MULTIPLE BLOCKS UPDATE ALGORITHM FOR MATCHING PURSUIT VIDEO CODING

Jian-Liang Lin[†*]*, Wen-Liang Hwang*[†]*, and Soo-Chang Pei*[*]

[*]Institute of Communication Engineering, National Taiwan University, Taiwan, R.O.C.
[†]Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C.

ABSTRACT

Matching pursuit (MP) video coding has been demonstrated to attain a better coding performance than DCT-based video coding in terms of PSNR and perceptual quality at very low bit rates. However, because of its massive computational complexity, the MP algorithm is usually only approximated. By approximating a residual in a subspace, we propose a multiple blocks search and update algorithm in MP video coding to achieve a faster and better MP approximation. In this paper, we evaluate the performance and compare it to the traditional one-block search algorithms. The experimental results show that our proposed algorithm can significantly improve the coding performance and encoding time.

1. INTRODUCTION

Matching pursuit, which is a frame-based algorithm, is a promising method for low bit-rate video coding [5]An MP-based codec yields a better PSNR and perceptual quality than a transform-based codec and its decoder is simpler . However, it cannot be used in applications that require real time bidirectional communications because the encoder consumes a massive amount of computational time. An MP encoder does not obtain all the coefficients in one step, but iteratively finds the frame coefficient that has the largest absolute inner product value between a residual and all the bases. The inner product value and the base from which the value is obtained are called an atom. Many approaches have been proposed to simplify the complex encoding stage. One approach approximates the codewords of a dictionary with a linear combination of simpler codewords so that the computation becomes easier [7, 2, 6, 8].

The most popular approach for finding an atom is proposed by Neff and Zakhor [5], in which a residual frame is divided into blocks, and an atom is found within the block with the largest amount of energy at each iteration. This approach is modified in [1], in which an energy weight is given to a block so that the greater the number of atoms chosen from the block, the smaller the energy weight of the block will be. Therefore, the block is less likely to be chosen in later iterations. The energy weight approach reduces the likelihood that most of the atoms will be selected from a few blocks, and improves the PSNR performance over that of Neff and Zakhor's algorithm. The above algorithms find an atom from the largest (weighted) energy block, we therefore call them one-block algorithms.

The one-block algorithm is simple and efficient at the cost of sacrificing coding performance. Although coding performance can be improved by finding an atom from more than one block, there is still the issue of the massive number of inner products between a residual and the bases in these blocks. To solve this problem, we approximate a residual in a subspace, spanned by a small number of bases within a few blocks. The bases and the blocks are selected according to the content of the residual, while the coding performance and efficiency are determined by the number of bases and the number of blocks. Our simulations show that our PSNR and efficiency are better than in a one-block algorithm at low bit-rates of various sequences.

2. MP UPDATE ALGORITHM AND ATOM EXTRACTION

Matching pursuit is a frame-based algorithm that represents a signal by a succession of greedy steps [4]. For each iteration, the signal is projected to a base that approximates the signal most efficiently. Let the over-complete image bases be $\{g_\gamma(\mathbf{x})\}$, where γ is the index. The matching pursuit algorithm decomposes an image into a linear expansion of the bases as follows.

The image $f(\mathbf{x})$ is first decomposed into

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

where $g_{\gamma_0}(\mathbf{x}) = \arg_{g_\gamma(\mathbf{x})} \max\{| < f(\mathbf{x}), g_\gamma(\mathbf{x}) > |\}$ and $Rf(\mathbf{x})$ is the residual image after approximating $f(\mathbf{x})$ in the direction of $g_{\gamma_0}(\mathbf{x})$. The $g_{\gamma_0}(\mathbf{x})$ and the inner product value $< f(\mathbf{x}), g_{\gamma_0}(\mathbf{x}) >$ are called an atom. The matching pursuit algorithm then decomposes the residual image $Rf(\mathbf{x})$ by projecting it onto the bases, as was done for $f(\mathbf{x})$. Instead of recalculating the inner products at each iteration, Mallat and Zhang [4] provide the MP update algorithm. At the kth iteration, let:

$$g_{\gamma_k} = \arg\max_{\gamma\in\Gamma} | < R^k f, g_\gamma > |$$

0-7803-9134-9/05/$20.00 ©2005 IEEE

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

$$R^{k+1}f = R^k f - < R^k f, g_{\gamma_k} > g_{\gamma_k}.$$

The inner products between $R^{k+1}f$ and the bases $\{g_\gamma\}$ can be represented by

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

Because $< R^k f, g_\gamma >$ and $< R^k f, g_{\gamma_k} >$ have been calculated in the previous iteration, and if $< g_{\gamma_k}, g_\gamma >$ is precalculated, this update needs only one addition and multiplication. Unfortunately, this update algorithm needs a huge amount of space to store all non-zero $< g_{\gamma_k}, g_\gamma >$ in an image and is only useful in an one-dimensional signal decomposition.

