Patch-Ordering-Based Wavelet Frame and Its Use in Inverse Problems

Idan Ram, Israel Cohen, *Senior Member, IEEE*, and Michael Elad, *Fellow, IEEE*

***Abstract*—In our previous work [1] we have introduced a redundant tree-based wavelet transform (RTBWT), originally designed to represent functions defined on high dimensional data clouds and graphs. We have further shown that RTBWT can be used as a highly effective image-adaptive redundant transform that operates on an image using orderings of its overlapped patches. The resulting transform is robust to corruptions in the image, and thus able to efficiently represent the unknown target image even when it is calculated from its corrupted version.**

In this work we utilize this redundant transform as a powerful sparsity-promoting regularizer in inverse problems in image processing. We show that the image representation obtained with this transform is a frame expansion, and derive the analysis and synthesis operators associated with it. We explore the use of this frame operators to image denoising and deblurring, and demonstrate in both these cases state-of-the-art results.

***Index Terms*—Patch-based processing, redundant wavelet, frames, denoising, deblurring, ordering, regularization.**

I. Introduction

Sparse and redundant representations and the processing of local patches have become two of the most popular approaches in image processing in recent years. While image processing algorithms may be based only on patch processing [2], [3] or sparse representations [4], [5], many current state-of-the-art algorithms make use of both concepts, usually by processing the image patches using sparsifying transforms or learned dictionaries [6]–[11].

In our previous work [1], [12] we have combined the two aforementioned approaches in a different manner, and used image patches to construct both an orthogonal and a redundant wavelet transforms, which efficiently (sparsely) represent entire images. These two wavelet transforms were originally designed to represent scalar functions defined on high-dimensional data clouds and graphs. However, we have also shown in [1], [12] that the very same construction can be used as an image-adaptive transform that is highly effective for sparsifying image content. This is obtained by converting the given image into a graph by considering all its overlapped patches as coordinates in high-dimensional space, and referring to them as features of the graph vertices. These are accompanied by their mutual Euclidean distance to define the graph edges, this way tying the vertices to each other. Due to the reliance on patches, both transforms are robust to corruptions in the image, such as additive noise, blur, or missing values, and are able to efficiently represent the unknown target image even when it is calculated from its corrupted version. As we shall see hereafter, this work will utilize this last property, and demonstrate the use of our redundant transform proposed in [1], (termed redundant tree-based wavelet transform – RTBWT), as a powerful sparsity-promoting regularizer in inverse problems.

More specifically, the RTBWT is calculated for an image by adding data-dependent operators, merged into the classical redundant wavelet filter-bank implementation [13], [14]. In each decomposition level several operators are used to reorder the approximation coefficients before the wavelet filters are applied to them. The reordering operators are obtained by organizing feature vectors constructed from the image patches, such that they are chained in the shortest possible path, essentially obtaining an approximation to the solution of the traveling salesman problem (TSP) [15]. These permutation operators increase the regularity of the approximation coefficient signals, thus causing their representation with the RTBWT to be more efficient (sparse).

As said above, in this work we utilize the RTBWT as a powerful sparsity-promoting regularizer in inverse problems in image processing. We start by introducing a simpler implementation to the RTBWT, which is based on the widely known "à trous" algorithm [16], [17]. This algorithm is a different implementation of the redundant wavelet transform that applies in each decomposition level upsampled versions of the wavelet filters to the whole approximation coefficient vectors. Thus, our scheme essentially adds to each decomposition level of this transform a single permutation operator that reorders the approximation coefficients before they are filtered, leading to the "à trous" implementation of our transform. We use this scheme and ideas from [18] to show that the RTBWT is a valid frame expansion, with the same bounds as the common redundant wavelet transform.

In our previous works [1], [3], [12] we observed that the performance of patch-ordering-based algorithms improve when a subimage averaging scheme is used. For algorithms using the RTBWT, this consists of applying the transform to different subimages of the treated image and then jointly operating on the transform coefficients of all the subimages. We refer to all the transform coefficients of the different subimages as a single extended representation of the treated image, and next take a path similar to the one described in

I. Ram and I. Cohen are with the Department of Electrical Engineering, Technion – Israel Institute of Technology, Technion City, Haifa 32000, Israel. E-mail addresses: idanram@tx.technion.ac.il (I. Ram), icohen@ee.technion.ac.il (I. Cohen); tel.: +972-4-8294731; fax: +972-4-8295757. M. Elad is with the Department of Computer Science, Technion – Israel Institute of Technology, Technion City, Haifa 32000, Israel. E-mail address: elad@cs.technion.ac.il

The research leading to these results has received funding from the European Research Council under European Union's Seventh Framework Program, ERC Grant agreement no. 320649, and was supported by Robert H. Hillman Foundation for Global Security - Collaboration Technion and University Northeastern, and by the Israel Science Foundation (grant no. 1130/11), and by Japan Technion Society Research Fund.

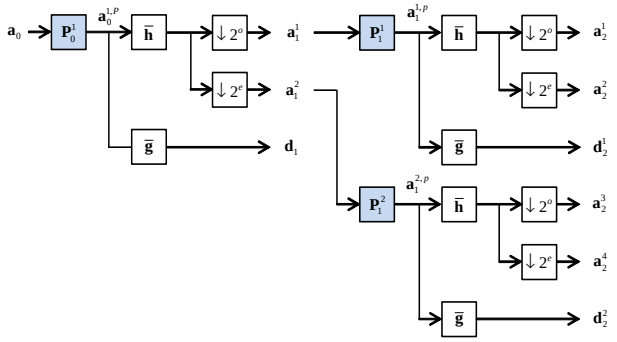

Fig. 1: RTBWT decomposition scheme.

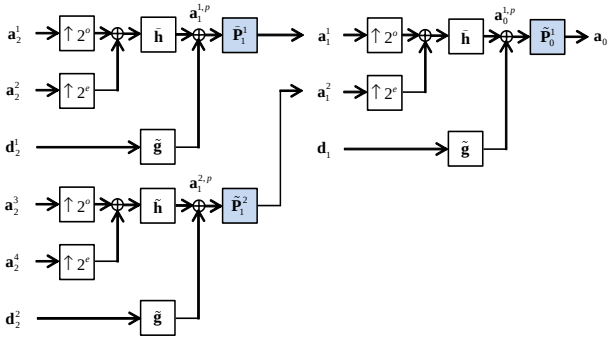

Fig. 2: RTBWT reconstruction scheme.

Task: Apply on $\mathbf{y}$ an L-level RTBWT decomposition.
Parameters: The image $\mathbf{y}$, the operators $\{\mathbf{P}_\ell^s\}_{s=1}^{2^\ell}$, $\ell = 0 \dots L-1$, and the filters $\bar{\mathbf{h}}$ and $\bar{\mathbf{g}}$.
Initialization: Set $\mathbf{a}_0^1 = \mathbf{a}_0 = \mathbf{y}$.
Main Iteration: Perform the following steps for $\ell = 0, \dots, L-1$:

- Perform the following steps for $s = 1, \dots, 2^\ell$:
  - Apply $\mathbf{P}_\ell^s$ on $\mathbf{a}_\ell^s$ and obtain $\mathbf{a}_\ell^{s,p}$.
  - Filter $\mathbf{a}_\ell^{s,p}$ with $\bar{\mathbf{g}}$ and obtain $\mathbf{d}_{\ell+1}^s$.
  - Filter $\mathbf{a}_\ell^{s,p}$ with $\bar{\mathbf{h}}$ and decimate the result with
    - $\downarrow 2^o$ and obtain $\mathbf{a}_{\ell+1}^{2s-1}$.
    - $\downarrow 2^e$ and obtain $\mathbf{a}_{\ell+1}^{2s}$

Output: The approximation coefficient vectors $\mathbf{a}_L$ and detail coefficient vectors $\{\mathbf{d}_\ell\}_{\ell=1}^L$.

Algorithm 1: RTBWT L-level decomposition scheme.

Task: reconstruct $\mathbf{y}$ from an L-level RTBWT decomposition.
Parameters: The approximation coefficient vectors $\mathbf{a}_L$ and detail coefficient vectors $\{\mathbf{d}_\ell\}_{\ell=1}^L$, the operators $\{\tilde{\mathbf{P}}_\ell^s\}_{s=1}^{2^\ell}$, $\ell = 0 \dots L-1$, and the filters $\tilde{\mathbf{h}}$ and $\tilde{\mathbf{g}}$.
Main Iteration: Perform the following steps for $\ell = L, \dots, 1$:

- Perform the following steps for $s = 1, \dots, 2^{\ell-1}$:
  - Interpolate $\mathbf{a}_\ell^{2s-1}$ with $\uparrow 2^o$ and $\mathbf{a}_\ell^{2s}$ with $\uparrow 2^e$, sum the results, and filter with $\tilde{\mathbf{h}}$.
  - Filter $\mathbf{d}_\ell^s$ with $\tilde{\mathbf{g}}$.
  - Sum the results of the two previous steps and obtain $\mathbf{a}_{\ell-1}^{s,p}$.
  - Apply $\tilde{\mathbf{P}}_{\ell-1}^s$ on $\mathbf{a}_{\ell-1}^{s,p}$ and obtain $\mathbf{a}_{\ell-1}^s$.

Output: The reconstructed signal $\mathbf{a}_0 = \mathbf{y}$.

Algorithm 2: RTBWT L-level reconstruction scheme.

[11] for the BM3D algorithm. We construct matrices which act as analysis and synthesis operators [19], and are used to obtain this extended representation and reconstruct the image from it. We then show that these matrices constitute a frame and its dual. We explore the use of these operators in image denoising and deblurring algorithms. Despite the fact that the resulting transform is image dependent, we treat it as a fixed linear operator and use it within a sparsity-promoting regularizer, when handling image-processing tasks posed as inverse problems. We demonstrate state-of-the-art results in denoising and deblurring using this approach.

