DICTIONARY APPROXIMATION FOR MATCHING PURSUIT VIDEO CODING

Ralph Neff and Avideh Zakhor

Department of Electrical Engineering
University of California, Berkeley

ABSTRACT

Previously, we demonstrated an efficient video codec based on overcomplete signal decomposition using matching pursuits. Dictionary design is an important issue for this system, and others have shown alternate dictionaries which lead to either coding efficiency improvements or reduced encoder complexity. In this work, we introduce for the first time a design methodology which incorporates both coding efficiency and complexity in a systematic way. The key to our new method is an algorithm which takes an arbitrary 2-D dictionary and generates approximations of the dictionary which have fast 2-stage implementations. By varying the quality of the approximation, we can explore a systematic tradeoff between the coding efficiency and complexity of the matching pursuit video encoder. As a practical result, we show cases where complexity is reduced by a factor of 500 to 1000 in exchange for small coding efficiency losses of around 0.1 dB PSNR.

1. INTRODUCTION

Most video codecs in use today are based on a hybrid motion-compensated transform structure. The current frame is predicted using motion compensation, and the prediction error residual is coded using a transform, typically the discrete cosine transform (DCT). In previous work, we demonstrated that improved coding efficiency may be achieved by replacing the DCT with an overcomplete transform [1]. Both PSNR and visual improvement over standard DCT-based video coders has been shown [2].

Dictionary design is an important topic, affecting both the coding efficiency and the complexity of the matching pursuit encoder. Complexity reduction is desirable, since matching pursuit is based on a potentially expensive local inner product search. For this reason, published matching pursuit dictionaries [1][3][4][5][6] have typically been based on a complexity-restricted design. A computable structure is initially assumed, for example 2-D separable functions. The dictionary is then designed within this restricted set of possibilities. This design paradigm is shown in Figure 1(a).

One problem with this design method is that the initial restrictions reduce the design space. Even if some "optimal" design procedure is used within the complexity-restricted set, a better design may exist outside of the set. As an example, restriction to separable 2-D functions is typical [1][3][5][6]. However, diagonal edges and curves are excluded from the resulting dictionary, which is then unable to efficiently code such structures. By lifting the initial restrictions and allowing a dictionary of arbitrary 2-D functions, coding efficiency may be improved. However, the resulting system may be complex, since it has no computationally efficient way to compute the inner products for matching pursuit.

In this work, we start with an arbitrary 2-D dictionary and develop a framework for generating approximations of that dictionary which have fast implementations. This enables a new design paradigm, as illustrated in Figure 1(b). An initial dictionary is chosen without complexity restrictions, using any design or optimization technique. This dictionary is shown by the "$\circ$" symbol in the figure. Our algorithm is then employed to generate a second dictionary, as shown by "$\bullet$". The second dictionary approximates the first, and is thus capable of similar coding efficiency. However, the second dictionary is built to accommodate an efficient 2-stage implementation introduced in [4], and in this way complexity is substantially reduced. A second advantage is that the approximation quality may be reduced in exchange for complexity gains. This results in a systematic tradeoff between the coding efficiency and complexity of the encoder.

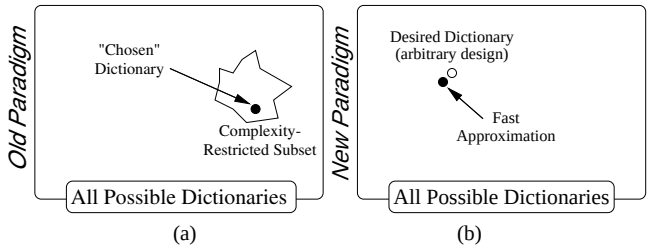


Fig. 1. Basis design paradigms for matching pursuit. (a) Old paradigm based on initial computational restrictions. (b) New paradigm enabled by dictionary approximation.

The paper is organized as follows. Section 2 reviews matching pursuit residual encoding and introduces a fast 2-stage filtering structure on which our approximation algorithm is built. Dictionary approximation is developed in Section 3. Coding results are presented in Section 4, and conclusions are summarized in Section 5.

