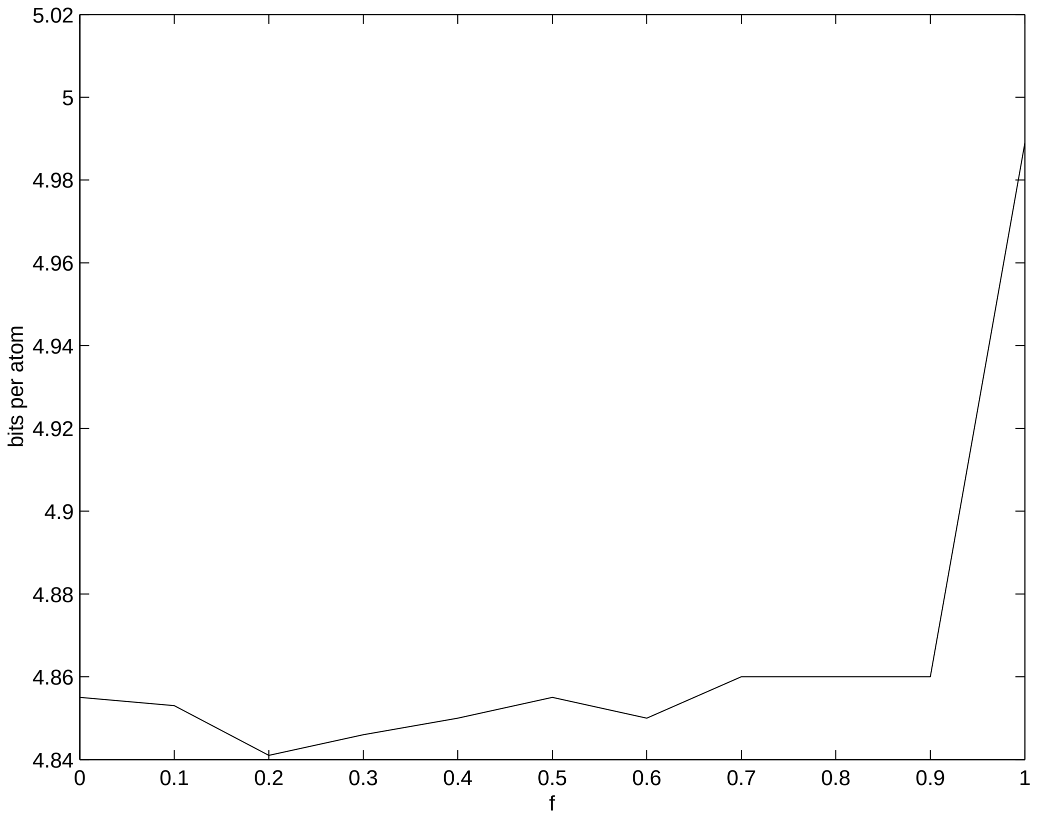


Figure 4.2. Average number of bits per atom for NumberSplit inside 16 by 16 block as a function of the clustering parameter f

Chapter 5

Results

In this Chapter we present results to compare NS+TC with the frame-based coding technique (P1P2P3), the macro-block based position coding (MB), and the NumberSplit algorithm (NS) described earlier. We use the following QCIF training sequences to compute the frequency tables used to build Huffman tables and for the arithmetic encoder: Silent, Mother, Coast and News. The QCIF test sequences are Akiyo, Carphone, Claire, Container, Foreman, Hall, Highway, Salesman and Sean. To approximate constant quality, we encode each frame such that the maximum mean square error (MSE) computed over each macro-block is smaller than a target. We use the values 5, 10 , 20 and 30 as target MSEs. We have found empirically that using values lower than 5 does not offer furhter visual improvements, and the resulting atoms mostly correspond to noise. The frame rate is 10 Hz, and we code one I frame followed by 99 P frames. We compare the effectiveness of each position coding technique by examining the number of position bits and its impact on the overall bit rate as shown in Table 5.2. We also examine the average number of bits per atom for each technique and compare it to the theoretical value, assuming the data is IID. This is shown in Table 5.1.

