Flexible Tree-Structured Signal Expansions Using Time-Varying Wavelet Packets

Zixiang Xiong, Member, IEEE, Kannan Ramchandran, Member, IEEE, Cormac Herley, Member, IEEE, and Michael T. Orchard, Member, IEEE

Abstract—In this paper, we address the problem of finding the best time-varying filter bank tree-structured representation for a signal. The tree is allowed to vary at regular intervals, and the spacing of these changes can be arbitrarily short.

The question of how to choose tree-structured representations of signals based on filter banks has attracted considerable attention. Wavelets and their adaptive version, known as wavelet packets, represent one approach that has proved very popular. Wavelet packets are subband trees where the tree is chosen to match the characteristics of the signal. Variations where the tree varies over time have been proposed as the double tree and the time-frequency tree algorithms. Time-variation adds a further level of adaptivity. In all of the approaches proposed so far, the tree must be either fixed for the whole duration of the signal or fixed for its dyadic subintervals (i.e., halves, quarters, etc.). The solution that we propose, since it allows much more flexible variation, is thus an advance on the wavelet packet algorithm, the double tree algorithm, and the recently proposed time-frequency tree algorithm.

Our solution to the problem is based on casting it in a dynamic programming (DP) setting. Focusing on compression applications, we use a Lagrangian cost of distortion \( + \lambda \times \)rate as the objective function and explain our algorithm in detail, pointing out its relation to existing approaches to the problem. We demonstrate the new algorithm indeed searches a larger library of representations than previously possible and that overcoming the constraint of dyadic time segmentations gives a significant improvement in practice. Compression experiments over various sources verify the superior performance of the new algorithm.

I. INTRODUCTION

ORTHOGONAL signal expansions play a key role in practical signal compression schemes from image or video coding standards using the discrete cosine transform (DCT) to the current state-of-the-art wavelet-based systems. The basic idea is to design a linear transformation to make the task of compressing a signal in the transform domain with (usually scalar) quantization easier than direct coding in the spatial domain. Orthogonal transformations are usually more desirable from a computational standpoint than nonorthogonal ones, due to their energy conservation property, which means that the \( L^2 \) spatial-domain error is equal to the \( L^2 \) transform-domain quantization error. For a given signal \( x \) and a fixed cost function \( J(\cdot) \), we seek a unitary transformation \( T \) such that \( J(Tx) \) is as small as possible. In order to describe a framework of universal representation, we include the overhead cost \( L(T) \) of sending a description of the transformation \( T \) and try to search for the best transformation that minimizes \( J(Tx) + L(T) \). An unconstrained search over all possible unitary transformations \( T \) is not feasible because if \( x \) is a 1–D signal of length \( N \), then the set of all real unitary \( N \times N \) matrices \( T \) is parametrized by a lattice of \( N(N-1)/2 \) rotation elements [1]. For example, when \( N = 64 \), finding the optimal \( T \) requires an exhaustive search over a continuous Lie group characterized by 2016 parameters [2].

In practice, suboptimal but signal-independent approximations like DCT are used for computational efficiency [3], with satisfactory results for many practical applications [4], [5]. The application of DCT is based on its proximity for Markov-1 source with large correlation to the Karhunen-Loève transform (KLT), which is the optimal block transform in terms of decorrelation and energy compaction under certain constraints [6].

Recently, wavelets and filter bank expansions, together with their generalizations like (time-varying) wavelet packets (WP), have appeared as alternatives to the classic Fourier and DCT expansions [7]–[10]. Instead of looking for the best unconstrained linear expansions, we limit ourselves to filter bank trees and seek the best (time-varying) tree-structured orthogonal basis for signal expansion (see Fig. 1). Although the bases we search are restricted to be tree structured, the set of all possible trees based on some filter set still provides a very large library of orthogonal bases. As we will show in Section IV-A, for a signal of length 64, the library size can be as large as \( 10^{17} \) bases. Furthermore, in addition to its size, the library of bases that we search is a sensible one and is motivated by time-frequency (TF) considerations. In all our algorithms, the library contains the wavelet and the uniform subband tree expansions, which are the most popular TF bases studied in the literature. An additional advantage of searching over a library of tree-structured bases is that the overhead cost \( L(T) \) can be easily quantified as the description length of the best filter bank tree structure. We will come back to this issue of side information in Section III-A.

In comparing expansions, we must fix a cost function \( J(\cdot) \) to use, which can be application dependent. Several cost measures have been suggested, such as the number of significant coefficients with respect to a threshold or entropy-
Based cost function [11], [12]. As the main interest of this paper is signal compression, we will use the rate-distortion \((R - D)\) cost function distortion +\(\lambda\times\)rate, where \(\lambda\) is the Lagrangian multiplier that equalizes rate and distortion [13]. This Lagrangian cost function that we use is a generalization of the distortion only and entropy only cost measures used in [11] since it includes both the distortion measure \((\lambda = 0)\) and the rate measure \((\lambda = -\infty)\) as special cases.

In Fig. 2, we plot the tiling representations of several tree-based expansions. This approach of viewing tree-structured bases as tilings of the time-frequency plane is explained in [14]. Fig. 2(a) shows the short time Fourier transform (STFT) like expansion, where all branches are grown to the same depth. Fig. 2(b) shows the familiar wavelet expansion or the logarithmic subband decomposition. Fig. 2(c) shows a wavelet packet expansion, where the best frequency division has been determined based on a chosen cost function, and Fig. 2(d) shows a time-varying WP expansion, where different trees are used over different segments.