2. MATCHING PURSUIT VIDEO CODING

As in previous papers, we will use matching pursuit to encode the motion residual signals in a hybrid video coding system. Matching pursuit decomposes motion residual $\vec{f}$ into a weighted combination of basis functions called *atoms* over multiple stages. At each stage, a single basis function $\vec{t}_\gamma$ is chosen from an overcomplete dictionary T in order to best represent the remaining signal energy in $\vec{f}$. An energy pre-search first identifies a local area in which to place the atom. The exact basis and weighting value p_i are then chosen by an exhaustive local inner product search. That is, each $\vec{t}_\gamma \in T$ is centered at each location (x, y) in the local region and the location and basis are chosen which maximize modulus p_i:

$$p_i = \left\langle \vec{f}_i, \vec{t}_{(\gamma,x,y)} \right\rangle$$

Here $\vec{f}_i$ is the remaining energy after i stages, with $\vec{f}_0 = \vec{f}$. After each atom is found, a modulus quantizer $Q(\cdot)$ is applied and the quantized atom is subtracted from a working copy of the residual image:

$$\vec{f}_i = \vec{f}_{i-1} - Q(p_i)\,\vec{t}_{(\gamma,x,y)_i}$$

This is called the "atom update" step. After the current atom has been subtracted, the remaining energy is then passed on to the next atom search stage. After N atoms have been found, the motion residual signal may be approximated as:

This work was supported by NSF grant CCR-9903368.

$$\vec{f} \approx \sum_{i=1}^{N} Q(p_i)\, \vec{t}_{(\gamma,x,y)_i}$$

The decoder reconstructs the residual by decoding the atom parameters and summing the atoms as above. This result is added to the motion prediction image to form the reconstructed frame.

The complexity of the encoder is mostly due to the exhaustive local inner product search. One way to reduce this complexity is to base the dictionary on a computable structure such as separable 2-D functions [1][6]. Further complexity reductions may be achieved if the separable 2-D dictionary can be realized as a successive application of short-kernal filters [3][5]. A more elaborate dictionary implementation scheme which accomodates nonseparable functions was introduced by Redmill, et.al. [4]. We will now review this work in detail.

Suppose each function in a target dictionary T is constructed as a weighted summation of functions from a simpler elementary dictionary S. It then becomes possible to compute the desired target inner products as weighted summations of the elementary inner products. For example, if $\vec{t}_1 = c_1\vec{s}_1 + c_2\vec{s}_2$, then

$$\left\langle \vec{f}_i, \vec{t}_1 \right\rangle = c_1 \left\langle \vec{f}_i, \vec{s}_1 \right\rangle + c_2 \left\langle \vec{f}_i, \vec{s}_2 \right\rangle \tag{1}$$

More generally, the inner products needed for matching pursuit decomposition are computed using the 2-stage filtering structure shown in Figure 2. For each atom stage i, the inner products between signal $\vec{f}_i$ and the required (x, y) positional translates of the elementary functions $\vec{s}_a \in S$ are first computed. This produces A elementary inner product buffers $u_a(x, y)$. Weighted combinations of these elementary inner products are then used to form B target inner product buffers $v_b(x, y)$. These represent the inner products between $\vec{f}_i$ and the positional translates of each target dictionary function $\vec{t}_b$. The largest inner product among all the $v_b(x, y)$ is then used to select the atom to code for stage i.

The efficiency of this two-stage structure depends on additional details which we must omit due to space limitations. For these, the interested reader is referred to [4]. To motivate our current method, we rely on two main conclusions from that work:

1. To use the efficient structure of Figure 2, T must be designed so that each $t_b \in T$ is a weighted combination of functions from a simpler dictionary S.
2. Further complexity reduction is achieved using a *dependency* technique. Here dictionary T is an ordered set, $\{t_b, b = 1 \ldots B\}$, and each t_b is constructed as a weighted combination of earlier targets $t_{b'}, b' < b$ and functions from S.

The advantage of dependency is that values in $v_{b'}(x, y)$ with $b' < b$ are recycled for use in the later computation of $v_b(x, y)$. For example, Redmill's dictionary builds each target inner product $v_b(x, y)$ from one previously computed target inner product $v_{b'}(x, y)$ and one or two elementary inner products. The target inner products are thus built progressively at low computational cost. Furthermore, nonseparable target functions may be efficiently built in this way even if the elementary dictionary is separable.