Our proposed scheme NS+TC outperforms all the other schemes in every case. The average number of bits to code the position of an atom is lower than the theoretical value in the case of IID data. For most of the sequences the reduction exceeds 1 bit per atom.

video sequence	MSE	atoms per frame	IID	P1P2P3	MB	NS	NS+TC
Akiyo	5	151	8.80	8.54	8.77	8.06	7.49
	10	86	9.59	9.54	9.57	9.09	8.43
	20	47	10.43	10.59	10.87	10.09	9.71
Carphone	5	490	7.11	8.08	8.82	7.01	6.51
	10	289	7.87	8.98	8.54	7.83	7.27
	20	165	8.67	9.90	9.10	8.73	8.12
Claire	5	161	8.71	8.47	8.78	7.86	7.04
	10	90	9.53	9.32	9.25	8.76	7.95
	20	49	10.37	10.32	10.58	9.85	9.4
Container	5	384	7.46	7.76	8.21	7.04	6.51
	10	207	8.35	8.50	8.70	7.84	7.18
	20	114	9.19	8.96	9.13	8.57	7.91
Foreman	10	345	7.62	8.78	8.59	7.75	7.30
	20	205	8.36	9.60	8.86	8.52	8.13
	30	148	8.83	10.14	9.20	9.02	8.47
Grandma	5	270	7.97	8.21	8.40	7.65	7.01
	10	141	8.89	9.22	8.98	8.73	7.90
	20	68	9.92	10.32	10.16	9.87	9.18
Hall	5	404	7.39	7.74	8.29	7.27	6.32
	10	185	8.51	8.61	8.76	7.95	7.18
	30	56	10.19	9.25	10.03	9.11	8.96
Highway	5	565	6.90	7.41	8.41	6.99	5.72
	10	217	8.28	8.67	8.54	7.92	6.85
	20	72	9.84	10.35	9.85	9.45	9.07
Salesman	5	300	7.82	8.31	8.43	7.54	7.07
	10	170	8.63	9.16	8.96	8.45	7.87
	20	93	9.48	9.92	9.75	9.34	8.78
Sean	5	238	8.15	8.10	8.59	7.47	7.09
	10	140	8.90	8.92	9.17	8.36	7.91
	20	82	9.66	9.81	9.90	9.23	8.84

Table 5.1. Comparison of the average number of bits per atom position

The sequence Claire has a 1.6 bit reduction at MSE 5 since it is a low motion sequence. Foreman is a high motion video sequence, and the reduction is 0.24 bit per atom at MSE 10, when all other coding schemes are worse than the IID value. This shows that our scheme effectively captures the spatial and temporal coherence. Furthermore, by comparing NS+TC to NS, our scheme uses an average of 0.6 fewer bits with an extremum of 1.6 bits for the sequence Claire. Since NumberSplit exploits the spatial coherence only, this indicates us that taking into account the temporal coherence gives us a significant gain in coding the atoms' positions.

video sequence	MSE	atoms per frame	P1P2P3	MB	NS	NS+TC	bit rate gain NS+TC vs MB	MB position bits %
Akiyo	5	151	32.42	32.76	31.66	30.81	6.0%	40.3%
	10	86	20.40	20.41	20.00	19.44	4.8%	39.3%
	20	47	12.94	13.03	12.69	12.50	4.1%	38.5%
Carphone	5	490	108.94	114.86	103.47	101.14	11.9%	38.8%
	10	289	72.94	72.05	69.37	67.88	5.8%	33.8%
	20	165	49.02	47.81	46.90	45.99	3.8%	30.8%
Claire	5	161	36.27	36.83	35.27	33.97	7.8%	38.2%
	10	90	22.85	22.80	22.34	21.61	5.2%	36.5%
	20	49	14.62	14.72	14.38	14.14	3.9%	34.6%
Container	5	384	73.83	75.61	71.05	69.05	8.7%	40.1%
	10	207	42.22	42.63	40.86	39.50	7.3%	37.6%
	20	114	25.12	25.32	24.68	23.93	5.5%	37.6%
Foreman	10	345	90.11	90.00	86.55	85.12	5.4%	33.1%
	20	205	64.10	62.15	61.28	60.28	3.0%	26.8%
	30	148	52.50	51.23	50.84	50.10	2.2%	26.4%
Grandma	5	270	58.43	58.99	56.88	55.18	6.5%	38.4%
	10	141	32.29	31.94	31.58	30.43	4.7%	39.4%
	20	68	18.27	18.15	17.97	17.49	3.6%	37.5%
Hall	5	404	86.83	89.11	84.94	81.11	9.0%	37.6%
	10	185	41.87	42.18	40.65	39.25	6.9%	33.7%
	30	56	15.01	15.45	14.94	14.85	3.9%	33.5%
Highway	5	565	130.05	135.79	127.66	120.49	11.3%	34.7%
	10	217	54.15	53.88	52.51	50.18	6.9%	34.5%
	20	72	22.28	21.92	21.63	21.36	2.6%	32.1%
Salesman	5	300	64.3	64.82	61.96	60.64	6.4%	38.9%
	10	170	38.65	38.29	37.41	36.45	4.8%	39.3%
	20	93	24.25	24.05	23.69	23.16	3.7%	37.2%
Sean	5	238	49.87	51.10	48.18	47.32	7.4%	39.4%
	10	140	30.82	31.20	29.95	29.33	6.0%	40.4%
	20	82	20.16	20.15	19.68	19.31	4.2%	38.7%