Wavelet packet trees [11] or arbitrary subband decomposition trees represent generalizations of the wavelet tree. A single tree algorithm was used in [13] to find the best WP tree structure for compression by recursively pruning a full subband tree based on comparisons between the Lagrangian costs of a parent node and its children nodes in a bottom-up fashion. While WP expansions can choose among many frequency segmentations, they do not vary over time, and they cannot adapt to nonstationary signals. Introduction of time variation, in which we segment the signal in time and allow the bases to evolve with the signal, is a natural extension of the WP expansions. When the best time-varying filter bank tree structure is searched with respect to operational rate-distortion criteria, we effectively have a time-varying transform coding scheme, where both quantization \((D)\) and entropy coding \((R)\) are involved in the Lagrangian cost function. For finding the optimal tree-structured signal expansions in such a time-varying system, computationally efficient fast algorithms are very important. Fortunately, the advantages of using filter bank trees are that they are efficiently implementable [1], and fast algorithms can be found to search signal adaptively for the best transform from a rich and useful library of bases. When the filter bank tree changes over time, special treatments are necessary at the boundaries between signal segments. There are a number of ways of solving this problem efficiently [14]–[17]; we will use the boundary filter approach of [14] and [18].

A first approach to achieving this time-variation, where we obtain the best single tree bases over binary time segments of the signal, was explored in the double tree algorithm of [14]. An improvement and extension of the same idea was presented as the time-frequency tree algorithm in [19]. We will briefly review both in Section II-A. A simplified version of the TF tree algorithm for the special case of block transforms is considered in [20]. A very similar approach to that of [20] was used in [21] to search for the best wavelet packet basis and in [22] for the best Walsh basis.

The differences among the single tree algorithm, the double tree algorithm, and the TF tree algorithm are easily seen from the tilings generated by each algorithm. In Fig. 3(a), we show an example tiling achievable by the single tree algorithm. Each frequency split (horizontal line) lasts for the whole duration of the signal since the tree structure does not vary over time.
Fig. 3. Examples of time-frequency tilings achievable by different algorithms. (a) Tiling achievable by the single tree algorithm. (b) Tiling achievable by the double tree algorithm but not by the single tree algorithm. (c) Tiling achievable by the TF segmentation tree algorithm but not by the single tree or the double tree algorithm.

The drawbacks of the double tree algorithm and the TF tree algorithm are at least twofold: First, the binary (or M-ary in general) time segmentation constraint is very restrictive. A second drawback, which is a direct consequence of the first, is that the time segmentations obtained from the double tree and TF tree algorithms are very sensitive to time shifts of the original signal. For example, if the input signal were shifted by $k$ samples, the resulting chosen basis would change a great deal. In this paper, we provide a way of mitigating this shift-variance problem by describing a new flexible spatial segmentation tree algorithm [23] that is periodically shift invariant with fixed period $L$. That is, if the input is shifted by $k$ samples, the basis is substantially unchanged, provided that $k = 0 \pmod{L}$. In the new algorithm, the constraint of binary time segmentations is removed, and instead, the tree is constrained to change at any integer multiples of $L$. We apply a fast dynamic programming (DP) procedure to search for the best “flexible” spatial segmentation tree. This makes the system considerably less sensitive to time shifts. Although shift invariance can be achieved by introducing nonorthogonal expansions with overcomplete bases, with “greedy” suboptimal algorithms like matching pursuits being used to search for the best bases [24], in this paper, we use only orthogonal signal expansions and optimal algorithms. Another approach, which is based on the Viterbi algorithm and appears to be closely related to ours, was developed independently in [25], although neither is the rate-distortion cost considered there, nor is the treatment at the segmentation boundaries discussed (recall that we use boundary filters of [14]).

The new flexible spatial segmentation tree represents a generalization of the existing trees. The library of bases searched in the new algorithm contains the wavelet transform tree, as well as all the single tree, double tree, and TF tree bases. More importantly, by allowing flexible segmentation in the new algorithm, we have overcome what we perceive to be real shortcomings of these existing bases: the stationary nature of the wavelet and single tree bases and the constrained nature of the time variation in the double tree and the TF tree. The main contribution of this paper lies, therefore, in its formulation of a library of time-varying bases that is much more flexible than the existing ones and maintains the common feature of having a fast search algorithm. Our framework could thus be considered under the class of universal linear expansions of signals using fast filter bank tree topologies.

One of the attributes of this distinctive flexible spatial segmentation tree algorithm is that extension to multiple dimensions requires specification of a scan order for the multidimensional sequence. The results will thus be different, depending on which scan order is chosen. This complicates the use of this algorithm for image coding, whereas the algorithms previously mentioned extend without difficulty [13], [19], [26]. Nevertheless, our proposed new algorithm finds applications (with appropriate cost functions) in 1-D signal identification and compression due to its capability to provide flexible segmentations. It is particularly relevant to audio compression, where the nonstationarities of audio signals can be explored more efficiently. Experimental results confirm that better tree-structured bases can indeed be found by our new algorithm for signal identification and compression of both synthetic and real speech signals.

The rest of this paper is organized as follows: In Section II, we will briefly review the single tree algorithm, the double tree algorithm, and the TF tree algorithm. The new DP-based flexible spatial segmentation tree algorithm appears in Section III. In Section IV, we compare the different algorithms both in terms of number of bases searched and computational complexity. Simulation results on different classes of sources for different algorithms are presented in Section V. Finally, in Section VI, we draw our conclusions.

II. Previous Work

A. Single Tree Algorithm

We now provide a summary of the single tree algorithm, which forms the core of all the algorithms we introduce later, while referring the reader to [13] for details. The algorithm searches for the best WP basis (i.e., the best “stationary” frequency decomposition corresponding to the unsegmented original signal) in a lossy compression framework. An arbitrary discrete set of admissible quantization choices is assumed to quantize the WP coefficients in each tree node, with both rate ($R$) and distortion ($D$) being assumed to be additive cost metrics over the WP tree, i.e., $R(\text{node}) = \Sigma R(\text{leaf nodes})$; and $D(\text{node}) = \Sigma D(\text{leaf nodes})$. The constrained problem of seeking the best WP basis that minimizes the average distortion $D$ for a target average bit rate $R$ (or vice versa) can be converted to an equivalent unconstrained problem by merging rate and distortion through the Lagrange multiplier $\lambda$ [27]. Thus, the unconstrained problem becomes the minimization of the Lagrangian cost function defined as $J(\lambda) = D + \lambda R$.