The paper is organized as follows: In Section II, we describe the image-derived redundant tree-based wavelet transform. We also introduce the à trous based implementation of the transform, and use it to analyze its frame properties. Section III introduces the RTBWT subimage averaging frame, and analyzes its properties. Sections II and III can be skipped by readers interested in the image processing applications side of this work. In Section IV, we explore the use of this frame to image denoising and deblurring, and present experimental results that demonstrate their advantages. We summarize the paper in Section V.

II. Redundant Tree-Based Wavelet Transform

This and the next sections are dedicated to a careful construction and study of the frames that will be later used for regularizing inverse problems. Readers that are more interested in the inverse problem formulation and its subsequent applications can skip these two sections and proceed their reading in Section IV.

A. Decomposition and Reconstruction Schemes

Let $\mathbf{Y}$ be an image of size $N_1 \times N_2$ where $N_1 N_2 = N$, and let $\mathbf{y}$ be the column stacked version of $\mathbf{Y}$. The redundant tree-based wavelet transform (RTBWT), introduced in [1], is designed to efficiently (sparsely) represent its input vector, which in our case is $\mathbf{y}$. The transform is constructed by modifying an implementation of the redundant wavelet transform proposed by Shensa [13] and Beylkin [14]. Figure 1 describes the decomposition scheme of the RTBWT. $\bar{\mathbf{h}}$ and $\bar{\mathbf{g}}$ denote the wavelet decomposition filters of an orthonormal discrete wavelet transform, and they are applied using cyclic convolution. $\downarrow 2^o$ and $\downarrow 2^e$ denote $2:1$ decimators that keep the odd and even samples of their input, respectively. $\mathbf{a}_\ell$ and $\mathbf{d}_\ell$ denote the approximation and detail coefficient vectors in the ℓth scale, respectively, where $\mathbf{a}_0 = \mathbf{y}$. We note that for $\ell > 1$ these signals do not appear explicitly in the decomposition scheme, and instead it employs the signals denoted by $\mathbf{a}_\ell^s$ and $\mathbf{d}_\ell^s$, which contain subsets of the samples in $\mathbf{a}_\ell$ and $\mathbf{d}_\ell$, respectively. $\mathbf{a}_\ell^s$ is obtained from $\mathbf{a}_\ell$ by starting from the sth

sample, and keeping every 2^ℓth sample. $\mathbf{d}_\ell^s$ is obtained from $\mathbf{d}_\ell$ by starting from the sth sample, and keeping every $2^{\ell-1}$th sample. We notice that $\mathbf{a}_0^1 = \mathbf{a}_0$ and $\mathbf{d}_1^1 = \mathbf{d}_1$.

The operators $\mathbf{P}_\ell^s$ make the difference between our proposed wavelet decomposition scheme and the common redundant wavelet transform [13], [14]. Each such operator produces a permuted version $\mathbf{a}_\ell^{s,p}$ of its input vector $\mathbf{a}_\ell^s$, and it may be interpreted as a linear and unitary operator given that vector. These operators "smooth" the approximation coefficient signals in the different levels of the decomposition scheme. In Section II-B we explain how to obtain these operators from the image patches. Assuming that the operators $\mathbf{P}_\ell^s$ are known, Algorithm 1 is used to apply an L-level RTBWT decomposition.

In a similar manner, Figure 2 describes the reconstruction scheme of the redundant tree-based wavelet transform. If $\mathbf{h}$ and $\mathbf{g}$ denote the wavelet reconstruction filters, then $\tilde{\mathbf{h}} = \frac{1}{2}\mathbf{h}$ and $\tilde{\mathbf{g}} = \frac{1}{2}\mathbf{g}$ are applied using cyclic convolution. The interpolators denoted by $\uparrow 2^o$ and $\uparrow 2^e$ place the samples of their input vector in the odd and even locations of their output vector, respectively. The operator $\tilde{\mathbf{P}}_\ell^s$ reorders a vector so as to cancel the ordering done by $\mathbf{P}_\ell^s$, i.e. $\tilde{\mathbf{P}}_\ell^s = (\mathbf{P}_\ell^s)^{-1} = (\mathbf{P}_\ell^s)^T$. Assuming that the operators $\tilde{\mathbf{P}}_\ell^s$ are known, Algorithm 2 reconstructs $\mathbf{y}$ from an L-level RTBWT decomposition. We next explain how the operators $\mathbf{P}_{\ell,s}^t$ are determined in each level of the RTBWT.

B. Building the Operators $\mathbf{P}_\ell^s$

We wish to design the operators $\mathbf{P}_\ell^s$ in a manner which results in an efficient (sparse) representation of the input image by the proposed transform. The wavelet transform is known to produce a small number of large coefficients when it is applied to piecewise regular signals [16]. Thus, we would like the operator $\mathbf{P}_\ell^s$, applied to $\mathbf{a}_\ell^s$, to produce a signal $\mathbf{a}_\ell^{s,p}$ which is as regular as possible. We start with the finest level, and try to find the permutation that the operator $\mathbf{P}_0^1$ applies to $\mathbf{a}_0 = \mathbf{y}$. When the image $\mathbf{Y}$ is known, the optimal solution would be to apply a simple sort operation on $\mathbf{y}$. However, since we are interested in the case where $\mathbf{y}$ may be corrupted (noisy, blurred, contain missing pixels, etc.), we would try to find a near-optimal ordering operation using the image patches.

Let y_i denote the ith sample in the vector $\mathbf{y}$, and let $\mathbf{x}_i$ denote the column stacked version of the $\sqrt{n} \times \sqrt{n}$ patch around the location of y_i in $\mathbf{Y}$. A key assumption in our work is that under some distance measure $w(\mathbf{x}_i, \mathbf{x}_j)$, proximity between the two patches $\mathbf{x}_i$ and $\mathbf{x}_j$ suggests proximity between their center pixels y_i and y_j. Thus, we shall try to reorder the patches $\mathbf{x}_i$ so that they form a smooth path, hoping that the corresponding reordered one-dimensional (1D) signal $\mathbf{a}_0^{1,p}$ will also be smooth. The "smoothness" of the reordered signal $\mathbf{a}_0^{1,p}$ can be measured using its total variation measure

$$\|\mathbf{a}_0^{1,p}\|_{TV} = \sum_{j=2}^{N} |a_0^{1,p}(j) - a_0^{1,p}(j-1)|. \tag{1}$$

Let $\{\mathbf{x}_j^p\}_{j=1}^N$ denote the patches $\{\mathbf{x}_i\}_{i=1}^N$ in their new order. Then by analogy, we evaluate the "smoothness" of the path through the patches $\mathbf{x}_j^p$ by the measure

$$TV(\mathbf{x}_j^p) = \sum_{j=2}^{N} w(\mathbf{x}_j^p, \mathbf{x}_{j-1}^p). \tag{2}$$

We note that we can treat the patches $\mathbf{x}_i$ as points in $\mathbb{R}^n$, and the image as a function y defined on these points. In case that w is the Euclidean distance, and assuming that y is Lipschitz continuous, i.e., there exists a real constant $K \geq 0$ such that

$$|y_i - y_j| \leq K\|\mathbf{x}_i - \mathbf{x}_j\|_2 \tag{3}$$

for every two patches $\mathbf{x}_i$ and $\mathbf{x}_j$, we obtain

$$\sum_{j=2}^{N} |a_0^{1,p}(j) - a_0^{1,p}(j-1)| \leq K \sum_{j=2}^{N} \|\mathbf{x}_j^p - \mathbf{x}_{j-1}^p\|_2, \tag{4}$$

which means that $K \cdot TV(\mathbf{x}_j^p)$ is an upper bound for $\|\mathbf{a}_0^{1,p}\|_{TV}$, and thus seeking the shortest path with respect to the set $\{\mathbf{x}_j^p\}_{j=1}^N$ does a good service also to the original TV measure in Equation (1).

Minimizing $TV(\mathbf{x}_j^p)$ comes down to finding the shortest path that passes through the set of points $\mathbf{x}_i$, visiting each point only once. This can be regarded as an instance of the traveling salesman problem (TSP) [15], which can become very computationally exhaustive for large sets of points. We choose a simple and crude approximation to the solution, which is to start from an arbitrary point (random or predefined), and continue from each point to its nearest neighbor, not visiting any point twice. Let $\mathbf{q}_j$ denote a vector containing the two-dimensional (2D) spatial coordinate of the patch $\mathbf{x}_j$ in the image $\mathbf{Y}$. We restrict the nearest neighbor search performed for each patch $\mathbf{x}_j$ to a square neighborhood of size $B \times B$ around $\mathbf{q}_j$. When no unvisited patch remains in that neighborhood, we search for nearest neighbors among all the unvisited patches in the whole image. This restriction decreases the overall computational complexity, and our experiments show that with a proper choice of B it also leads to improved results, as it forces more relevant neighbors in the ordering. The permutation applied by the operator $\mathbf{P}_0^1$ is defined as the order of the found path.