For image decomposition by MP, because the number of bases is huge, the complexity of applying the MP to the entire image is too high. To overcome this problem, the proposed approach in [5, 1] divides a residual into blocks, and at each iteration, the MP is applied only to the block with the largest energy. This approach is both simple and efficient and has, therefore, been implemented in many MP-based video codecs.

3. MULTIPLE BLOCKS APPROXIMATION

The approach in [5, 1] assumes that the probability of the current largest energy block containing the *maximum atom* is high. This assumption can be further developed with the correlation between a block containing the maximum atom and the energy of the block. Thus, the energy of a block can be used to determine whether a block should be included in the procedure to find atoms.

3.1. Blocks Selection

Let $\mathbb{B}$ be the set of blocks in which to search for atoms. Before we propose our multiple block selection algorithm, we present the optimal set of blocks for atom selection and show the difficulties in obtaining the optimal set in practice. For a block b, let $P_0(b)$ be the probability that the maximum atom is not within b, and let $P_1(b)$ be the probability that the block b contains the maximum atom. The miss probability P_M means that the block, containing the maximum atom, is excluded from finding the atom:

$$P_M(\mathbb{B}) = \sum_{b \notin \mathbb{B}} P_1(b). \quad (2)$$

The false alarm probability P_F means that an atom is found in a block that does not contain the maximum atom:

$$P_F(\mathbb{B}) = \sum_{b \in \mathbb{B}} P_0(b). \quad (3)$$

We define the average performance loss of selecting a non-maximum atom incurred by $\mathbb{B}$ as:

$$R(\mathbb{B}) = P_F(\mathbb{B})C_F + P_M(\mathbb{B})C_M,$$

where the non-negative numbers C_F and C_M are the respective average conditional performance losses when a false alarm or a miss occur. From Equations 3 and 2, we can derive that

$$\begin{aligned} R(\mathbb{B}) &= \sum_{b \in \mathbb{B}} P_0(b)C_F + \sum_{b \notin \mathbb{B}} P_1(b)C_M \\ &= \sum_{b \in \mathbb{B}} P_0(b)C_F + \sum_{b \in \mathbb{B}} (1 - P_1(b))C_M \\ &= \sum_{b \in \mathbb{B}} (P_0(b)C_F - P_1(b)C_M) + \sum_{b \in \mathbb{B}} C_M \\ &= \sum_{b \in \mathbb{B}} (P_0(b)C_F - P_1(b)C_M) + C_M|\mathbb{B}|. \quad (4) \end{aligned}$$

Let the optimal set $\mathbb{B}^*$ be the block set that minimizes the above equation. Let $\tilde{\mathbb{B}}$ be the set of blocks satisfying

$$\tilde{\mathbb{B}} = \{b | P_0(b)C_F - P_1(b)C_M \leq 0\}. \quad (5)$$

Equation 5 can be rewritten as

$$\tilde{\mathbb{B}} = \{b | P_1(b) \geq \tau P_0(b)\}, \quad (6)$$

where $\tau = \frac{C_F}{C_M}$. The likelihood of block b can be defined as

$$L(b) = \frac{P_1(b)}{P_0(b)},$$

and Equation 6 becomes

$$\tilde{\mathbb{B}} = \{b | L(b) \geq \tau\}. \quad (7)$$

Because any block in $\mathbb{B}^*$ must be in $\tilde{\mathbb{B}}$, we have

$$\mathbb{B}^* \subseteq \tilde{\mathbb{B}}. \quad (8)$$

The optimal block set $\mathbb{B}^*$ is too difficult to determine. Thus, we propose the following ad-hoc procedure to construct the block set $\mathbb{B}$. This procedure is simple and has proven to be effective in our simulations.

The correlation between the block containing the maximum atoms and the blocks with larger amounts of energy is high. Therefore, at each iteration, we include the blocks of relatively large energy levels into $\mathbb{B}$. We normalize the energy of the blocks at each iteration so that the block with the largest energy becomes 1. A block b is assigned to $\mathbb{B}$ according to its normalized energy $||\tilde{b}||^2$:

$$b \in \mathbb{B} \quad \text{if} \quad ||\tilde{b}||^2 \geq \eta,$$

where $0 \leq \eta \leq 1$ is a threshold. An atom is then chosen from the blocks in $\mathbb{B}$.