The single tree algorithm is implemented through the following three stages.
Fig. 4. Single tree algorithm finds the best tree-structured wavelet packets basis for a given signal. (a) Algorithm starts from the full STFT-like tree and prunes back from the leaf nodes to the root node until the best pruned subtree is obtained. (b) At each node, the split-merge decision is made according to the criterion: prune if $J(\text{parent node}) \leq [J(\text{child1}) + J(\text{child2})].$

- **Stage 1:** Initialization.
- **Stage 2:** Optimality for a given Lagrange multiplier $\lambda$.
- **Stage 3:** Finding the optimal $\lambda$ for a given target bitrate.

In Stage 1, a full WP tree is grown (see Fig. 4(a)), with both rate and distortion gathered for each node of the tree.

In stage 2, each node of the full WP tree is first populated with the optimal Lagrangian cost associated with the best quantizer that minimizes the rate-distortion tradeoff, i.e.

$$J(\text{node}) = \min_{\text{quantizer}} [D(\text{node}) + \lambda R(\text{node})].$$

Then, this full WP tree is pruned recursively at each node by comparing its cost to the summation of the costs of its children nodes following a policy of (see Fig. 4 (b)):

**Prune if:** $J(\text{parent node}) \leq [J(\text{child1}) + J(\text{child2})].$

The pruning procedure starts from the leaf nodes and proceeds toward the root. At the end of this procedure, an optimal pruned subtree is obtained for a fixed $\lambda$, as shown in Fig. 4(a).

The desired optimal Lagrange multiplier is not known *a priori* and depends on the desired target bitrate or quality constraint. It is obtained in stage 3 via a fast convex recursion in $\lambda$ using the bisection algorithm.

The name for this algorithm is derived from the fact that a single (frequency) tree is optimally pruned. The minimum (Lagrangian) costs associated with the frequency trees for these signal segments are used to populate the hierarchically higher level spatial tree, which is then pruned in an identical manner as each of the frequency trees, i.e., using a single tree-like algorithm. The name here is derived from the two kinds of trees that are pruned: frequency trees (corresponding to the solid-line trees of Fig. 5) associated with each binary segment of the original signal and spatial trees (corresponding to the dotted-line trees of Fig. 5) associated with the spatial (or temporal) segmentations of the signal. The computational complexity of the double tree algorithm can be shown to be of $O(Nd^2)$ for a size $N$ (assumed again to be a power of 2) signal and a maximum tree depth $d$.

**B. Double Tree Algorithm**

While the single tree algorithm finds the best WP (stationary) frequency decomposition for the entire unsegmented signal, the double tree algorithm [14] addresses (binary) tree-structured time segmentations along with the best WP decompositions for each segment. It thus represents a hierarchical extension of the single tree to accommodate binary time splits motivated by the need to more efficiently address time-varying signal characteristics. The basic idea is simple and is easily explained through the example in Fig. 5, where a full double tree of depth 2 is shown. Note that the labels A, B, C, and D correspond to the first, second, third, and fourth quarters of the signal, respectively. The single tree algorithm is run for each binary segment of the original signal, i.e., in the example, corresponding to the full-length signal ABCD, the two half-length signals AB and CD and the four quarter-length signals A, B, C, and D. The minimum (Lagrangian) costs associated with the frequency trees for these signal segments are used to populate the hierarchically higher level spatial tree, which is then pruned in an identical manner as each of the frequency trees, i.e., using a single tree-like algorithm.

The double tree algorithm is run for each binary segment of the original signal, i.e., in the example, corresponding to the full-length signal ABCD, the two half-length signals AB and CD and the four quarter-length signals A, B, C, and D. The minimum (Lagrangian) costs associated with the frequency trees for these signal segments are used to populate the hierarchically higher level spatial tree, which is then pruned in an identical manner as each of the frequency trees, i.e., using a single tree-like algorithm.

The name here is derived from the two kinds of trees that are pruned: frequency trees (corresponding to the solid-line trees of Fig. 5) associated with each binary segment of the original signal and spatial trees (corresponding to the dotted-line trees of Fig. 5) associated with the spatial (or temporal) segmentations of the signal. The computational complexity of the double tree algorithm can be shown to be of $O(Nd^2)$ for a size $N$ (assumed again to be a power of 2) signal and a maximum tree depth $d$.

**C. Time-Frequency Tree Algorithm**

In the double tree algorithm, although we perform arbitrary binary frequency splits over various time segments, there is some asymmetry about the splits generated in that there are many frequency splitting trees grown over time segments, but the converse is not true. This is rectified by the TF tree shown in Fig. 6, in which both spatial and frequency subsignals are
treated as candidates for further segmentation and a so-called balanced double tree is thus obtained [19]. Obviously, a double tree in Fig. 5 is a special case of the TF tree in Fig. 6. Furthermore, the TF tree has an inherent quadtree structure, with each parent node having two pairs of children nodes: one pair corresponding to the two resulting spatial segments (dotted lines in the tree) and another pair corresponding to the two bands in frequency decomposition (solid lines in the tree).

The pruning of the TF tree amounts to comparing the Lagrangian costs of the spatial pair versus that of the frequency pair at each node of the tree from the leaf nodes to the root node and making a spatial/frequency segmentation choice based on the winner. While the loser from this comparison is eliminated, the winner moves on to the next round of comparison. At the end of the TF tree pruning, an optimal binary full depth tree is obtained, with each path indicating the decomposition choice (time or frequency) of the next level. This TF pruning procedure is different from the split-merge decision in the single tree pruning, where the optimal tree is a pruned binary subtree of the full subband tree. The name of this TF tree algorithm comes from the balanced treatment of frequency decomposition of time-segmented signals and time segmentation of frequency-decomposed signals. It can be shown that the computational complexity of the TF tree algorithm is \(O(N2^d)\) for a size \(N\) signal and a maximum tree depth \(d\) [19]. Although the complexity of the TF tree grows exponentially with tree depth, faster approximations like the block transform tree algorithm in [20], whose complexity grows linearly with tree depth, are possible when boundary effects can be neglected.