In order to use the aforementioned method to find the operators $\{\mathbf{P}_\ell^s\}_{s=1}^{2^\ell}$ which are applied to the signals $\{\mathbf{a}_\ell^s\}_{s=1}^{2^\ell}$ in a scale $\ell > 0$, we are required to associate a set of feature points with these signals . More specifically, we predict the proximity between the samples of the signal $\mathbf{a}_\ell^s$, by associating a feature point $\mathbf{x}_{\ell,j}^s$ with each sample $a_\ell^s(j)$. Also, in order to measure the spatial proximity between the feature points, we associate a 2D spatial coordinate $\mathbf{q}_{\ell,j}^s$ to each feature point $\mathbf{x}_{\ell,j}^s$. The calculation of the feature points and their coordinates is carried out in a recursive manner. We use the set of ℓ-th scale feature points $\{\mathbf{x}_{\ell,j}^s\}_{j=1}^{2^{-\ell}N}$ to calculate the two sets of feature points $\{\mathbf{x}_{\ell+1,j}^{2s-1}\}_{j=1}^{2^{-\ell-1}N}$ and $\{\mathbf{x}_{\ell+1,j}^{2s}\}_{j=1}^{2^{-\ell-1}N}$, used in the ℓ+1th scale. This is done in analogy to the way we obtain the signals $\mathbf{a}_{\ell+1}^{2s-1}$ and $\mathbf{a}_{\ell+1}^{2s}$ from the signal $\mathbf{a}_\ell^s$. We first order the feature points $\{\mathbf{x}_{\ell,j}^s\}_{j=1}^{2^{-\ell}N}$ according to the permutation defined by $\mathbf{P}_\ell^s$, and place them in a matrix $\mathbf{X}_\ell^{s,p}$. We then filter the rows of the result with $\bar{\mathbf{h}}^T$, and obtain the sets of feature points

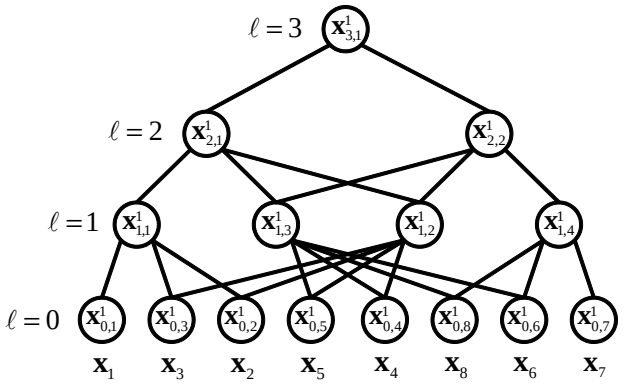

Fig. 3: An illustration of a "generalized" tree.

$\{\mathbf{x}^{2s-1}_{\ell+1,j}\}_{j=1}^{2^{-\ell-1}N}$ and $\{\mathbf{x}^{2s}_{\ell+1,j}\}_{j=1}^{2^{-\ell-1}N}$ by keeping only the odd and even columns of the resulting matrix, respectively.

A similar process is used to calculate the two sets of spatial coordinates $\{\mathbf{q}^{2s-1}_{\ell+1,j}\}_{j=1}^{2^{-\ell-1}N}$ and $\{\mathbf{q}^{2s}_{\ell+1,j}\}_{j=1}^{2^{-\ell-1}N}$ corresponding to $\{\mathbf{x}^{2s-1}_{\ell+1,j}\}_{j=1}^{2^{-\ell-1}N}$ and $\{\mathbf{x}^{2s}_{\ell+1,j}\}_{j=1}^{2^{-\ell-1}N}$, respectively. However, here we filter the rows of the result with a normalized filter $\bar{\mathbf{h}}_a^T$, which satisfies

$$\bar{h}_a[n] = |\bar{h}[n]| / \sum_k |\bar{h}[k]|, \tag{5}$$

so as to guarantee no drift in the coordinates.

We note that every time we advance from one scale to the next, the number of features points used to calculate a single reordering operator decreases by a factor of two. Therefore, when we move to the coarser level we increase both dimensions of the search area by a factor of $\sqrt{2}$, so as to keep the number of candidate neighbors to consider. Thus, the size of the search area used in the ℓth scale is $2^{\frac{\ell}{2}}B \times 2^{\frac{\ell}{2}}B$.

Having calculated the feature points $\mathbf{x}^s_{\ell+1,j}$ and their coordinates $\mathbf{q}^s_{\ell+1,j}$, they can be fed to the approximate shortest path search method described above to obtain each of the operators $\mathbf{P}^s_{\ell+1}$ in the $\ell+1$th scale. The calculation scheme of all the ordering operators $\mathbf{P}^s_\ell$ that are used in an L-level RTBWT decomposition is summarized in Algorithm 3. We note that Similarly to the decomposition scheme of the generalized tree-based wavelet transform (GTBWT) described in [12], the relation between the feature points in a full decomposition can be described using tree-like structures. Each such "generalized" tree contains all the feature points which have participated in the calculation of a single feature point $\mathbf{x}^s_{L,j}$ from the coarsest scale. Figure 3 shows an example of such a "generalized" tree, corresponding to length $N = 8$, using a filter $\bar{\mathbf{h}}$ of length 4 and disregarding boundary issues in the different levels. We next describe an alternative implementation of the RTBWT, and use it to show the frame properties of the transform.

Task: calculate all the operators $\{\mathbf{P}^s_\ell\}_{s=1}^{2^\ell}$, $\ell = 0 \dots L-1$, used in an L-level RTBWT decomposition.
Parameters: The image patches $\{\mathbf{x}_j\}_{j=1}^N$, their 2D coordinates $\{\mathbf{q}_j\}_{j=1}^N$, the distance function w, the filters $\bar{\mathbf{h}}$ and $\bar{\mathbf{h}}_a$, and the search area size B.
Initialization: Set $\mathbf{x}^1_{0,j} = \mathbf{x}_j$ and $\mathbf{q}^1_{0,j} = \mathbf{q}_j$.
Main Iteration: Perform the following steps for $\ell = 0, \dots, L-1$:
Set $\tilde{B} = 2^{\frac{\ell}{2}}B$ and $\tilde{N} = 2^{-\ell}N$.

- Perform the following steps for $s = 1, \dots, 2^\ell$:
  Calculate $\mathbf{P}^s_\ell$:
  - Choose a random index $1 \le j \le \tilde{N}$ and set $\Omega(1) = \{j\}$.
  - Perform the following steps for $k = 1, \dots, \tilde{N}-1$:
    - Set A_k to be the set of indices of the coordinates $\mathbf{q}^s_{\ell,j}$ which reside inside a $\tilde{B} \times \tilde{B}$ neighborhood around $\mathbf{q}^s_{\ell,\Omega(k)}$.
    - If $|A_k \setminus \Omega| > 0$, find $\mathbf{x}^s_{\ell,j_1}$ – the minimizer of $w(\mathbf{x}^s_{\ell,j}, \mathbf{x}^s_{\ell,\Omega(k)})$ such that $j_1 \in A_k$ and $j_1 \notin \Omega$.
    - Else, find $\mathbf{x}^s_{\ell,j_1}$ – the minimizer of $w(\mathbf{x}^s_{\ell,j}, \mathbf{x}^s_{\ell,\Omega(k)})$ such that $j_1 \notin \Omega$.
    - Set $\Omega(k+1)$ to be j_1.
  - The set Ω holds the ordering applied by $\mathbf{P}^s_\ell$.

  Calculate $\ell+1$th scale feature points:
  - Order the feature points $\{\mathbf{x}^s_{\ell,j}\}_{j=1}^{\tilde{N}}$ according to the permutation defined by $\mathbf{P}^s_\ell$.
  - Place the reordered points in the columns of a matrix $\mathbf{X}^{s,p}_\ell$.
  - Apply the filter $\bar{\mathbf{h}}^T$ to the rows of $\mathbf{X}^{s,p}_\ell$ and
    - keep the odd columns of the result so as to obtain the points $\{\mathbf{x}^{2s-1}_{\ell+1,j}\}_{j=1}^{\frac{\tilde{N}}{2}}$.
    - keep the even columns of the result so as to obtain the points $\{\mathbf{x}^{2s}_{\ell+1,j}\}_{j=1}^{\frac{\tilde{N}}{2}}$.

  Calculate $\ell+1$th scale coordinates:
  - Order the coordinates $\{\mathbf{q}^s_{\ell,j}\}_{j=1}^{\tilde{N}}$ according to the permutation defined by $\mathbf{P}^s_\ell$.
  - Place the reordered points in the columns of a matrix $\mathbf{Q}^{s,p}_\ell$.
  - Apply the filter $\bar{\mathbf{h}}_a^T$ to the rows of $\mathbf{Q}^{s,p}_\ell$ and
    - keep the odd columns of the result so as to obtain the points $\{\mathbf{q}^{2s-1}_{\ell+1,j}\}_{j=1}^{\frac{\tilde{N}}{2}}$.
    - keep the even columns of the result so as to obtain the points $\{\mathbf{q}^{2s}_{\ell+1,j}\}_{j=1}^{\frac{\tilde{N}}{2}}$.

Output: The operators $\{\mathbf{P}^s_\ell\}_{s=1}^{2^\ell}$, $\ell = 0 \dots L-1$.

Algorithm 3: Calculation of all the reordering operators used in an L-level RTBWT decomposition.

C. À Trous Implementation and Frame Properties

The common and widely known "à trous" algorithm [16], [17] is an alternative implementation for the redundant wavelet transform. Instead of explicitly applying the filters $\bar{\mathbf{h}}$ and $\bar{\mathbf{g}}$ to subsets of the signals $\mathbf{a}_\ell$, the à trous algorithm applies upsampled versions of these filters directly to $\mathbf{a}_\ell$. The upsampled filters are obtained by inserting "holes" (trous in French) between nonzero filter taps. We next show that the RTBWT can also be applied using a modification of the à trous algorithm. We then use this scheme to show that the RTBWT is a frame expansion, with the same bounds as the common redundant wavelet transform. We note that the à trous implementation and the frame properties of the RTBWT are valid in general, and are not restricted to the image-derived RTBWT.

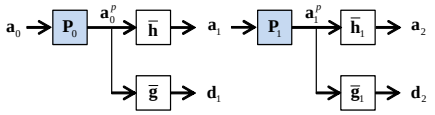

Fig. 4: RTBWT à trous based decomposition scheme.

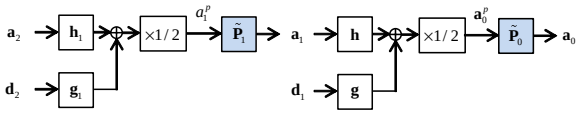

Fig. 5: RTBWT à trous based reconstruction scheme.