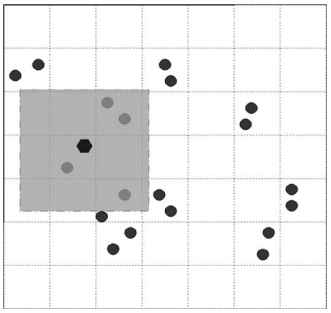


Fig. 1. After the base corresponding to the black hexagon at the center of the gray area is selected, the inner products with the bases covered in the gray area are updated. The black dots are bases.

3.2. Block Content Approximation

Our approach is to approximate the content of a block in $\mathbb{B}$ in a subspace spanned by a few MP bases. Let a block be $|s|^2$ pixels, and $|D|$ be the size of the dictionary D. The bases in a block are $|s|^2|D|$. For computational efficiency, we reduce the number of bases in a block to L. Figure 1 illustrates an example in which $L = 2$ and $|\mathbb{B}| = 10$.

Our algorithm that multiple blocks and the MP update is described below.

Multiple Blocks Update Algorithm

1. **Initialization (k=0)**: The residual f is first divided into blocks.If the normalized energy $||\tilde{b}||^2$ of the block b is larger than a threshold $0 < \eta < 1$, i.e.

$$||\tilde{b}||^2 \geq \eta, \tag{9}$$

   we assign the block to $\mathbb{B}$ and calculate the inner products between the residual and the bases of the block. We then record the L bases $\{g^b_{\gamma_l}, l = 1, \cdots, L\}$ giving the largest absolute inner products and assign them to B_L.

2. **Atom Extraction and Update of the Inner Products (at k-th iteration)**: Let $g_{\gamma_{k_{max}}}$ be the basis that gives the largest absolute inner product value. We then update the non-zero inner products according to $g_{\gamma_{k_{max}}}$ and Equation 1.

3. **Update of Block Set** $\mathbb{B}$: When an atom be extracted from a block, the energy of some blocks may change. For a block that is not in $\mathbb{B}$, if it's normalized energy is larger than η, we include it in $\mathbb{B}$. We then calculate the inner products between $R^{k+1}f$ and the bases within this block, and record the L best bases, as we do in Step 1.

4. **Next iteration**: $k = k + 1$. If $k < n$, then go to Step 2.

In the first phase, for each block in $\mathbb{B}$ we compute the inner products of the block with all the bases, and from them we select the L bases producing the L largest absolute values. Let B_L be the union of the bases of all blocks. Because we assume that the approximating residual is in the subspace spanned by B_L, in the second phase we apply MP to a residual with the restriction on the bases in B_L. Each block in the first phase obtains $|s|^2|D|$ inner product values. It takes a complexity of $|s|^2|D|$ to obtain the L largest absolute inner products from them.

The first phase takes $|s|^2|D||\mathbb{B}|$ inner products, and the second phase takes at most $|\mathbb{B}|L$ inner products to obtain an atom. If the non-zero inner products between bases in B_L are at most $m \leq |\mathbb{B}|L$ and if a residual takes n iterations on average, we perform

$$|s|^2|D||\mathbb{B}| + (n-1)m$$

inner products to approximate a residual. To obtain a better efficiency than the one-block algorithm, we require that, after n iterations,

$$|s|^2|D||\mathbb{B}| + (n-1)m < |s|^2|D|n. \tag{10}$$

The term $n|s|^2|D|$ is the total number of inner products that find n atoms by using the one-block algorithm. Because the number of bases of a block is much smaller than $|s|^2|D|$, we can use the MP update algorithm to update the inner products at each iteration.

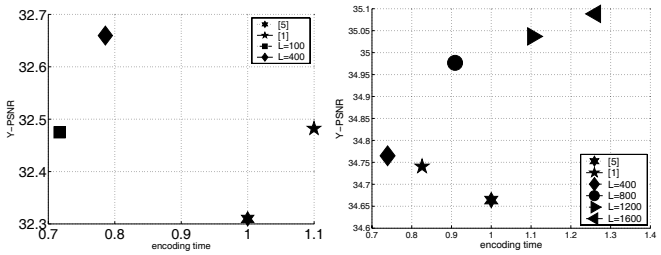
4. PERFORMANCE EVALUATIONS AND COMPARISONS

We evaluated the performance of our algorithm and compared it with the popular algorithms in [5, 1]. An MP atom contains a base and an inner product value. The index of a codeword is encoded by an adaptive arithmetic code. The inner product value is encoded by a bit-plane based approach and the position of a base is located by a quadtree and quadtree representation [3]. Other different MP atom encoding methods can be used, but they change the average number of bits to encode an atom r_a. The first frame of a video sequence is an intra-frame (I-frame), encoded by DCT, and all other frames are inter-frames (P-frames), encoded by MP.