III. FLEXIBLE TIME SEGMENTATIONS

As mentioned in Section I, a significant limitation of the double tree algorithm and the TF tree algorithm is their binary restriction. As an example to highlight this point, suppose that we are given a signal with the statistical characteristics of its first quarter significantly different from the remaining three quarters; then, this signal should be segmented into two subsignals, with the first one consisting of just the first quarter and the second one the rest three quarters. However, when a binary tree-structured spatial segmentation is attempted on this signal, as in the double tree algorithm, in order to have the first quarter of the signal as a segmented subsignal, the original signal has to be segmented into two halves (AB and CD) at the first level of the tree, which is inefficient for the given statistics of the signal.

We now describe a flexible time segmentation tree algorithm aimed at relaxing this binary time segmentation constraint at the cost of added complexity. This is achieved by permitting time segmentations of “resolution” \(L\), which can begin at time instants comprising arbitrary integer multiples of \(L\) (see Fig. 7). Our new segmentation algorithm assumes that the cost metric distortion +\(\lambda\)rate is additive over disjoint spatial segments, which holds if orthogonal filter banks with appropriate orthogonal boundary filters [14] and entropy coding scheme are implemented.

For a 1-D signal of length \(N\) that we assume is a multiple of \(L\), say, \(N = ML\), with a maximum of \(M\) time segments (numbered from 0 to \(M - 1\)), the basic idea is to apply the procedure of DP, which is a widely used technique in solving engineering problems [28]. DP involves making decisions in
Fig. 8. Basic idea of DP in the flexible segmentation algorithm. (a) It starts with focusing on the first two segments \([0, 2L - 1]\) of the signal. There are only two possible segmentations in this case, and the one with smaller cost is chosen as the winner. (b) Then, it looks at the first three segments \([0, 3L - 1]\) of the signal. By utilizing the winner in (a), this DP-based algorithm needs only three comparisons to find the best segments instead of four comparisons in an exhaustive search. In general, by using the best segmentations of all of its admissible subsignals, the DP-based algorithm reduces an exponential search to a linear search.

stages, where at each stage, the best decision so far is made to minimize the cost function, and the outcome of each decision is observable before the next decision is made.

In our new algorithm, we apply DP to find the optimal segmentation for signal \([0, kL - 1]\) recursively for \(k \in \{1, 2, \ldots, M\}\) based on those for all of its admissible subsignals. Computational saving is achieved by using the known optimal segmentations for all previous subsignals in a recursive manner, thus avoiding an exhaustive search. This idea of DP in our new algorithm can be easily explained as in Fig. 8. We start by focusing on the first two segments \([0, 2L - 1]\) of the signal. There are only two possible segmentations, and finding the winning one with smaller cost is trivial (see Fig. 8(a)). Then, we look at the first three segments \([0, 3L - 1]\) of the signal. There are four possible ways of segmenting in this case, but by utilizing the information about the winner in Fig. 8(a), only three (instead of four) comparisons are needed to find the best segments, as shown in Fig. 8(b). In general, by using the best segmentations of all of its admissible subsignals, we can reduce an exponential search to a linear one for each subsignal. Therefore, although an exhaustive search for the best segmentation for the whole signal would require a searching complexity of \(O(2^M)\), the new DP-based flexible segmentation tree algorithm has a searching complexity of only \(O(M^2)\). The reason for this saving in searching complexity is the additivity assumption of the cost function, which guarantees that some searching operations for a longer signal are redundant once they are performed for its shorter admissible subsignals, and thus, suboptimal choices can be eliminated efficiently as soon as possible.

For a formal description of our fast DP-based flexible segmentation tree algorithm, we denote by \(\pi_k\) a segmentation of subsignal \([0, kL - 1]\), which consists of a sequence of \(p\) segments \(s_1, s_2, \ldots, s_p\). See Fig. 7 for an illustration of \(\pi_k\). Then, by the additivity assumption, the Lagrangian cost \(J_k\) associated with segmentation \(\pi_k\) for coding subsignal \([0, kL - 1]\) can be written as the sum of the costs for each individual segment, i.e., \(J_k = \sum_{i} \text{cost}(s_i)\), where \(\text{cost}(s_i)\) is the minimum cost to code segment \(s_i\).

The goal of optimal segmentation involves searching for the best \(\pi_M\), which is denoted as \(\pi_M^*\), that minimizes the Lagrangian cost \(J_M\) associated with segmentation \(\pi_M\) for coding the whole signal \([0, ML - 1]\), that is,

\[
\pi_M^* = \arg \min_{\pi_M} J_M = \arg \min_{\pi_M} \sum_{s_i \in \pi_M} \text{cost}(s_i). \tag{1}
\]

In order to determine a fast algorithm for the best segmentation, we make use of the right-hand side of (1) being an additive sum of independent terms corresponding to the segments in \(\pi_M\). This suggests using the standard approach of DP, which involves computing a set of recursively defined cost functions. Indeed, if we define \(J_k^*\) as the minimum Lagrangian cost associated with the best segmentation \(\pi_k^*\) for coding subsignal \([0, kL - 1]\), i.e., \(J_k^* = \sum_{s_i \in \pi_k^*} \text{cost}(s_i)\), then \(J_k^*\) can be seen to satisfy the recursive relationship

\[
J_k^* = \min_{0 \leq \ell < k} [J_{\ell}^* + \Delta J_{k, \ell}^*] \tag{2}
\]

for \(k \in \{1, 2, \ldots, M\}\), where \(\Delta J_{k, \ell}^*\) is the minimum Lagrangian cost (including side information) associated with the best WP frequency decomposition for segment \([\ell L, kL - 1]\).