Table 5.2. Comparison of the overall bit rate in kbps

Table 5.2 compares the overall bit rates for the various coding techniques, and the two last columns of the table show the reduction in bit rate when using NS+TC instead of MB, and the percentage of bits used to code the atom positions using MB. Since the percentage of bits used to code the atom positions can exceed 40% of the overall bit rate, there is a significant reduction in the overall bit rate when using a more efficient position coding technique such as NS+TC. We observe that the performance improves with the number of atoms to code. Indeed, at higher bit rates, the percentage of bits allocated to atom coding becomes higher, resulting in more atoms to be coded, and hence larger gain by using a more efficient position coding method. We observe this trend for each of the encoded sequences in Table 5.2. The lower the MSE target, the larger the reduction in bit rate. The average reductions for MSE 5, 10 and 20 are 8.3%, 5.8% and 3.8% respectively. The largest gain is obtained with the sequence Highway coded with MSE 5 where the reduction in bit rate is 11.3% when using NS+TC instead of MB. The sequence Foreman has the lowest reduction in bit rate of all sequences, due to its high motion nature. The percentage of the bit rate going to the motion information is higher than for the other sequences even at high bit rate, and thus the gains for the coding of the DFD have less influence on the overall bit rate.

Figure **??** compares the number of bits for position per frame for the techniques MB and NS+TC for four sequences. We can observe that for each of the 99 P frames, the performance of NS+TC is better. The scheme can thus be applied with success to groups of pictures having patterns such as IBBP, IBBBP and so on since we have a net increase in performance even for the first P frames in Figure 5.1, 5.2, 5.3 and 5.4.