We first notice that in the aforementioned implementation of the RTBWT decomposition scheme, the operators $\mathbf{P}_\ell^s$ operate in the ℓth scale on disjoint subvectors $\mathbf{a}_\ell^s$ of $\mathbf{a}_\ell$. Therefore, these operators can be replaced by a single operator $\mathbf{P}_\ell$ which operates on $\mathbf{a}_\ell$, and produces the reordered signal $\mathbf{a}_\ell^p$. Let $\mathbf{R}_\ell^s$ be a matrix which extracts the signal $\mathbf{a}_\ell^s$ from the signal $\mathbf{a}_\ell$. Then we construct the permuted signal $\mathbf{a}_\ell^p$ by extracting all the signals $\{\mathbf{a}_\ell^s\}_{s=1}^{2^\ell}$ from $\mathbf{a}_\ell$, applying to each signal $\mathbf{a}_\ell^s$ the corresponding operator $\mathbf{P}_\ell^s$, and returning all the permuted signals back to their original location. More specifically,

$$\mathbf{a}_\ell^p = \sum_{s=1}^{2^{\ell-1}} (\mathbf{R}_\ell^s)^T \mathbf{P}_\ell^s \mathbf{R}_\ell^s \mathbf{a}_\ell = \mathbf{P}_\ell \mathbf{a}_\ell \tag{6}$$

where the matrices $(\mathbf{R}_\ell^s)^T$ return the permuted signals back to their locations. This way we have defined the permutation matrix

$$\mathbf{P}_\ell = \sum_{s=1}^{2^{\ell-1}} (\mathbf{R}_\ell^s)^T \mathbf{P}_\ell^s \mathbf{R}_\ell^s. \tag{7}$$

We can use the operators $\mathbf{P}_\ell$ to modify the à trous algorithm, and obtain the à trous implementation of the RTBWT decomposition scheme. Let $\bar{\mathbf{h}}_\ell$ be a filter obtained from $\bar{\mathbf{h}}$ by inserting $2^\ell - 1$ zeros between each sample, where $\bar{\mathbf{h}}_0 = \bar{\mathbf{h}}$. Also, let $\bar{\mathbf{g}}_\ell$ be a filter obtained from $\bar{\mathbf{g}}$ in a similar manner. Both filters are applied using cyclic convolution. Then assuming that the operators $\mathbf{P}_\ell$ are known, we can apply the RTBWT by repeating for $\ell = 0 \dots L-1$ the following filter bank operations, described in Figure 4,

$$\mathbf{a}_\ell^p = \mathbf{P}_\ell \mathbf{a}_\ell \tag{8}$$

$$a_{\ell+1}[n] = a_\ell^p[n] * \bar{h}_\ell[n] \tag{9}$$

$$d_{\ell+1}[n] = a_\ell^p[n] * \bar{g}_\ell[n]. \tag{10}$$

We note that when the operators $\mathbf{P}_\ell$ are removed, our scheme coincides with the common à trous algorithm.

In a similar manner, the operators $\tilde{P}_\ell^s = (\mathbf{P}_\ell^s)^{-1}$, which operate on the signals $\mathbf{a}_\ell^{s,p}$ can also be replaced by a single operator $\tilde{P}_\ell = (\mathbf{P}_\ell)^{-1}$. We use the operators $\tilde{P}_\ell$ to construct the à trous implementation of the RTBWT reconstruction scheme. Let $\mathbf{h}_\ell$ be a filter obtained from $\mathbf{h}$ by inserting $2^\ell - 1$ zeros between each sample, where $\mathbf{h}_0 = \mathbf{h}$. Also, let $\mathbf{g}_\ell$ be a filter obtained from $\mathbf{g}$ in a similar manner. Both filters are applied using cyclic convolution. Then assuming that the operators $\tilde{P}_\ell$ are known, we can apply the RTBWT reconstruction scheme by repeating for $\ell = L \dots 1$ the following filter bank operations, described in Figure 5,

$$a_\ell^p[n] = \frac{1}{2}\left(a_{\ell+1}[n] * h_\ell[n] + d_{\ell+1}[n] * g_\ell[n]\right) \tag{11}$$

$$\mathbf{a}_\ell = \tilde{\mathbf{P}}_\ell \mathbf{a}_\ell^p. \tag{12}$$

D. Frame Properties

Let $\mathbf{\Phi}$ denote an $(L+1)N \times N$ transform matrix, which applies an L-level RTBWT decomposition. We recall that a sequence $\{\phi_n\}$ is a frame if there exist two constants $\beta \geq \alpha > 0$ such that for all $\mathbf{y}$

$$\alpha\|\mathbf{y}\|^2 \leq \sum_n |\langle \phi_n, \mathbf{y}\rangle|^2 \leq \beta\|\mathbf{y}\|^2. \tag{13}$$

The à trous implementation of the RTBWT will enable us to show that the rows of $\mathbf{\Phi}$ constitute a frame $\{\phi_n\}$ in $\mathbb{R}^N$, with the same frame bounds as the common redundant wavelet transform [18].

Proposition 1: Let $\mathbf{c} = \mathbf{\Phi}\mathbf{y} = [\mathbf{a}_L^T, \mathbf{d}_L^T, \dots, \mathbf{d}_1^T]^T$ denote an L-level RTBWT decomposition of a signal $\mathbf{y}$. Then $\mathbf{c}$ is a frame expansion with frame bounds $\alpha = 2$ and $\beta = 2^L$, and these bounds are the tightest possible.

The proof is presented in Appendix A.

Also, let $\mathbf{\Psi}$ denote an $N \times (L+1)N$ matrix that applies the L-level RTBWT reconstruction. $\mathbf{\Phi}$ and $\mathbf{\Psi}$ apply the RTBWT analysis and synthesis operations, respectively, and $\mathbf{\Psi}\mathbf{\Phi} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix. Thus $\mathbf{\Psi}$ is the pseudo inverse of $\mathbf{\Phi}$, and the columns of $\mathbf{\Psi}$ constitute a frame $\{\psi_n\}$ which is dual to $\{\phi_n\}$, i.e. it satisfies

$$\frac{1}{2^L}\|\mathbf{y}\|^2 \leq \sum_n |\langle \psi_n, \mathbf{y}\rangle|^2 \leq \frac{1}{2}\|\mathbf{y}\|^2. \tag{14}$$

III. RTBWT Subimage Averaging Frames

Once the RTBWT is calculated for an image, we can apply this transform to that image, and then process it in the transform domain. However, we observed in [1] that improved results are obtained when a subimage averaging scheme is used instead. This scheme consists of applying the RTBWT to different subimages of the treated image and processing each subimage in the transform domain. Then the reconstructed image is obtained by plugging each processed subimage into its original place in the image canvas, and averaging the different values obtained for each pixel. As we show next, we can describe the transform coefficients of the different subimages as a single extended representation of the image. We construct matrices that act as analysis and synthesis operators, which are used to obtain this extended representation and reconstruct the image from it. We also show that the rows and columns of the analysis and synthesis matrices, respectively, constitute a frame and its dual.

Let $N_p = (N_1 - \sqrt{n}+1)(N_2 - \sqrt{n}+1)$ denote the number of overlapped patches in the image $\mathbf{Y}$, and let $\mathbf{X}$ be an $n \times N_p$ matrix, containing column stacked versions of these patches. We extract these patches column by column, starting from the top left-most patch. When the RTBWT was constructed

in the previous section, it was assumed that each patch is associated only with its middle pixel. Therefore the transform was designed to efficiently represent the signal composed of the middle points in the patches, that reside in the middle row of $\mathbf{X}$. However, we can alternatively choose to associate all the patches with a pixel located in a different position, e.g., the top left pixel in each patch. This means that the transform can be used to represent any one of the signals located in the rows of $\mathbf{X}$. These signals are the column stacked versions of all the n subimages of size $(N_1 - \sqrt{n}+1) \times (N_2 - \sqrt{n}+1)$ contained in the image $\mathbf{Y}$. We denote these subimages by $\tilde{\mathbf{Y}}_j, j = 1, 2, \ldots, n$.

Let $\mathbf{\Phi}$ denote the RTBWT transform matrix, here it is of size $(L+1)N_p \times N_p$, applying an L-level decomposition. Also, let the vector $\tilde{\mathbf{y}}_j = \mathbf{R}_j\mathbf{y}$ of length N_p be the column stacked version of $\tilde{\mathbf{Y}}_j$, where the $N_p \times N$ matrix $\mathbf{R}_j$ extracts the jth subimage from the image $\mathbf{y}$. We first apply the RTBWT to each of the n subimages $\tilde{\mathbf{y}}_j$ and obtain the n vectors

$$\mathbf{c}_j = \mathbf{\Phi}\tilde{\mathbf{y}}_j = \mathbf{\Phi}\mathbf{R}_j\mathbf{y}. \tag{15}$$

We obtain the extended representation vector $\mathbf{c}_{SA}$ by concatenating all these vectors into a single column

$$\mathbf{c} = \left[\mathbf{c}_1^T, ..., \mathbf{c}_n^T\right]^T = \begin{bmatrix} \mathbf{\Phi}\mathbf{R}_1 \\ \vdots \\ \mathbf{\Phi}\mathbf{R}_n \end{bmatrix} \mathbf{y} = \mathbf{\Phi}_{SA}\mathbf{y} \tag{16}$$

where we defined the matrix

$$\mathbf{\Phi}_{SA} = \begin{bmatrix} \mathbf{\Phi}\mathbf{R}_1 \\ \vdots \\ \mathbf{\Phi}\mathbf{R}_n \end{bmatrix}. \tag{17}$$

This matrix applies the analysis operator used to obtain the representation $\mathbf{c}_{SA}$.