Note that our aim is to find the optimal segmentation \(\pi_M^*\) for the signal, but once the minimum Lagrangian costs \(J_k^*\)’s are found for \(k \in \{1, 2, \ldots, M\}\), we are only one step away from our goal. Actually, if we use the minimizing argument of (2) in the process of finding the optimal \(J_k^*\’s\) to define a backtracking function \(b(k)\) of \(k\) for each \(k \in \{1, 2, \ldots, M\}\), that is,

\[
b(k) = \arg \min_{0 \leq \ell < k} [J_{\ell}^* + \Delta J_{k, \ell}^*], \tag{3}
\]

and maintain a record of \(b(k)\) during the computation of (2), then it is straightforward to recover the best segmentation \(\pi_M^*\) through the backtracking relationship \(t_i = b(t_{i+1})\), starting from the end of the signal. Here, \(t_i\) is the integer with \(t_i L\) being the beginning point of the \(i\)th segment of \(\pi_M^*\). Fig. 10 depicts this process of backtracking.

The fast DP-based flexible segmentation tree algorithm may be summarized as follows:

1) Run the single tree algorithm (see Section II-A) to find the best WP frequency split over all segments \([iL, jL - 1]\), where \(i \in \{0,1,\ldots,M-1\}\), \(j \in \{i+1,\ldots,M\}\). This will provide all Lagrangian costs \(\Delta J_{i,j}\).
The minimization of $J_S^k$ is achieved by the recursive relationship

$$J_S^k = \min_{0 \leq m < k} \{ J_{S,k}^l + \Delta_k \}, \quad \text{for } k \in \{1, 2, \ldots, M\}.$$ 

2) Compute $J_S^k$ as defined in (2) (initialized with $J_S^0 = 0$), and record $\bar{b}(k)$ for $k \in \{1, 2, \ldots, M\}$.

3) Read off the optimal segmentation for the original signal $[0, M L - 1]$, starting from the end of the signal, with the last segment being $b(0, M) = \bar{b}(M)$, the second to the last segment $b(1, M) = \bar{b}(M - 1)$, and so on. This backtracking procedure terminates when $b(t_i) = 0$ for some $i \leq M$.

A. Side Information to Send the Flexible Segmentation Tree

We now address the problem of side information. In all the algorithms covered in Section II, we did not explicitly mention the cost of sending the side information, namely, the tree-structure itself. For 2-D applications like image coding, it can be shown that the side information needed to transmit the time (or space)-frequency tree is equal to $1 + 4^1 + 4^2 + \cdots + 4^{d-1} = \frac{4^d - 1}{3}$ bits, where $d$ is the maximum tree depth [19]. When we limit the tree to some maximum depth in practice (e.g., four for 512 x 512 images), then the total number of overhead is 85 bits, which is reasonable to be ignored given the large amount of data in an image. Since the single tree algorithm and the double tree algorithm search fewer bases than the TF tree algorithm, even less side information is needed, and it remains reasonable to ignore side information in those cases.

In our proposed flexible segmentation tree algorithm, however, the side information, which contains all the overhead for conveying as many as $M = N/L$ frequency and spatial tree maps, will increase as the minimum segment length $L$ decreases, and it is improper to ignore the side information.

This is especially true in our treatment of 1-D signals, where the data size gain in dimensionality is lost.

In order to take into account the side information in our current DP-based flexible segmentation framework, we separate the spatial tree maps from the frequency tree maps. One bit is assigned for each $L$ length segment for sending the spatial tree maps; therefore, $M$ bits are needed as spatial side information. This corresponds to the worst case when we assume that segmentation occurs at each integer multiples of $L$ with equal probability. In practice, a better estimation of this spatial side information can be obtained by run length coding or arithmetic coding.

We incorporate the frequency side information for sending the frequency tree maps into the single tree pruning process. One bit of frequency tree information is added at each node of the single tree over all segments $[L_j, jL - 1]$, where $i \in \{0, 1, \ldots, M - 1\}, j \in \{i + 1, \ldots, \bar{M}\}$. In this way, the single tree algorithm automatically decides the best WP tree for each segment with the frequency side information being taken into account, trading off side information and the rate in sending quantized coefficients.

IV. COMPARISONS OF THE DIFFERENT ALGORITHMS

The fast DP-based flexible tree segmentation algorithm presented in the previous section is a new technique for applications using time-varying wavelet packets. It represents original contribution to this topic. Early results of this work appeared in [23]. The flexible tree segmentation algorithm improves the double tree algorithm in [14], which only allows tree-structured binary time segmentation. The TF tree algorithm described in [19] extends the double tree algorithm with added choice of applying time segmentation in the frequency domain. Both the double tree and the TF tree are subject to the binary (or $M$-ary in general) tree-structured time segmentation constraint, which we release in the flexible tree. A simplified version of the TF tree algorithm for the special case of block transforms is in [20]
TABLE I

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of Bases Searched</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>$S(N) = [S(N/2)]^2 + 1$, $S(2) = 2$</td>
</tr>
<tr>
<td>Double</td>
<td>$D(N) = [D(N/2)]^2 + S(N) - S(N/2)$, $D(2) = 2$</td>
</tr>
<tr>
<td>TF</td>
<td>$B(N) = 2[B(N/2)]^2 - [B(N/4)]^4$, $B(2) = 2$</td>
</tr>
<tr>
<td>Flexible</td>
<td>$F(N) = \sum_{i=1}^{\lfloor \log_2 N \rfloor} (S(2^i) - S(2^{i-1}))F(N - 2^i)$, $F(2) = 2$, $S(1) = 0$</td>
</tr>
</tbody>
</table>

A. Number of Bases Searched

For a 1-D signal of length $N$, when a two-channel filter bank is used, it can be easily shown that the number of WP bases (pruned subtrees) $S(N)$ searched by the single tree algorithm is given by the recursion $S(N) = [S(N/2)]^2 + 1$ with $S(2) = 2$. This is easily shown as follows. Any binary tree (except a tree with depth 0) can be considered having trees of depth one less than the original tree as its high and low branches. If the number of bases in each branch is given by $S(N/2)$, the total number of bases for the original tree is $S(N) = [S(N/2)]^2 + 1$.