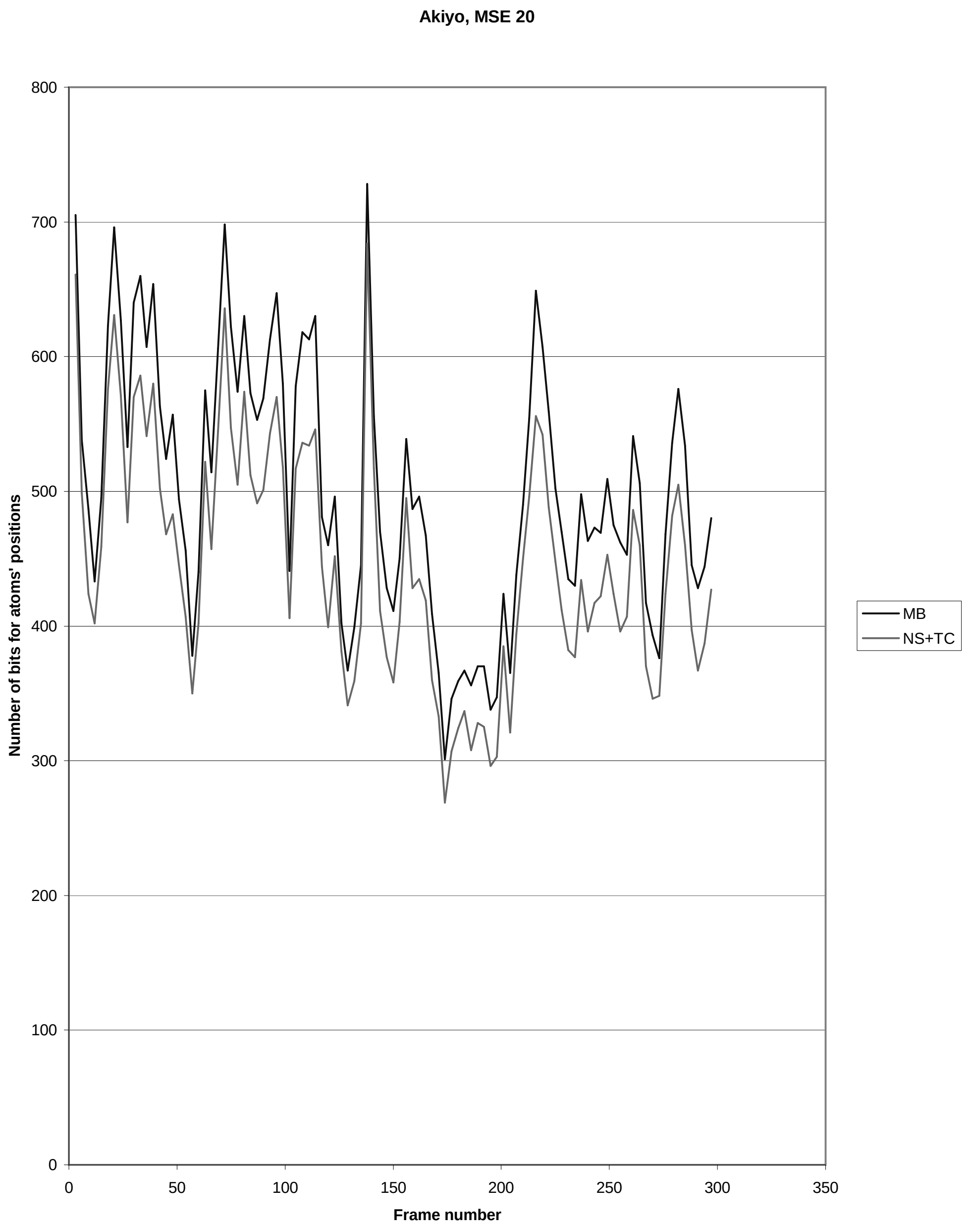


Figure 5.1. Comparison of the number of bits for atom position using MB and NS+TC for the video sequence Akiyo