The image $\mathbf{y}$ is reconstructed from the representation $\mathbf{c}$ in the following manner. Let $\mathbf{\Psi}$ denote the $(L+1)N_p \times N_p$ matrix that applies the L-level RTBWT reconstruction. We apply the matrix $\mathbf{\Psi}$ to each of the vectors $\mathbf{c}_j$, plug each of the obtained subimages into its original place in the image, and average the different values obtained for each pixel. More formally,

$$\begin{aligned} \mathbf{y} &= \mathbf{D}^{-1}\sum_{j=1}^{n}\mathbf{R}_j^T\mathbf{\Psi}\mathbf{c}_j \\ &= \mathbf{D}^{-1}\left[\mathbf{R}_1^T\mathbf{\Psi}, \ldots, \mathbf{R}_n^T\mathbf{\Psi}\right]\mathbf{c} = \mathbf{\Psi}_{SA}\mathbf{c}, \end{aligned} \tag{18}$$

where the matrix $\mathbf{R}_j^T$ returns the jth subimage into its original place in the image, and $\mathbf{D} = \sum_{j=1}^{n}\mathbf{R}_j^T\mathbf{R}_j$ is a diagonal weight matrix. The matrix

$$\mathbf{\Psi}_{SA} = \mathbf{D}^{-1}\left[\mathbf{R}_1^T\mathbf{\Psi}, \ldots, \mathbf{R}_n^T\mathbf{\Psi}\right] \tag{19}$$

applies the synthesis operator used to reconstruct $\mathbf{y}$ from the representation $\mathbf{c}$. As we see next, the rows of $\mathbf{\Phi}^{SA}$ constitute a frame $\{\phi_n^{SA}\}$ and the columns of $\mathbf{\Psi}^{SA}$ constitute its dual $\{\psi_n^{SA}\}$.

Proposition 2: The extended representation $\mathbf{c}^{SA}$ is a frame expansion with frame bounds $\alpha = 2$ and $\beta = 2^L n$.
The proof is presented in Appendix B.

Here we also see that $\mathbf{\Phi}^{SA}$ and $\mathbf{\Psi}^{SA}$ apply analysis and synthesis operations, and $\mathbf{\Psi}^{SA}\mathbf{\Phi}^{SA} = \mathbf{I}$. Therefore, $\mathbf{\Psi}^{SA}$ is the pseudo-inverse of $\mathbf{\Phi}^{SA}$, and the columns of $\mathbf{\Psi}^{SA}$ constitute a frame $\{\psi_n^{SA}\}$ which is dual to $\{\phi_n^{SA}\}$, i.e. it satisfies

$$\frac{1}{2^L n}\|\mathbf{y}\|^2 \leq \sum_n |\langle\psi_n, \mathbf{y}\rangle|^2 \leq \frac{1}{2}\|\mathbf{y}\|^2. \tag{20}$$

Before concluding this section, we add a remark regarding the frame bounds calculated above. Let us assume that the RTBWT and corresponding frame have been calculated from an image $\mathbf{y}_0$. Then the transform is adaptive to the content in this image, since it is designed by ordering of its patches or their descendants (i.e. filtered patches), and so does the corresponding frame. Therefore, the frame bounds obtained above are in fact misleading, since inequalities such as (20) are true for a general image $\mathbf{y}$, but the frame was designed to handle the image $\mathbf{y}_0$. Thus, perhaps a better definition of the frame bounds would be one that has to hold true for all the images $\mathbf{y}$ in a small sphere $\|\mathbf{y} - \mathbf{y}_0\|_2 \leq \epsilon$, for which the transform (and the resulting frame) retain its topology. In such a case, it is quite likely that the actual frame obtained is nearly tight. We leave this matter open at this stage, as our next part of the paper does not rely directly on these frame properties.

IV. Image Reconstruction using RTBWT SA Frames

A. Image Reconstruction Scheme

Let $\mathbf{Y}$ be an image of size $N_1 \times N_2$ where $N_1 N_2 = N$, and let $\mathbf{Z}$ be its corrupted version. Also, let $\mathbf{z}$ and $\mathbf{y}$ be the column stacked versions of $\mathbf{Z}$ and $\mathbf{Y}$, respectively. Then we assume that the corrupted image is obtained via

$$\mathbf{z} = \mathbf{H}\mathbf{y} + \mathbf{v}, \tag{21}$$

where the $N \times N$ matrix $\mathbf{H}$ denotes a linear operator that corrupts the data, and $\mathbf{v}$ denotes an additive white Gaussian noise independent of $\mathbf{y}$ with zero mean and variance σ^2. This setting can be used to describe several of the classic image inverse problems such as image denoising, deblurring, and inpainting. Our goal is to reconstruct $\mathbf{y}$ from $\mathbf{z}$ by optimizing an objective function that uses the analysis and synthesis operators[1] $\mathbf{\Phi}_{SA}$ and $\mathbf{\Psi}_{SA}$ as sparsity promoting regularizers.

A common approach for image reconstruction is to solve an inverse problem which uses either an analysis-based or synthesis-based image priors [19]. Here we take a different approach, follow the footsteps of [11], and consider the following combination of an analysis and a synthesis problems

$$\begin{aligned} \{\hat{\mathbf{y}}, \hat{\mathbf{c}}\} = \underset{\mathbf{y},\mathbf{c}}{\operatorname{argmin}} \; &\|\mathbf{z} - \mathbf{H}\mathbf{y}\|_2^2 + \eta\|\mathbf{y} - \mathbf{\Psi}\mathbf{c}\|_2^2 \\ &+ \lambda\|\mathbf{c}\|_0 + \mu\|\mathbf{c} - \mathbf{\Phi}\mathbf{y}\|_2^2. \end{aligned} \tag{22}$$

Indeed, it can be seen that the problem (22) reduces to a synthesis problem when $\eta \to \infty$ and $\mu = 0$, and to an analysis when $\eta = 0$ and $\mu \to \infty$. The authors of [11] brilliantly handle

[1] As of this point in the paper, we will simplify our notations and use $\mathbf{\Phi}$ and $\mathbf{\Psi}$, referring to the full adaptive frames $\mathbf{\Phi}_{SA}$ and $\mathbf{\Psi}_{SA}$.

the problem (22) as a generalized Nash equilibrium (GNE) process, and split it into two subproblems, a denoising task:

$$\hat{\mathbf{c}} = \underset{\mathbf{c}}{\operatorname{argmin}} \ \frac{\lambda}{\mu}\|\mathbf{c}\|_0 + \|\mathbf{c} - \mathbf{\Phi}\mathbf{y}\|_2^2, \tag{23}$$

and an inversion task:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmin}} \ \|\mathbf{z} - \mathbf{H}\mathbf{y}\|_2^2 + \eta\|\mathbf{y} - \mathbf{\Psi}\mathbf{c}\|_2^2 \tag{24}$$

which are solved sequentially. We note that $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are calculated in the beginning of this iterative process, and one approach is to keep them fixed throughout the iterative process. Alternatively, one might update these operators based on the updated image. Unless said otherwise, our scheme takes the first path of fixing these frames during the iterations.

The solution of the inversion stage is simply

$$\hat{\mathbf{y}} = \left[\mathbf{H}^T\mathbf{H} + \eta\mathbf{I}\right]^{-1} \times \left[\mathbf{H}^T\mathbf{z} + \eta\mathbf{\Psi}\mathbf{c}\right]. \tag{25}$$

The solution of the denoising problem is also quite simple, given by

$$\hat{\mathbf{c}} = S^{hard}\left\{\mathbf{\Phi}\mathbf{y}\right\} \tag{26}$$

where the operator S^{hard} applies hard thresholding with the threshold $\sqrt{\lambda/\mu}$. We choose to replace S^{hard} with an operator S_τ that sets to zero coefficients in $\mathbf{c}$ in such a manner that the coefficient vectors $\mathbf{c}_j$ (see Equation (15)) share a common support. This way, the set of thresholded subimage coefficients form a joint sparsity pattern in the transform domain, and our experiments show that when such patterns are used, the quality of the reconstructed images improves. Let $c_j[k]$ be the kth sample in the coefficient vector $\mathbf{c}_j$. Then the operator S_τ goes over the indices $k = 1\ldots,(L+1)N_p$, and for each index sets the coefficients $\{c_1[k],\ldots,c_n[k]\}$ to zero if

$$\sqrt{\frac{1}{n}\sum_{j=1}^{n} c_j^2[k]} < \tau\sigma \tag{27}$$

where τ is a design parameter. This corresponds to a simple modification in the original inverse problem formulation in Equation (22), replacing the term $\|\mathbf{c}\|_0$ by a mixed norm applied on the different pieces of this representation, namely, $\|\mathbf{c}\|_{2,0}$.

The obtained image reconstruction scheme is described in Algorithm 4. It can be seen that this algorithm is similar to the IDD-BM3D algorithm proposed in [11], to which we will later compare our image deblurring results. We next demonstrate the use of the scheme in Algorithm 4 for image denoising and deblurring. In all the experiments described next we choose the distance function w to be the Euclidean distance, and use a 9-level RTBWT decomposition with the Symmlet 8 wavelet filter.

Task: recover the image $\mathbf{y}$ from the corrupted image $\mathbf{z}$.
Parameters: We are given the corrupted image $\mathbf{z}$, an initial estimate $\mathbf{y}_{init}$, the matrix $\mathbf{H}$, the iteration number G, the threshold τ, and the design parameter η.
Initialization: Set $\mathbf{y}_0 = \mathbf{y}_{init}$ and use it to calculate $\mathbf{\Phi}$ and $\mathbf{\Psi}$.
Main Iteration: For $t = 1,\ldots,G$ perform:

- Denoising: $\mathbf{c}_t = S_\tau\{\mathbf{\Phi}\mathbf{y}_{t-1}\}$
- Inversion: $\mathbf{y}_t = \left[\mathbf{H}^T\mathbf{H} + \eta\mathbf{I}\right]^{-1}\left[\mathbf{H}^T\mathbf{z} + \eta\mathbf{\Psi}\mathbf{c}_t\right]$

Output: The estimate $\hat{\mathbf{y}} = \mathbf{y}_G$.