3. DICTIONARY APPROXIMATION

Consider an arbitrary initial dictionary $T_0 = \{\vec{t}_{(0,b)}, b = 1 \ldots B\}$ which has been designed without encoder complexity constraints. Suppose T_0 has in some way been optimized for coding efficiency.

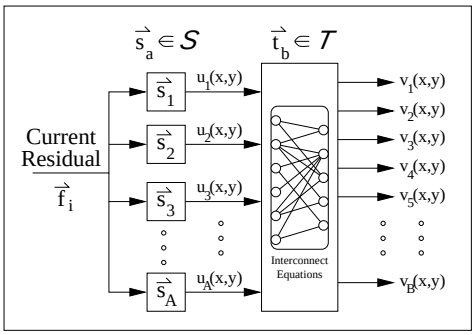


Fig. 2. A two-stage filtering structure used by Redmill, et.al., for efficient implementation of a nonseparable dictionary.

We wish to generate approximations of T_0 which have similar coding efficiency, but have reduced complexity encoder implementations. We now show a method for generating a series of dictionaries $T_q = \{\vec{t}_{(q,b)}, b = 1 \ldots B\}$, each of which approximates T_0 to some quality index q. Each index q has an associated target distortion $D(q)$ which is set by the user to control the quality of the approximation. Further, T_q is designed to allow an efficient 2-stage implementation using the structure proposed in [4].

To generate T_q, we apply the matching pursuit algorithm itself in a novel way. Each $\vec{t}_{(0,b)}$ is to be approximated as a summation of simpler elements called "construction atoms." For a given b, each construction atom ψ has an associated weight $c_{(\psi,b)}$ and a unit norm basis shape $\vec{r}_{(\psi,b)}$, which is selected from an appropriately defined construction dictionary R_b. We use matching pursuit to decompose $\vec{t}_{(0,b)}$ on R_b. This produces a set of construction atoms Ψ_b which approximate $\vec{t}_{(0,b)}$ to quality index q as:

$$\vec{t}_{(q,b)} = \sum_{\psi \in \Psi_b} c_{(\psi,b)} \vec{r}_{(\psi,b)} \tag{2}$$

By expressing each original target $\vec{t}_{(0,b)}$ as a summation of simpler elements, the complexity advantages discussed in Section 2 may be exploited. Specifically, the collection $\{\Psi_b, b = 1 \ldots B\}$ is used to define the interconnect equations for a fast 2-stage matching pursuit encoder based on Figure 2.

A special dependency structure was built into the dictionary of Redmill, et.al. [4] in order to further reduce complexity. We build a similar structure into our approximated dictionary by constructing R_b in a special way. Consider the original dictionary functions $\vec{t}_{(0,b)}$ to be ordered by index b, with the approximations generated sequentially. For the first target with $b = 1$, we define R_b to be some pre-designed elementary dictionary S. For the later targets, R_b is defined as:

$$R_b = S \cup \{\vec{t}_{(q,b')}, b' < b\} \tag{3}$$

Later targets are thus constructed from the approximations of earlier targets. This allows inner product computation to be efficiently recycled in the 2-stage encoder implementation.

The generation algorithm decomposes each $\vec{t}_{(0,b)}$ on the appropriate dictionary R_b. Construction atoms are added until the following distortion criterion is met:

$$\left\| \vec{t}_{(q,b)} - \vec{t}_{(0,b)} \right\|^2 \leq D(q) \tag{4}$$

In this way, $D(q)$ controls the approximation quality. A smaller value of $D(q)$ results in a higher quality approximation. However, this requires additional construction atoms, each of which adds

Initialization:

Assume S, T_0, and $D(q)$ are given. Set T_q is initially empty.