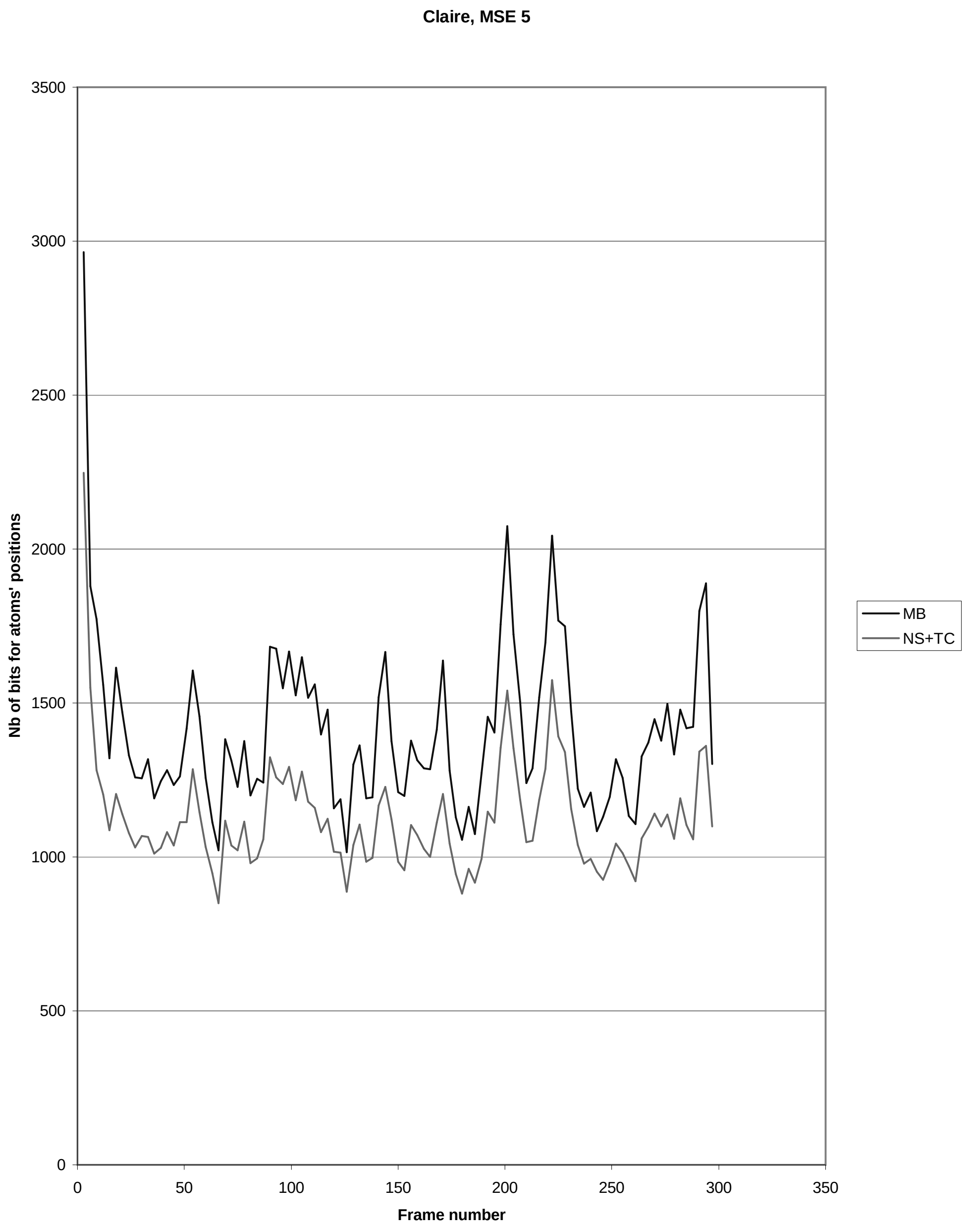


Figure 5.2. Comparison of the number of bits for atom position using MB and NS+TC for the video sequence Claire