Algorithm 4: Image reconstruction scheme.

B. Image Denoising

In the case of image denoising $\mathbf{z} = \mathbf{y} + \mathbf{v}$ and therefore $\mathbf{H} = \mathbf{I}$. We perform denoising using a simplified version of the scheme in Algorithm 4, which uses the initial estimate $\mathbf{y}_{init} = \mathbf{z}$, and applies only one iteration. Also, we notice that since $\mathbf{H} = \mathbf{I}$, the solution to the inversion problem (24) becomes:

$$\hat{\mathbf{y}} = \frac{1}{1+\eta}\mathbf{z} + \frac{\eta}{1+\eta}\mathbf{\Psi}\mathbf{c}. \tag{28}$$

which is a weighted average between a denoised image and the original noisy image. We further simplify our scheme and cancel the addition of the noisy data to the clean image by setting $\eta \to \infty$. All these simplifications were done since we found experimentally that the resulting simplified algorithm is already producing near state-of-the-art results. Thus, the obtained denoising scheme consists of calculating $\mathbf{c} = \mathbf{\Phi}_{SA}\mathbf{z}$, applying to it the thresholding operator S_τ, and reconstructing the image using $\mathbf{\Psi}_{SA}$, i.e.

$$\hat{\mathbf{y}} = \mathbf{\Psi}S_\tau\{\mathbf{c}\} = \mathbf{\Psi}S_\tau\{\mathbf{\Phi}\mathbf{z}\}. \tag{29}$$

We explore two different methods to further improve the results obtained with the denoising algorithm described above. Both these methods apply the above scheme, and then use the patches from the "cleaned" result $\hat{\mathbf{y}}^1$ to construct a "better version" of the RTBWT, and use this transform to calculate new analysis and synthesis operators $\mathbf{\Phi}^1$ and $\mathbf{\Psi}^1$. The first method obtains the denoised image $\hat{\mathbf{y}}^2$ by applying Equation (29) again with these modified operators, i.e.

$$\hat{\mathbf{y}}^2 = \mathbf{\Psi}^1 S_\tau\{\mathbf{\Phi}^1\mathbf{z}\}. \tag{30}$$

The second method applies a different scheme in its second stage. Let $\mathbf{C}_{diag}$ be a diagonal matrix that contains the vector $\mathbf{\Phi}^1\hat{\mathbf{y}}^1$ in its main diagonal. Then we replace the ℓ_0-norm term $\|\mathbf{c}\|_0$ in the denoising problem (23) with a different sparsity promoting term $\mathbf{c}^T\mathbf{C}_{diag}^{-1}\mathbf{c}$, and obtain the new problem

$$\hat{\mathbf{c}} = \underset{\mathbf{c}}{\operatorname{argmin}} \ \frac{\lambda}{\mu}\mathbf{c}^T\mathbf{C}_{diag}^{-1}\mathbf{c} + \|\mathbf{c} - \mathbf{\Phi}\mathbf{y}\|_2^2. \tag{31}$$

The solution to this problem is

$$\hat{\mathbf{c}} = \left[\mathbf{C}_{diag} + \frac{\lambda}{\mu}\mathbf{I}\right]^{-1}\mathbf{C}_{diag}\mathbf{\Phi}^1\mathbf{y} = \mathbf{W}\mathbf{y}, \tag{32}$$

where we have defined the diagonal matrix

$$\mathbf{W} = \left[\mathbf{C}_{diag} + \frac{\lambda}{\mu}\mathbf{I}\right]^{-1}\mathbf{C}_{diag}. \tag{33}$$

Thus, the second method obtains the denoised image $\hat{\mathbf{y}}_w^2$ by calculating in its second stage the coefficient vector $\mathbf{c} = \mathbf{\Phi}^1\mathbf{z}$,

TABLE I: Parameters Used in the Denoising Experiments.

σ	Stage	L	$\sqrt{n}$	B	τ
5	1	9	7	21	1.5
	2	9	4	211	1.45
	Wiener	9	5	191	1.05
10	1	9	9	21	1.5
	2	9	6	181	1.45
	Wiener	9	5	191	1.05
15	1	9	10	21	1.5
	2	9	7	211	1.45
	Wiener	9	5	191	1.05
20	1	9	11	21	1.5
	2	9	9	211	1.45
	Wiener	9	5	191	1.05
25	1	9	12	21	1.5
	2	9	10	211	1.45
	Wiener	9	6	191	1.05
50	1	9	14	21	1.5
	2	9	13	211	1.45
	Wiener	9	7	191	1.05
75	1	9	16	21	1.5
	2	9	14	211	1.45
	Wiener	9	7	191	1.05
100	1	9	16	21	1.5
	2	9	16	211	1.45
	Wiener	9	9	191	1.05

multiplying it with the matrix $\mathbf{W}$, and reconstructing the image using $\mathbf{\Psi}^1$, i.e.

$$\hat{\mathbf{y}}_w^2 = \mathbf{\Psi}^1 \mathbf{W} \mathbf{\Phi}^1 \mathbf{z}. \tag{34}$$

We note that by choosing $\lambda/\mu = (\tau\sigma)^2$, where τ is a design parameter, multiplication by the matrix $\mathbf{W}$ is equivalent to applying some sort of a Wiener filter to the noisy image, similarly to what is done in the second stage of the algorithms in [9] and [20].

In order to assess the performance of the proposed image denoising scheme we apply it to noisy versions of 6 images, with 8 different noise standard deviations. The parameters employed by the proposed denoising scheme for the different noise levels are shown in Table I. Table II shows the PSNR values of the results obtained with the BM3D algorithm, two stages of the RTBWT denoising scheme, and the RTBWT followed by Wiener filtering. The noisy and recovered images obtained for $\sigma = 25$ with the BM3D algorithm, and our results are shown in Figure 6. First, it can be seen that both the second stage of the RTBWT scheme and the Wiener filter improve the first iteration results of the RTBWT scheme in many $(70\% - 85\%)$ of the cases. The Wiener filter results are generally better or comparable to the results of both the first and second stages of the RTBWT scheme. Finally it can be seen that for $\sigma \geq 25$, the RTBWT denoising scheme followed by Wiener filtering achieves in most of the cases either the best results or results that are nearly as good. As the noise decreases in strength, the performance of the RTBWT based denoising schemes deteriorates compared to the BM3D.

C. Image Deblurring

In the case of image deblurring, $\mathbf{H}$ is a blur matrix, and we perform deblurring by simply applying Algorithm 4 as is. We demonstrate the image deblurring performance obtained with this algorithm on the images Lena, Barbara, House, and Cameraman, for the 6 scenarios described in Table III.

TABLE III: Blur Point Spread Functions (PSF) and Noise Variances used in the Different Deblurring Scenarios.

Scenario	PSF	σ^2
1	$1/(1+x_1^2+x_2^2)$, $x_1, x_2 = -7, \ldots, 7$	2
2	$1/(1+x_1^2+x_2^2)$, $x_1, x_2 = -7, \ldots, 7$	8
3	9×9 uniform	≈ 0.3
4	[1 4 6 4 1]^T[1 4 6 4 1] / 256	49
5	Gaussian with $std = 1.6$	4
6	Gaussian with $std = 0.4$	64

We compare the results obtained with this algorithm to the ones obtained with the BM3DDEB [10] and the IDD-BM3D algorithms. As the IDD-BM3D algorithm is initialized with the BM3DDEB results, for a fair comparison we examine the results obtained with our scheme using the same initialization. We also examine the results obtained with our scheme by initializing it with the blurry image, and try to improve them by applying two more stages of our scheme, where we initialize each stage with result of the previous ones. The parameters employed in the different stages of the proposed deblurring scheme, with the different initializations and for the different scenarios, are shown in Table IV. We note that similarly to what was done in [11], we optimized the parameters γ and τ separately for each stage and each deblurring scenario to provide best reconstruction quality.

Table V shows the ISNR results obtained with the different algorithms. The blurred and recovered images obtained with the BM3DDEB and IDD-BM3D algorithms, our proposed scheme initialized with the BM3DDEB results, and three stages of the proposed scheme initialized with the blurry images, are shown in Figure 7. It can be seen that for all the scenarios except for scenario 3, the proposed scheme initialized with BM3DDEB results achieves the best performance for every image. Our scheme obtains inferior results when the blurry image is used to initialize it, however its performance improve when two more rounds are applied. In fact, in some cases three rounds of our scheme initialized with the blurry image obtain the best or second best results.

D. Computational Complexity

We next evaluate the computational complexity of the image denoising and deblurring algorithms described above. We start by calculating the complexity of a single iteration of the denoising algorithm. First, extracting all the overlapped image patches requires $O(nN)$ operations. Next, we assume that for the calculation of each one of the 2^ℓ operators $\mathbf{P}_\ell^s$ in the ℓth scale of the RTBWT, each patch requires approximately B^2 distance calculations. As calculating the Euclidean distance between two patches requires $O(n)$ operations, the number of operations required to calculate a single reordering operator $\mathbf{P}_\ell^s$ from $\tilde{N}_\ell = N2^{-\ell}$ patches is approximately $O(\tilde{N}_\ell B^2 n)$. Reordering these patches and applying them the two wavelet filters requires $O(n\tilde{N}_\ell)$ operations. Therefore the number of operations required in order to calculate from an image all the

TABLE II: Denoising Results (PSNR in dB) of Noisy Versions of 6 Images, Obtained with the BM3D Algorithm, Two Stages of the RTBWT Denoising Scheme, and the RTBWT Scheme Followed by Wiener Filtering. For Each Image and Noise Level the Best Result and Results within a Distance of 0.05 dB from It, are Highlighted.