Approximation:

For $b = 1$ to B;

- Decompose $\vec{t}_{(0,b)}$ using matching pursuit with dictionary R_b as defined in Equation (3). Take each coded atom to be a construction atom ψ in an approximation of the form given in Equation (2). Stop when the constructed approximation $\vec{t}_{(q,b)}$ satisfies distortion criterion (4).
- Update $T_q = T_q \cup \{\vec{t}_{(q,b)}\}$

Next b;

Fig. 3. A summary of the generation algorithm.

complexity to the interconnect equations shown in Figure 2, and hence to the final 2-stage implementation. There is thus a tradeoff between the quality of the approximation and the complexity of the resulting video encoder.

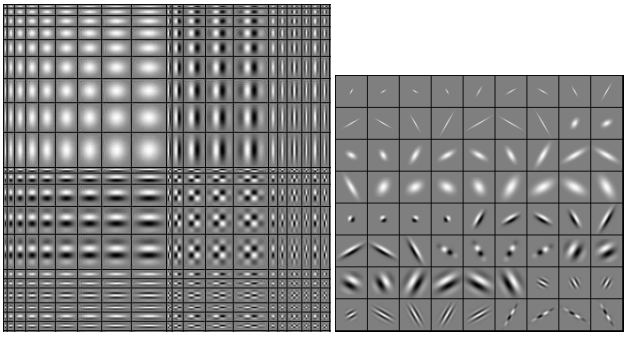
The dictionary approximation algorithm is summarized in Figure 3. Additional techniques are used to improve on this algorithm [8]. For example, the complexity gains associated with dependency are a function of the order in which the $\vec{t}_{(0,b)}$ are approximated. We employ a heuristic ordering algorithm which approximately orders these functions by size. Functions with large extent are placed late in the order, which allows them to be efficiently constructed from previously approximated functions of smaller extent. Complexity is further reduced by using orthogonal matching pursuit [7] in the generation algorithm. This reduces the average number of construction atoms required to achieve a given target distortion $D(q)$, and hence reduces the complexity of the final 2-stage implementation. Finally, it has been observed that similar target functions in T_0 may produce identical approximations in T_q. Such duplication is common when $D(q)$ is large and each target is built from only a few construction atoms. We add a natural merging step to eliminate these duplicates in the final 2-stage implementation. The dictionary is in this way pruned during the approximation process, and complexity is further reduced.

As a final note, our elementary dictionary S consists of 16 separable 2-D functions which approximate Gaussians. We use a factorized implementation similar to the one given in [5] to efficiently compute the elementary inner products.

4. RESULTS

To test the method, we generate approximations of two different original dictionaries with varying $D(q)$, and we implement each resulting dictionary in an efficient 2-stage matching pursuit encoder based on the structure of Figure 2. Each resulting encoder is used to code sequences from the standard MPEG-4 test set [2]. Average coding efficiency (Y-PSNR) and encoder complexity (ops/frame) are recorded for each combination of sequence, original dictionary, and approximation distortion level $D(q)$. The two original dictionaries are pictured in Figure 4. Figure 4(a) shows the separable 2-D Gabor dictionary used in our previous work [1][2]. We refer to this as the "std" dictionary, and we note that it may be implemented efficiently using separable 2-D filtering [1]. Figure 4(b) shows a set of Gabor functions rotated at multiples of 30 degrees which are added to the "std" set to produce a second dictionary which we call "h30".

Figure 5 shows two sample dictionary functions approximated to various $D(q)$. The first column shows the original target functions, and the remaining columns show the approximated functions. As seen in the figure, $D(q) = 0.1$ results in an accurate approximation of the original function. As $D(q)$ increases, each approximated function looks less and less like the original.



(a) (b)

Fig. 4. (a) The separable 2-D Gabor dictionary, "std". (b) Oriented Gabor functions added to form the "h30" dictionary.

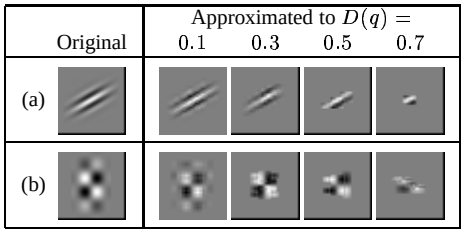


Fig. 5. Sample approximated functions from "h30" dictionary.