In the double tree algorithm, for the sake of simplicity in the analysis, we assume that Haar filters (for lowpass and highpass filter, respectively) are used when no boundary filters are necessary. A similar recursive construction technique as shown above can be used to show that the number of bases $D(N)$ searched is given by $D(N) = [D(N/2)]^2 + S(N) - S(N/2)$ with $D(2) = 2$, where $S(N) - S(N/2)$ is the number of new single tree bases searched in the double tree algorithm when there is no spatial segmentation.

The number of bases $B(N)$ searched by the TF tree algorithm can be similarly shown (for Haar filters) as given by the recursion $B(N) = 2[B(N/2)]^2 - [B(N/4)]^4$, with $B(2) = 2$, See [19] for details. When non-Haar filters are implemented, the number of bases $B(N)$ searched by the TF tree algorithm becomes $B(N) = 2[B(N/2)]^2$, with $B(2) = 2$. The difference between the Haar filters and the non-Haar filters case arises because special treatments cross boundaries (boundary filters or periodic extensions) and, in the non-Haar filters case, generate new orthogonal bases.

For the flexible segmentation tree algorithm, we have the following proposition:

**Proposition 1:** For a 1-D signal of length $N$, for Haar filters, i.e., no boundary effect, and $L = 2$, the number of bases $F(N)$ searched by the DP-based flexible segmentation tree algorithm is given by the recursion

$$F(N) = \sum_{i=1}^{\lfloor \log_2 N \rfloor} (S(2^i) - S(2^{i-1}))F(N - 2^i)$$

with $F(2) = 2$, and $S(1) = 0$.

**Proof:** See Appendix A.

The number of bases searched by different algorithms for a signal of size $N$ is tabulated in Table I (assuming again that Haar filters are used ($L = 2$)). For example, when $N = 8$, $S(8) = 26$, $D(8) = 70$, $B(8) = 82$, $F(8) = 94$. When $N = 64$, $S(64) = 210 \times 10^{10}$, $D(64) = 9.78 \times 10^{14}$, $B(64) = 6.41 \times 10^{16}$, $F(64) = 1.06 \times 10^{17}$. Note that the gap between the algorithms widens exponentially as the signal length increases.

B. Computational Complexity

As stated in Section II, for a 1-D signal of length $N$ (assumed to be a power of 2) and a maximum tree depth $d$, the computational complexities of the single tree algorithm, the double tree algorithm, and the TF tree algorithm are $O(Nd)$, $O(Nd^2)$, and $O(N2^d)$, respectively. The computational complexity of the above flexible segmentation tree algorithm lies mainly in step 1 in the *population of the Lagrangian costs* $\Delta J_{ij}$, which requires running the single tree algorithm once for an $M_1L$ long segment (the original signal), twice for two $(M-1)L$ long segments, thrice for three $(M-2)L$ long segments, etc. Thus, we have the following proposition concerning the computational complexity of this algorithm.

**Proposition 2:** For a 1-D signal of length $N$ and a maximum number of segments $M(N = ML)$, suppose that the maximum single tree depth is $d$ for each segment; then, the computational complexity of the flexible segmentation tree algorithm is $O(NM2^d)$.

**Proof:** See Appendix B.

We want to point out that the above computational complexity can be further reduced in real implementation by exploiting the redundancies between the wavelet coefficients of the segmented signal and those of the unsegmented signal [29]. All that is needed is to update the wavelet coefficients at segmentation boundaries. Unfortunately, no closed-form formula can be derived for this case, which requires complicated bookkeeping.

For comparison with the single tree algorithm and the double tree algorithm [14], [13], [19], we list the computational complexity of each algorithm in Table II.

TABLE II

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Computational Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>$O(Nd)$</td>
</tr>
<tr>
<td>Double</td>
<td>$O(Nd^2)$</td>
</tr>
<tr>
<td>TF</td>
<td>$O(N2^d)$</td>
</tr>
<tr>
<td>Flexible</td>
<td>$O(NM2^d)$</td>
</tr>
</tbody>
</table>

V. EXPERIMENTAL RESULTS

We test our proposed flexible time segmentation algorithm on both synthetic sources and real speech signals. In all the experiments below, we choose $N = 768$ and $M = 128$ so that the smallest time segment is $L = 6$, although theoretically, arbitrary spatial resolution (with $L = 2$ by using the Haar filters) can be achieved through our algorithm since $L$ is only limited by filter length. Length 4 Daubechies $D_2$ filters and their boundary filters in [14] are used as the frequency
decomposition kernel $T$ because of their simplicity, i.e.,

$$
T = \begin{bmatrix}
    a_1 & a_2 & a_3 \\
    b_1 & b_2 & b_3 \\
    -h_3 & h_2 & h_0 \\
    -h_3 & -h_1 & h_0 \\
    c_1 & c_2 & c_3
\end{bmatrix} = \\
\begin{bmatrix}
    0.9941 & 2.977 & -1.719 \\
    -2.437 & 0.8133 & -0.4956 \\
    0.4830 & 0.8365 & 0.2241 & -1.1294 & 0.4830 \\
    0.1294 & 0.2241 & -0.8365 & 0.4034 & 0.6988 & -0.5907 \\
    0.2933 & 0.5116 & -0.8099
\end{bmatrix}.
$$

We use mean square error (MSE) as distortion (D) and the statistical first-order entropy of the quantized transform coefficients as rate (R) in the Lagrangian cost function distortion $+\lambda R$. Both spatial segmentation and the frequency tree-structured WP decomposition results within each spatial segment for each source signal are given.