Image	Method	σ/PSNR							
		5/34.16	10/28.14	15/24.61	20/22.11	25/20.18	50/14.16	75/10.63	100/8.14
Lena	BM3D	**38.72**	**35.93**	**34.27**	33.05	32.08	29.05	27.26	25.95
	proposed (1 stage)	38.44	35.70	34.12	32.98	32.06	28.97	27.19	25.90
	proposed (2 stages)	38.08	35.45	33.91	32.99	32.11	29.18	27.43	**26.31**
	proposed (1 stage+Wiener)	38.40	35.75	**34.22**	**33.11**	**32.26**	**29.30**	**27.50**	**26.36**
Barbara	BM3D	**38.31**	**34.98**	**33.11**	31.78	30.72	27.23	25.12	23.62
	proposed (1 stage)	37.54	34.50	32.89	31.70	30.73	27.39	25.38	23.97
	proposed (2 stages)	37.62	34.55	32.91	31.71	30.76	27.65	25.73	**24.44**
	proposed (1 stage+Wiener)	37.67	34.55	32.97	**31.85**	**30.90**	**27.78**	**25.82**	**24.46**
Boats	BM3D	**37.28**	**33.92**	**32.14**	**30.88**	**29.91**	26.78	**25.12**	23.97
	proposed (1 stage)	36.86	33.68	31.91	30.65	29.67	26.59	24.85	23.68
	proposed (2 stages)	36.86	33.64	31.92	30.67	29.71	26.80	**25.13**	**24.03**
	proposed (1 stage+Wiener)	36.98	33.79	32.05	30.82	**29.88**	**26.91**	**25.15**	**24.04**
Fingerprint	BM3D	**36.51**	**32.46**	**30.28**	**28.81**	**27.70**	**24.53**	**22.83**	**21.61**
	proposed (1 stage)	35.03	31.28	29.47	28.21	27.16	23.90	22.20	21.19
	proposed (2 stages)	35.86	31.88	29.80	28.17	27.01	23.74	22.22	21.37
	proposed (1 stage+Wiener)	35.38	31.60	29.73	28.45	27.32	24.06	22.47	21.53
House	BM3D	**39.83**	**36.71**	**34.94**	**33.77**	**32.86**	**29.69**	**27.51**	25.87
	proposed (1 stage)	39.32	36.41	34.75	33.61	32.74	29.55	27.43	25.75
	proposed (2 stages)	38.40	35.68	34.01	33.27	32.46	29.49	27.42	**25.96**
	proposed (1 stage+Wiener)	38.42	35.86	34.25	33.08	32.37	29.56	27.37	**25.98**
Peppers	BM3D	**38.12**	**34.68**	**32.70**	31.29	30.16	26.68	24.73	23.39
	proposed (1 stage)	37.65	34.27	32.36	30.99	29.90	26.48	24.50	23.08
	proposed (2 stages)	37.64	34.44	32.61	31.21	30.14	26.76	24.89	**23.53**
	proposed (1 stage+Wiener)	37.81	34.49	**32.66**	**31.35**	**30.33**	**26.93**	**24.98**	**23.56**

Fig. 6: Denoising results (PSNR) for the images Lena and Barbara ($\sigma = 25$, input PSNR=20.18 dB): First column: noisy images, Second column - BM3D results, Third column - 2 stages results of the RTBWT denoising scheme, Fourth column - results of the RTBWT scheme followed by Wiener filtering.

operators $\mathbf{P}_\ell^s$ used in an L-level RTBWT is

$$O(nN) + \sum_{\ell=0}^{L-1} 2^\ell \left[O(\tilde{N}_\ell B^2 n) + O(n\tilde{N}_\ell) \right] = O(LNB^2 n). \tag{35}$$

Now, reordering each signal $\mathbf{a}_\ell^s$ and applying on it the two wavelet filters requires $O(\tilde{N}_\ell)$ operations, therefore applying either of the analysis and synthesis operators $\mathbf{\Phi}$ and $\mathbf{\Psi}$, corresponding to the RTBWT calculated above, requires $\sum_{\ell=0}^{L-1} 2^\ell O(\tilde{N}_\ell) = O(LN)$ operations.

Next, each of the following actions require $O(nLN)$ operations: applying the operator $\mathbf{\Phi}$ to the n subimages $\tilde{\mathbf{z}}_j$, applying the threshold operator S_τ or the Wiener filter to the transform coefficients, and applying $\mathbf{\Psi}$ to the result, Constructing an estimate image by averaging the pixel values obtained with the different subimages requires $O(nN)$ operations, therefore the total complexity of a single denoising iteration is

$$O(LNB^2 n) + O(nLN) + O(nN) = O(LNB^2 n) \tag{36}$$

operations, which means that, as might be expected, the overall complexity is dominated by the calculation of the RTBWT permutation operators. For a typical case in our experiments,

TABLE V: Deblurring Results (ISNR in dB) of Blurry Versions of 4 Images, Obtained with the BM3DDEB and IDD-BM3D Algorithms, the Proposed Scheme Initialized with the BM3DDEB Results, and 3 Stages of the Proposed Scheme Initialized with the Blurry Image (BI). For Each Image and Scenario the Best Result and Results within a Distance of 0.05 dB from It, are Highlighted.

Image	Method	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
Lena	BM3DDEB	7.96	6.55	7.99	4.80	4.34	6.42
	IDD-BM3D	7.97	6.61	8.91	4.97	4.85	6.34
	proposed (BM3DDEB init.)	**8.56**	**6.92**	8.86	**5.52**	**4.95**	**6.91**
	proposed (BI 1 iter.)	7.36	5.99	8.43	4.66	4.56	5.58
	proposed (BI 2 iter.)	7.43	6.25	8.69	4.76	4.83	5.89
	proposed (BI 3 iter.)	7.75	6.34	**8.98**	4.88	**4.91**	5.89
Barbara	BM3DDEB	7.88	4.13	5.91	2.05	1.29	5.85
	IDD-BM3D	7.64	3.96	**6.05**	1.88	1.16	5.45
	proposed (BM3DDEB init.)	**8.06**	**4.57**	**6.01**	**2.20**	**1.41**	**6.06**
	proposed (BI 1 iter.)	4.98	2.54	4.52	1.48	0.97	4.59
	proposed (BI 2 iter.)	5.98	2.68	4.76	1.48	0.99	4.88
	proposed (BI 3 iter.)	6.31	2.76	4.99	1.49	1.00	4.88
House	BM3DDEB	9.34	8.22	10.94	5.20	4.59	7.34
	IDD-BM3D	9.95	8.55	12.89	5.79	5.74	7.13
	proposed (BM3DDEB init.)	**10.44**	**8.79**	**13.11**	**6.38**	**5.95**	**7.56**
	proposed (BI 1 iter.)	9.09	7.58	12.03	5.27	5.30	6.16
	proposed (BI 2 iter.)	9.10	7.81	12.69	5.29	5.58	6.52
	proposed (BI 3 iter.)	9.58	8.01	**13.16**	5.58	5.74	6.56
Cameraman	BM3DDEB	8.20	6.46	8.37	3.35	3.72	4.70
	IDD-BM3D	8.85	7.12	**10.45**	3.98	4.31	4.89
	proposed (BM3DDEB init.)	**9.24**	**7.38**	10.21	**4.34**	**4.68**	**5.26**
	proposed (BI 1 iter.)	7.92	6.18	8.19	3.46	3.97	3.92
	proposed (BI 2 iter.)	8.27	6.65	9.57	3.80	4.39	4.27
	proposed (BI 3 iter.)	8.29	6.69	9.81	3.94	4.58	4.26

Fig. 7: Deblurring results (ISNR) for the images Lena and Cameraman (Scenario 2): First column: blurry images, Second column: BM3DDEB results, Third column: IDD-BM3D results, Fourth column: results of the proposed scheme initialized with the BM3DDEB results, Fifth column: results of the three rounds of the proposed scheme initialized with the blurry images.

$N = 512^2$, $L = 9$, $n = 121$ and $B = 21$, the above amounts to $1.26 \cdot 10^{11}$ operations.

We proceed to calculate the complexity of the proposed image deblurring algorithm. As we saw above, the calculation of all the RTBWT operators requires $O(LNB^2n)$ operations, and each denoising step requires $O(nLN) + O(nN) = O(nLN)$ operations. The deblurring step is performed in the Fourier domain and therefore requires $O(N \log N)$ operations. Thus, the overall complexity of the algorithm is

$$\begin{aligned} &O(LNB^2n) + G[O(nLN) + O(N \log N)] \\ &= O(LNB^2n + GnLN + GN \log N) \end{aligned} \tag{37}$$

and since the parameter we use satisfy $n(L+1) > \log N$ and $B^2 > G$, the complexity of the algorithm is again dominated by the calculation of the RTBWT permutation operators. For a typical case in our experiments, $N = 512^2$, $L = 9$, $n = 9$, $B = 51$ and $G = 50$, the above amounts to $5.64 \cdot 10^{10}$ operations.

In order to better illustrate these numbers, we also provide run-times: applying two iterations of our denoising scheme to a 512×512 image with noise level $\sigma = 25$ using a non optimized and non parallel matlab implementation, on an Intel(R) Core(TM) i7-2600 CPU @ 3.40 GHz, takes about 45 minutes. However, applying a single iteration followed

TABLE IV: Parameters Used in the Deblurring Experiments for the Proposed Scheme Initialized with the BM3DDEB Results (BM3DDEB init.) and for 3 Stages of the Proposed Scheme Initialized with the Blurry Image (BI).