Table 1 shows the computing time taken to encode the Akiyo sequence by various searching algorithms. In our testing platform, the CPU speed is 2.4 GHz per second. Our algorithm has three components: computing the inner products between a residual and bases (T_{ip}); sorting the largest L bases for each block (T_{sort}); and updating inner products for atom candidates (T_{up}). The computing time of each component is also shown in the table. Our algorithm computes inner products at the first iteration, and updates the inner products at the following iterations. Because updating inner products is relatively faster than computing inner

Table 1. Elapsing time (sec) for encoding the Akiyo sequence by different methods. Our algorithms are in the third and the fourth columns with L=100 and $L = 400$, respectively.

Algorithm	[5]	[1]	L=100	L=400
Total time	321.20	287.44	183.89	205.33
T_{ip}	185.94	160.65	56.67	67.34
T_{sort}	9.23	7.76	2.85	3.51
T_{up}	0	0	3.45	15.43



(a) 24 Kbps (b) 44 Kbps

Fig. 2. The plot of average computing time versus PSNR of various.

products, the overall time of our algorithm is constrained by T_{ip}.

Figures 2 plots of shows the average time versus the PSNR of various sequences at different bit-rates. Our sequences include the following slow motion sequences: Akiyo, Sean, Miss America, Container, Mother and Daughter, Salesman, and the fast motion sequences: Carphone and Foreman. The PSNR performance of our method increases as L increases. This implies that using more bases to approximate a residual yields a better PSNR, but the overall computing time increases. Figure 2 shows that overall computing time increases linearly as a function of L. The data in Figure 2 (a) shows that our approach with $L = 400$ gives the best average performance in terms of time and PSNR of all methods at 24 Kbps. The PSNR gains with this parameter ($L = 400$) over that of the Neff and Zakhor's one-block algorithm is on average of 0.4 dB. We normalize the computing time of the Neff and Zakhor's algorithm to 1, so that the comparison will not be affected by the speed of the CPU. The computing time of our algorithm with $L = 400$ is a factor of $0.7 - 0.8$ of that of the one-block algorithms. Figure 2 (b) shows that for L between 800 and 1200, our method has a PSNR gain $0.4 - 0.5$ dB over that of Neff and Zakhor's one-block algorithm at 44 Kbps.

5. CONCLUSION

In contrast to the traditional approach, in which an atom is chosen from the block with the largest (weighted) energy, we approximate a residual in a subspace spanned by a few MP bases. From this approximation, we obtain a new MP atom finding algorithm that uses multiple blocks for atom searching, and uses the MP update algorithm to update inner product values. The simulations show that our proposed algorithm outperforms one-block algorithms, in terms of PSNR, as well as computing time. The performance of our method depends on two parameters, namely: the number of blocks and the number of bases in each block. Adaptation of the parameters for different video sequences to achieve the best performance is an issue worthy of further study.

6. REFERENCES

[1] O. Al-Shaykh, E. Miloslavsky, T. Nomura, R. Neff, and A. Zakhor, "Video compression using matching pursuits", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 1, pp. 123–143, Feb. 1999.

[2] P. Czerepiński, C. Davies, N. Canagarajah, and D. Bull "Matching pursuits video coding: dictionaries and fast implentation", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 7, pp. 1103–1115, Oct. 2000.

[3] J.L. Lin, W.L. Hwang, and S.C. Pei, "SNR scalability based on bitplane coding of matching pursuit atoms at low bit rates: fine-grained and two-layer", to appear in *IEEE Trans. Circuits Syst. Video Technol.*

[4] G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries", *IEEE Trans. Signal Processing*,Vol. 41, pp. 3397–3415, December 1993.

[5] R. Neff and A. Zakhor, "Very low bit-rate video coding based on matching pursuits", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 158–171, Feb. 1997.

[6] R. Neff and A. Zakhor, "Matching pursuit video coding–part I: dictioanry approximation", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, pp. 13–26, Jan. 2002.

[7] D.W. Redmill, D.R. Bull, and P. Czerepinki, "Video coding using a fast non-separable matching pursuits algorithm", *Proc. IEEE Int. Conf. Image Processing.*, pp. 769–773, 1998.

[8] C. De Vleeschouwer and B. Macq, "Subband dictionaries for low-cost matching pursuits of video residues", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, No. 7, pp. 984–993, Oct 1999.