A. Sensitivity to Time-Shifts: An Example

To demonstrate the reduced time-shift sensitivity of our proposed flexible segmentation tree algorithm in comparison with the double tree algorithm, we choose as our input signal a $D_2$ wavelet basis function derived from a wavelet tree of depth 3 as in [14]. The support of the basis signal is 22. When it is positioned “in-phase” (starting at $t = 8s + k + 2$), the double tree algorithm, which degenerates to the the single tree algorithm for this input, gives the correct depth 3 frequency split with one nonzero WP coefficient. Fig. 11(a) shows the input signal with the wavelet basis placed from $t = 314$ to $t = 335$ (in-phase), whereas Fig. 11(b) depicts how this basis function is localized in its tiling diagram. When we shift the signal in Fig. 11(a) by 60 samples, the tiling diagram obtained by the double tree algorithm is given in Fig. 11(c), which clearly illustrates that the tiling diagram given by the double tree algorithm is not shift invariant. By using the new flexible segmentation tree algorithm, however, we can alleviate, although not completely eliminate, the problem of sensitivity to time shifts. When the time shift is a multiple of segmentation resolution $L$, the new algorithm is able to segment the signal and give the tiling diagram that is shift invariant (see the tiling diagram in Fig. 11(d) for the basis signal shifted by $60 (=10L)$).

B. A Segmentation Example

1) Synthetic Signal: A Sinusoid Plus Triangular Impulses:
Fig. 12(a) shows a synthetic signal consisting of a sinusoid (from $t = 1$ to $t = 384$) and two triangular impulses: one (dashed line) from $t = 597$ to $t = 603$ and another one (solid line) from $t = 672$ to $t = 678$. Note that the peak of the first triangular impulse is located at a possible segmentation point ($t = 600$), whereas the second one can be ideally fitted into one segment. This signal is designed to further test the sensitivity of segmentation with respect to time shifts. The best WP decomposition tree associated with the whole signal is also plotted in Fig. 11(a). Fig. 11(b) and (c) show the optimal segmentation results using the double tree algorithm and the flexible segmentation tree algorithm with a scalar quantization stepsize of 0.1 (fine quantization) and $\lambda = 0$. We choose $\lambda = 0$ because rate is not the major concern in signal segmentation applications. As seen from Fig. 12(b), the double tree algorithm correctly splits the signal into two half but is not able to locate the impulse. The flexible segmentation tree algorithm, however, not only separates the sinusoid from the rest of the signal but also finds the impulses. A closer examination of Fig. 12(c) reveals that the flexible segmentation tree algorithm perfectly locates the second triangular impulse, whereas there is some segmentation artifacts for the first triangular impulse. A better segmentation.
point should be at $t = 606$, but the algorithm instead decides to put it at $t = 618$. This example shows that segmentation artifacts exist for our algorithm when the time shift is not a multiple of $L$. Nevertheless, it is capable of segmenting out both triangular impulses in the signal.

C. Compression Examples

1) First-Order Autoregressive Signal: To show the advantage of the flexible segmentation tree algorithm over the double tree algorithm, in this experiment, we use a sequence of samples generated from an autoregressive (AR) first-order Markov source with time-varying autocorrelation and noise power. The source for the first quarter of the sequence has variance = 100.0 and autocorrelation coefficient $\rho = 0.1$, the one for the second and third quarter has variance = 100.0 and $\rho = 0.9$, whereas the one for the fourth quarter has variance = 1.0 and $\rho = 0.1$. Fig. 13(a) shows this signal with its best WP tree (quantization stepsize of 10.0, $\lambda = 5.0$) with Fig. 13(b) and (c) showing the segmentation results of the double tree algorithm and the flexible segmentation tree algorithm. The new algorithm gives the correct segmentation.
algorithm. From Fig. 13(b) and (c), we can see that although the flexible segmentation tree algorithm gives the correct segments, the double tree algorithm fails to give any splits, i.e., it degenerates to the single tree algorithm. This is because of the constraint that spatial segmentation in the double tree algorithm has to be tree structured.

2) Speech Signal: Experiments were also performed on a speech signal with quantization stepsize = 100 and \( \lambda = 1.0 \), with results shown in Fig. 14(a)–(c) for the single tree algorithm, the double tree algorithm, and the flexible segmentation tree algorithm, respectively. As seen from Fig. 14(b) and (c), the best segmentations from the double tree algorithm and the flexible segmentation tree algorithm can be quite different.

To show the improved coding gain achievable by the flexible segmentation tree algorithm over the single tree and double tree algorithms, a comparison of rate and distortion (MSE) is depicted in Table III. In our experiments, we use first-order entropy as an approximation to the rate in the segmentation algorithms. A decodable bitstream is also generated by compressing the quantized transform coefficients of each segment using an arithmetic coder that is based on [30]. Note that in the adaptive wavelet packets case, the total rate consists of the rate of encoding the WP coefficients together with the overhead information needed to send the frequency WP trees and the spatial segmentation. From both FOE and the real coding rate, we see that the flexible segmentation tree algorithm performs the best, achieving about 10% bitrate saving at comparable or lower distortion.

VI. CONCLUSION AND DISCUSSIONS

In this paper, we have formulated a linear tree-structured signal expansion scheme optimized within the framework of time-varying filter banks. A fast DP-based flexible time segmentation algorithm is described that jointly finds the optimal time and frequency segmentations of a nonstationary signal. Compared with other WP tree algorithms, the new algorithm is more flexible and less sensitive to time shifts, and it also has its superiority in terms of better matching the time-frequency characteristics of the input signal for compression applications. Experimental results verify its superior performances.