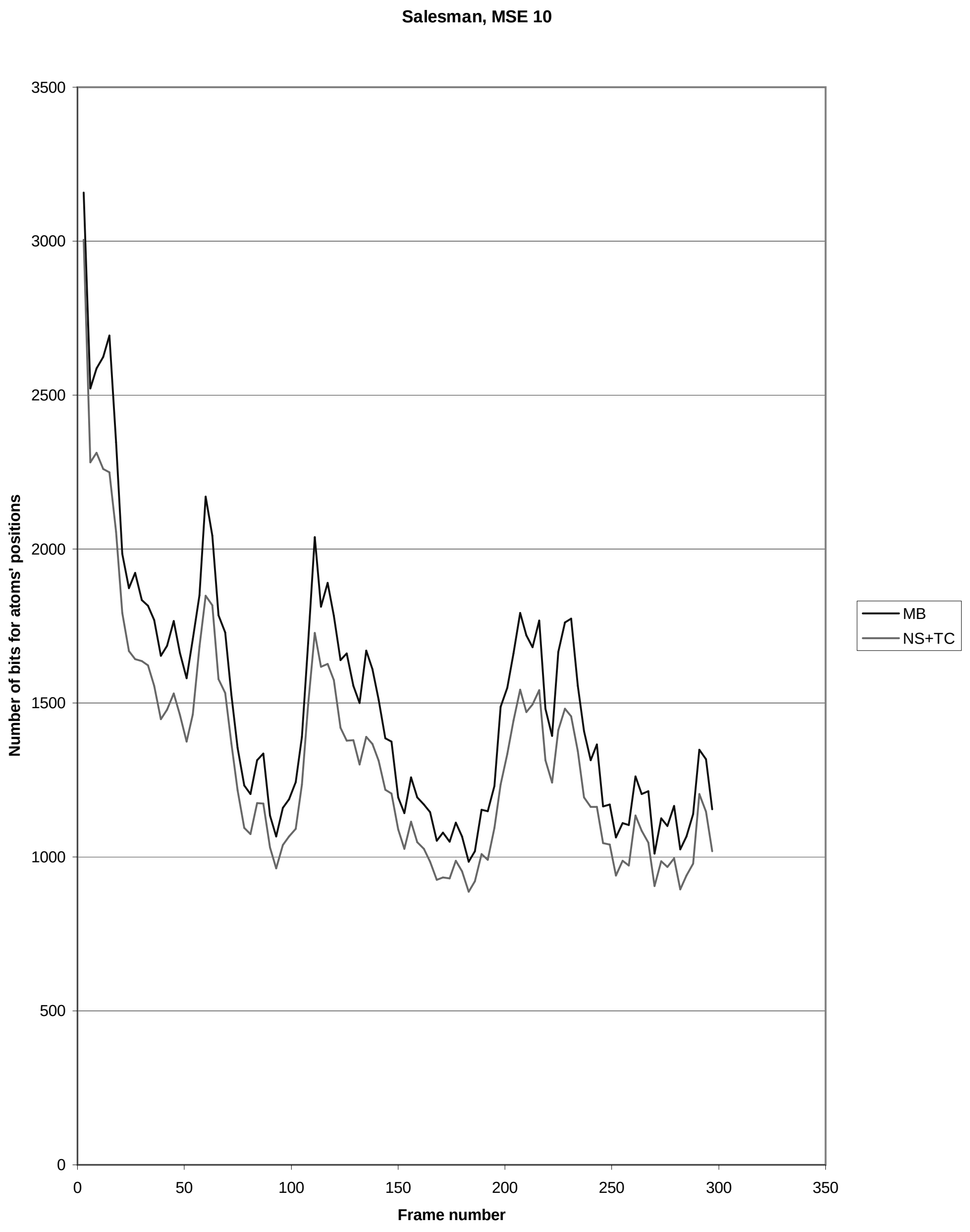


Figure 5.3. Comparison of the number of bits for atom position using MB and NS+TC for the video sequence Salesman