Figure 6 shows the coding efficiency and complexity tradeoffs which result when the Container sequence is coded with various approximations of the "std" dictionary. Figure 6(a) shows the average Y-PSNR plotted against $D(q)$. The dotted line shows the result for the original dictionary without approximation, and so matches our earlier published results [2]. Note that when $D(q)$ is small, the approximated dictionary performs about as well as the original. The approximate dictionary may even exceed the original in PSNR, since the merging of similar target functions tends to increase coding efficiency. We see near-original coding efficiency up to about $D(q) = 0.6$. At this point, Y-PSNR begins to decrease rapidly with $D(q)$. The corresponding complexity plot is shown in Figure 6(b). As expected, encoder complexity decreases as a function of $D(q)$. Note that the largest complexity, corresponding to $D(q) = 0.1$, is already about 7 times less complex than an efficient 2-D separable implementation of the original "std" dictionary. For $D(q) = 0.6$, the approximated dictionary is about 50 times less complex than the original [8].

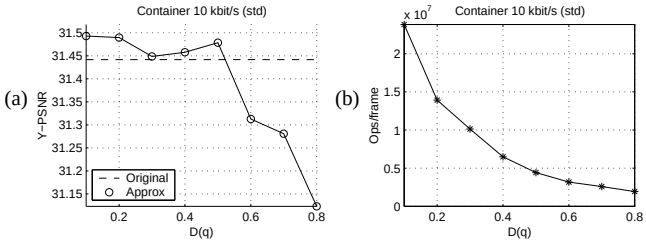


Fig. 6. Encoding result for "std" dictionary approximated to various distortion levels $D(q)$. (a) Average Y-PSNR for original and approximated dictionaries. (b) Average complexity of approximate dictionary implementations.

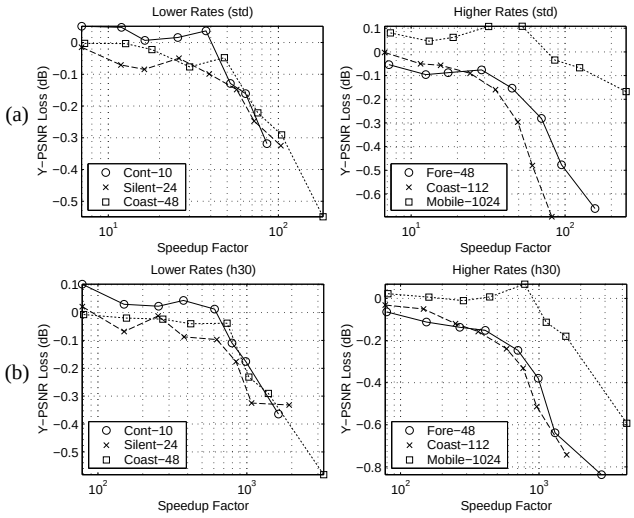


Fig. 7. Loss in Y-PSNR due to approximation error plotted directly against speedup factor. (a) Coding results using the "std" dictionary. (b) Results using "h30" dictionary.

Approximation tradeoffs for several sequences are compactly shown in Figure 7. The vertical axis of each plot shows the Y-PSNR loss due to approximation. In the context of Figure 6(a), this is computed by subtracting the dotted line from the solid line. On the horizontal axis of Figure 7, we plot the speedup factor due to the approximation. This is the complexity of the original dictionary divided by that of the approximated dictionary. Each data point represents the encoding of a single test sequence using one of our approximated dictionaries. Figure 7(a) shows the tradeoffs for the "std" dictionary. Original dictionary complexity is computed assuming the efficient 2-D separable implementation in [1]. The plots show that speedup factors above 40 are typically reached in exchange for PSNR losses of around 0.1 dB. Larger speedup factors are achieved in exchange for additional PSNR loss. For some sequences, speedup factors of 100 or more are seen for large $D(q)$.

Figure 7(b) shows the same tradeoffs for the "h30" dictionary. The plots show that speedup factors greater than 500 are typically seen in exchange for PSNR losses between 0.1 and 0.2. Speedup factors beyond 1000 are also possible, although the associated PSNR loss varies significantly by sequence. Note that speedup factors are computed relative to the complexity of the original dictionary, and the original "h30" dictionary is costly to implement. The oriented functions in "h30" lack a separable implementation, and so require full 2-D inner product computation. This makes the original "h30" dictionary about 20 times more complex than "std," and so the potential for complexity reduction is larger.