Although this paper addresses new applications of time-varying wavelet packets, open problems still remain. One is the optimal segmentation resolution \( L \), which is obviously limited by the filter length. Computationally, as \( L \) decreases, more segmentation points need to be checked, and thus, the algorithm becomes more complex. Another topic for future research is the extension of the DP-based segmentation algorithm presented in this paper from 1-D to higher dimensions. In particular, it will be an interesting and challenging research project to build an optimal segmentation-based wavelet packet image coder.

APPENDIX A
PROOF OF PROPOSITION 1

Suppose \( N = 2K \) is the signal length (\( N \) has to be even); then, we can relate \( F(N) \) and the number of bases for its shorter subsignals by the “convolution sum”

\[
F(N) = \sum_{j=1}^{K} ST(jL)F(N - jL)
\]

where \( ST(jL) \) is the number of new single tree bases for a subsignal of length \( jL \), which is also the number of single
trees having at least one branch going to depth \( \log(jL) \). We assume that it is impossible to have a full frequency split when the signal length \( jL \) is not a power of two, i.e., \( ST(jL) = 0 \) when \( jL \neq 2^i \) for some \( i \). Then

\[
F(N) = \sum_{i=1}^{\lceil \log N \rceil} ST(2^i)F(N - 2^i),
\]

Note that \( ST(2^i) = S(2^i) - S(2^{i-1}) \), where \( S(2^i) \) is the number of bases searched by the single tree algorithm for a signal of length \( 2^i \). Therefore, we have

\[
F(N) = \sum_{i=1}^{\lceil \log N \rceil} (S(2^i) - S(2^{i-1}))F(N - 2^i).
\]

**APPENDIX B**

**PROOF OF PROPOSITION 2**

Since the computational complexity of the single tree algorithm for a length \( N \) signal with maximum tree depth \( d \) is \( O(Nd) \) and the flexible segmentation tree algorithm requires running the single tree algorithm once for an \( ML \)-long segment (the original signal), twice for two \((M - 1)\) long segments, thrice for three \((M - 2)\) long segments, etc., then the complexity of the flexible segmentation tree algorithm is

\[
O(MLd + 2(M - 1)Ld + \cdots + MLPd) = O\left( \sum_{i=1}^{M} k(M - k + 1)Ld \right) = O\left( \frac{M(M + 1)(M + 2)}{6} Ld \right) = O(N^3d) = O(NM^2d).
\]

**ACKNOWLEDGMENT**

The authors would like to thank the reviewers for their insightful comments, which greatly improved the quality of this paper.

**REFERENCES**


**Zixiang Xiong** (S’92–M’96) was born in Huangshi, Hubei, P.R. China, on April 21, 1967. He received the B.S.E.E. degree from Wuhan University, China, in 1987, the M.A. degree in mathematics from the University of Kansas, Lawrence, in 1991, the M.S.E.E. degree from the Illinois Institute of Technology, Chicago, in 1992, and the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign, in 1996.

From September 1987 to December 1989, he was a graduate student at the Institute of Electronics, Academia Sinica, Beijing, China. During the summer of 1995, he interned with Videonics Inc., San Jose, CA, and Xerox Company, Webster, NY. Since September 1995, he has been visiting Princeton University, Princeton, NJ, and working for the David Sarnoff Research Center, Princeton, NJ. His research interests include image/video compression, wavelets, image printing and display, and image recovery.
Kannan Ramchandran (M’93) received the B.S. degree from the City College of New York, and the M.S. and Ph.D. degrees from Columbia University, New York, NY, all in electrical engineering in 1982, 1984, and 1993, respectively.

From 1984 to 1990, he was a Member of the Technical Staff at AT&T Bell Laboratories, Murray Hill, NJ, in the telecommunications research and development area in optical fiber carrier systems. From 1990 to 1993, he was a Graduate Research Assistant at the Center for Telecommunications Research at Columbia University. Since 1993, he has been with the Department of Electrical and Computer Engineering and a Research Assistant Professor at the Beckman Institute and the Coordinated Science Laboratory, University of Illinois. His research interests include image and video compression, multirate signal processing and wavelets, and image communications.

Dr. Ramchandran was the recipient of the Columbia University 1993 “Eliahu I. Jury Award” for the best doctoral thesis in the area of systems, signal processing, or communications. He received an NSF Research Initiation Award in 1994 and an Army Research Office Young Investigator Award in 1996.

Cormac Herley (M’93) was born in Cork, Ireland, in 1964. He received the B.E. (Elect.) degree from the National University of Ireland in 1985, the M.S.E.E. degree from the Georgia Institute of Technology, Atlanta, in 1987, and the Ph.D. degree from Columbia University, New York, NY, in 1993.

From 1987 to 1989, he worked for Kay Elemetrics Corp., Pine Brook, NJ. In 1993, he was with AT&T Bell Laboratories, Murray Hill, NJ. Since 1994, he has worked at Hewlett-Packard Laboratories, Palo Alto, CA. During 1996, he also taught at the University of California, Berkeley.

Dr. Herley is an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY.

Michael T. Orchard (S’88–M’90) was born in Shanghai, China, and grew up in New York, NY. He received the B.S. and M.S. degrees in electrical engineering from San Diego State University, La Jolla, CA, in 1980 and 1986 and the M.A. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, in 1988 and 1990.

He worked at the Government Products Division of Scientific Atlanta from 1982 to 1986, developing passive sonar digital signal processing applications. Since 1988, he has consulted with the Visual Communications Department of AT&T Bell Laboratories. From 1990 to 1995, he was an Assistant Professor with the Department of Electrical and Computer Engineering at the University of Illinois, Urbana-Champaign, where he served as Associate Director of the Image Laboratory of the Beckman Institute. Since Summer 1995, he has been an Associate Professor with the Department of Electrical Engineering at Princeton University. His research interests include image and video coding with emphasis on motion estimation and compensation, wavelet representations of images and video, and image display.

Dr. Orchard received the National Science Foundation Young Investigator Award in 1993 and the Army Research Office Young Investigator Award in 1996.