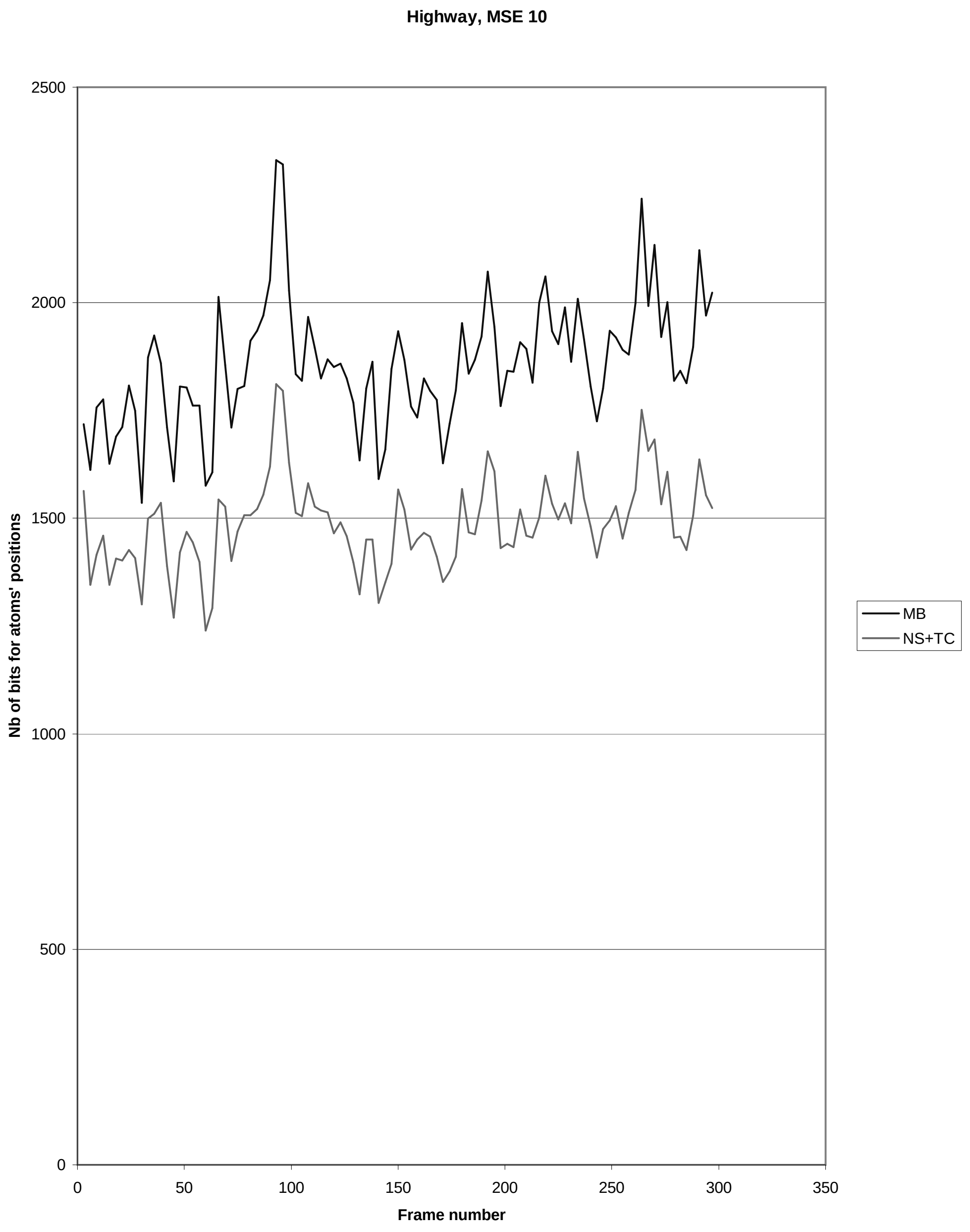


Figure 5.4. Comparison of the number of bits for atom position using MB and NS+TC for the video sequence Highway

Chapter 6

Conclusion

In this thesis we have proposed a new scheme to code the positions of the atoms in a Matching Pursuit based video encoder. If the atom positions were independent and identically distributed within the frame (IID), we could apply a traditional coding technique such as enumerative coding that performs close to the entropy for IID data. Since the atom positions are not IID, we do not know their true statistics and thus their entropy. We have empirically discovered and exploited some characteristics of their statistical structure, namely the spatial and temporal coherence in order to derive a new algorithm for position coding. This algorithm allows us to compress the atom positions more efficiently, and to obtain compression rates superior to the theoretical value if the atom positions were IID. The proposed approach results up to 11% lower bit rate as compared to existing schemes.

References

[1] O. Al-Shaykh, E. Miloslavsky, T. Nomura, R. Neff and A. Zakhor, "Video Compression using Matching Pursuits," *IEEE Trans. on Circuits and Systems for Video Tech.*, Vol.9, no.1, pp. 123-143, Feb.1999.

[2] M. Banham and J. Brailean, "A selective update approach to matching pursuits video coding," *IEEE Trans. on Circuits and Systems for Video Tech.*, Vol.7, no.1, pp. 119-129, Feb.1997.

[3] T.M. Cover, "Enumerative Source Coding," *IEEE Trans. on Information Theory*, IT-19(1), pp.73-77, 1973.

[4] J.H. Friedman and W. Stuetzle, "Projection Pursuit Regression," *Journal of the American Statistical Association*, vol.76, pp.817-823, 1981.

[5] P. Frossard, P. Vandergheynst, R.M. Figueras i Ventura and M. Kunt, "A Posteriori Quantization of Progressive Matching Pursuit Streams," *IEEE Trans. on Signal Processing*, vol.52, no.2, Feb.2004, pp.525-535.

[6] S.W. Golomb, "Run-length Encodings," *IEEE Trans. on Information Theory*, IT-12(3), pp.399-401, 1966.

[7] J-L. Lin, W-L. Hwang and S-C. Pei, "SNR Scalability Based on Bitplane Coding of Matching Pursuit Atom s at Low Bit Rates: Fine-Grained and Two-Layer," *IEEE Trans. on Circuits and Systems for Video Tech.*, vol.15, no.1, Jan.2005.

[8] S. Mallat and Z. Zhang, "Matching Pursuits with time-frequency dictionaries," *IEEE Trans. on Signal Processing*, vol.41, no.12, pp.3397-3415, Dec.1993.

[9] D.M. Monro and W. Poh, "Improved Coding of Atoms in Matching Pursuits, " *in ICIP 2003*, Barcelona, 2003.

[10] F. Moschetti, K. Sugimoto, S. Kato and M. Etoh, "Bidimensional Dictionary and Coding Scheme for a Very Low Bitrate Matching Pursuit Video Coder," *in ICIP 2004*, Singapore, 2004.

[11] R. Neff and A. Zakhor, "Very Low Bit-Rate Video Coding based on Matching Pursuits," *IEEE Trans. on Circuits and Systems for Video Tech.*, vol.7, no.1, pp. 158-171, Feb.1997.

[12] E. Shusterman and M. Feder, "Image compression via improved quadtree decomposition algorithms," *IEEE Trans. Image Process.*, vol.3, no. 2, pp.207-215, Mar.1994.