The advantage of the "h30" dictionary is improved visual and PSNR performance on sequences with diagonal edge content [8]. This is partially illustrated in Figure 8, which compares the tradeoffs between PSNR and complexity for the "h30" and "std" dictionaries. We see that "h30" shows small Y-PSNR gains over "std" at the same complexity for mild approximation levels, e.g. $D(q) < 0.5$. The curves merge beyond this point, indicating similar PSNR/complexity performance. This can be attibuted to the fact that when $D(q)$ is large, few dictionary functions survive the pruning process. Approximations of "std" and "h30" tend to become more similar through this process, and so have similar tradeoffs. Note that the Mother and Foreman sequences were selected because they contain significant diagonal edge content. For sequences like Container and Coast which lack diagonal edges, the "h30" dictionary shows no advantage over "std" [8].

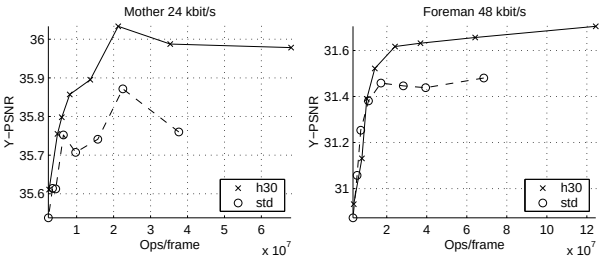


Fig. 8. Approximation tradeoffs for two sample sequences. Average Y-PSNR and and complexity are shown for each dictionary approximated with $D(q)$ ranging from 0.1 to 0.8.

5. CONCLUSION

In this work, we developed a method for generating approximations of arbitrary dictionaries with fast 2-stage implementations. For a given dictionary, this allows coding efficiency to be traded for encoder complexity in a systematic way. It also enables a new dictionary design paradigm which is free from initial restrictions based on complexity concerns. We applied the method to the 2-D separable "std" dictionary, and showed that complexity reduction factors between 40 and 80 are typical in exchange for small reductions in coding efficiency. We showed that larger gains are possible when the method is applied to the "h30" dictionary which does not have an efficient computational structure. For this dictionary, complexity reduction factors up to 1000 were shown in exchange for small losses in Y-PSNR. Dictionary approximation may thus be employed to automatically generate fast, approximate implementations of arbitrary matching pursuit dictionaries.

6. REFERENCES

[1] Ralph Neff and Avideh Zakhor, "Matching pursuit video coding at very low bit rates," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 158–171, February 1997.

[2] Ralph Neff, Toshio Nomura, and Avideh Zakhor, "Decoder complexity and performance comparison of matching pursuit and MPEG-4 video codecs," in *Proc. of IEEE International Conference on Image Processing*, 1998.

[3] C. de Vleeschouwer and B. Macq, "New dictionaries for matching pursuit video coding," in *Proc. of the IEEE International Conference on Image Processing*, 1998, pp. 764–768.

[4] David Redmill, David Bull, and Przemyslaw Czrepiński, "Video coding using a fast non-separable matching pursuits algorithm," in *Proc. of the IEEE International Conference on Image Processing*, 1998, pp. 769–773.

[5] Przemyslaw Czerepiński, Colin Davies, Nishan Canagarajah, and David Bull, "Dictionaries and fast implementation for matching pursuits video coding," in *Proceedings of the Picture Coding Symposium*, April 1999, pp. 41–44.

[6] Yao-Tang Chou, Wen-Liang Hwang, and Chung-Lin Huang, "Very low bit rate video coding based on gain-shape VQ and matching pursuits," in *Proc. of the IEEE International Conference on Image Processing*, 1999.

[7] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Proc. of 27th Asilomar Conference on Signals*, 1993, pp. 40–44.

[8] Ralph Neff, *Practical Methods for Matching Pursuit Video Compression*, Ph.D. thesis, U.C. Berkeley, December 2000.