Scenario	Stage	L	$\sqrt{n}$	B	η	τ	G
1	1 (BM3DDEB init.)	9	3	51	0.07	20	50
	1 (BI)	9	6	31	0.03	60	30
	2(BI)	9	4	41	0.03	50	30
	3 (BI)	9	4	41	0.06	30	30
2	1 (BM3DDEB init.)	9	3	51	0.08	30	50
	1 (BI)	9	6	31	0.07	60	30
	2 (BI)	9	4	41	0.1	50	30
	3 (BI)	9	4	41	0.16	30	30
3	1 (BM3DDEB init.)	9	3	51	0.01	20	50
	1 (BI)	9	6	31	0.005	50	30
	2 (BI)	9	4	41	0.005	60	30
	3 (BI)	9	4	41	0.015	30	30
4	1 (BM3DDEB init.)	9	3	51	0.9	20	50
	1 (BI)	9	6	31	0.5	50	30
	2 (BI)	9	4	41	0.8	40	30
	3 (BI)	9	4	41	1.9	20	30
5	1 (BM3DDEB init.)	9	3	51	0.05	20	50
	1 (BI)	9	6	31	0.03	60	30
	2 (BI)	9	4	41	0.07	40	30
	3 (BI)	9	4	41	0.14	20	30
6	1 (BM3DDEB init.)	9	3	51	3.7	20	50
	1 (BI)	9	6	31	3.2	50	30
	2 (BI)	9	4	41	3.9	30	30
	3 (BI)	9	4	41	3.8	30	30

by Wiener filtering takes only about 20 minutes. Applying a single round of our deblurring scheme, initialized with the BM3DDEB result, to a 512×512 image corrupted according to scenario 1 takes about 4 minutes. We should note that while in our experiments we employed exact exhaustive search, approximate nearest neighbor algorithms may be used to alleviate the computational burden.

V. Conclusions

We have revisited the redundant tree-based wavelet transform proposed in [1], described it in greater details and analyzed its properties. We have introduced an alternative implementation for this transform which is based on the à trous algorithm, and used it to show that the RTBWT is a frame. We have also shown that the image representation obtained with the RTBWT combined with the subimage averaging scheme also constitutes a frame, and calculated the synthesis and analysis operators associated with it. We have proposed image denoising and deblurring algorithms which make use of these operators as sparsity promoting regularizers, and demonstrated state-of-the-art results.

There are several research directions to extend this work that we are currently considering. The first is to explore the use of a weighted average in the subimage averaging scheme. Intuitively it seems that the RTBWT would better represent subimages which reside closer to the center of the image, and therefore their pixels should receive higher weights when the final estimate is calculated. A different direction is to replace the thresholding operator S_τ used in the proposed image processing schemes with an operator that takes into consideration the change in the noise levels in the different wavelet scales. Finally, the operators $\mathbf{\Phi}_{SA}$ and $\mathbf{\Psi}_{SA}$ may be used to solve different image processing problems such as image inpainting, superresolution, tomographic reconstruction, and more.

Appendix

A. Proof of Proposition 1

Our proof is similar in spirit to the one given in [18] for the common redundant wavelet transform. By the definition of a frame, it is sufficient to show that the frame bounds exist in order to show that the RTBWT is a frame. We notice that $\|\mathbf{c}\|^2 = \sum_n |\langle \phi_n, \mathbf{y} \rangle|^2$, and we next show that the coefficient vector $\mathbf{c}$ satisfies

$$2\|\mathbf{y}\|^2 \leq \|\mathbf{c}\|^2 \leq 2^L \|\mathbf{y}\|^2. \tag{A.1}$$

We first recall that the orthogonal wavelet filters satisfy [16]

$$\bar{h}[n] = h[-n] \text{ and } \bar{g}[n] = g[-n] \tag{A.2}$$

$$\sum_n h[n] = \sqrt{2} \text{ and } \sum_n g[n] = 0 \tag{A.3}$$

and that the filters $\mathbf{h}$ and $\mathbf{g}$ are power complementary, i.e. their Fourier transforms satisfy

$$|\hat{h}(2^\ell \omega)|^2 + |\hat{g}(2^\ell \omega)|^2 = 2. \tag{A.4}$$

We apply the Fourier transform to (9) and (10), and using (A.2) we obtain that

$$\hat{a}_{\ell+1}(\omega) = \hat{a}^p_\ell(\omega) \hat{h}^*(2^\ell \omega) \tag{A.5}$$

$$\hat{d}_{\ell+1}(\omega) = \hat{a}^p_\ell(\omega) \hat{g}^*(2^\ell \omega). \tag{A.6}$$

Next we use (A.4) and the fact that $\|\mathbf{a}^p_\ell\|^2 = \|\mathbf{a}_\ell\|^2$ and obtain that

$$\begin{aligned} &\|\mathbf{d}_{\ell+1}\|^2 + \|\mathbf{a}_{\ell+1}\|^2 \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |a^p_\ell(\omega)|^2 \left(|\hat{h}(2^\ell \omega)|^2 + |\hat{g}(2^\ell \omega)|^2 \right) d\omega \\ &= 2\|\mathbf{a}_\ell\|^2. \end{aligned} \tag{A.7}$$

We notice that

$$\begin{aligned} \|\mathbf{c}\|^2 &= \|\mathbf{a}_L\|^2 + \|\mathbf{d}_L\|^2 + \sum_{\ell=1}^{L-1} \|\mathbf{d}_\ell\|^2 \\ &= 2\|\mathbf{a}_{L-1}\|^2 + \|\mathbf{d}_{L-1}\|^2 + \sum_{\ell=1}^{L-2} \|\mathbf{d}_\ell\|^2 \end{aligned} \tag{A.8}$$

where we used (A.7) in the transition from the second to the third line. Finally from (A.7) and (A.8) and the fact that $\mathbf{a}_0 = \mathbf{y}$ we get that

$$\|\mathbf{c}\|^2 \leq 2\|\mathbf{a}_{L-1}\|^2 + 2\|\mathbf{d}_{L-1}\|^2 + \sum_{\ell=1}^{L-2} \|\mathbf{d}_\ell\|^2 \tag{A.9}$$

$$\leq 2^2 \|\mathbf{a}_{L-2}\|^2 + 2^2 \|\mathbf{d}_{L-2}\|^2 + \sum_{\ell=1}^{L-3} \|\mathbf{d}_\ell\|^2 \leq \ldots \leq 2^L \|\mathbf{y}\|^2 \tag{A.10}$$

and that

$$\|\mathbf{c}\|^2 \geq \|\mathbf{a}_{L-1}\|^2 + \|\mathbf{d}_{L-1}\|^2 + \sum_{\ell=1}^{L-2} \|\mathbf{d}_\ell\|^2 \tag{A.11}$$

$$\geq \|\mathbf{a}_{L-2}\|^2 + \|\mathbf{d}_{L-2}\|^2 + \sum_{\ell=1}^{L-3} \|\mathbf{d}_\ell\|^2 \geq \ldots \geq 2\|\mathbf{y}\|^2. \tag{A.12}$$

We now show that the bounds $\alpha = 2$ and $\beta = 2^L$ are the tightest possible frame bounds since we can find vectors that meet them. We first show that the vector $\mathbf{y}_N$, which satisfies $y_N[k] = \frac{1}{\sqrt{N}}$, $k = 1 \ldots, N$ meets the upper bound. Since $\mathbf{a}_0 = \mathbf{y}_N$ is a constant signal, we get that $\mathbf{a}_0^{1,p} = \mathbf{a}_0 = \mathbf{y}_N$. Using (A.2) and (A.3) we get that

$$\bar{h}_j[n] * y_N[n] = \frac{1}{\sqrt{N}} \sum_n \bar{h}[n] = \frac{1}{\sqrt{N}} \sum_n h[n]$$
$$= \sqrt{2} y_N[n] \tag{A.13}$$

and

$$\bar{g}_j[n] * y_N[n] = \frac{1}{\sqrt{N}} \sum_n \bar{g}[n] = \frac{1}{\sqrt{N}} \sum_n g[n] = 0. \tag{A.14}$$

From (A.13) and (A.14) we get that $\mathbf{a}_1 = \sqrt{2}\mathbf{a}_0 = \sqrt{2}\mathbf{y}_N$ and $\mathbf{d}_1 = \mathbf{0}$, where $\mathbf{0}$ denotes a vector of all zeros. Using similar calculations it can be shown that $\mathbf{a}_\ell = \left(\sqrt{2}\right)^\ell \mathbf{y}_N$ and $\mathbf{d}_\ell = \mathbf{0}$, therefore $\|\mathbf{c}\|^2 = 2^L \|\mathbf{y}_N\|^2$. Now, let $\tilde{y}_N[k] = \frac{1}{\sqrt{N}}(-1)^k$, $k = 1 \ldots, N$. Then using a similar procedure it can be shown that the vector $\tilde{\mathbf{P}}_0 \tilde{\mathbf{y}}_N$ meets the lower bound.

B. Proof of Proposition 2

We show that the representation $\mathbf{c}^{SA}$ is a frame by noticing that $\|\mathbf{c}^{SA}\|^2 = \sum_n |\langle \phi_n^{SA}, \mathbf{y} \rangle|^2$, and showing that

$$2\|\mathbf{y}\|^2 \leq \|\mathbf{c}^{SA}\|^2 \leq 2^L n \|\mathbf{y}\|^2. \tag{A.15}$$

We first use (16) and see that

$$\|\mathbf{c}^{SA}\|^2 = \mathbf{y}^T \left(\mathbf{\Phi}^{SA}\right)^T \mathbf{\Phi}^{SA} \mathbf{y} = \sum_{j=1}^{n} \mathbf{y}^T \mathbf{R}_j^T \mathbf{\Phi}^T \mathbf{\Phi} \mathbf{R}_j \mathbf{y}$$
$$= \sum_{j=1}^{n} \|\mathbf{\Phi} \mathbf{R}_j \mathbf{y}\|^2 \tag{A.16}$$

From Proposition 1 we have that

$$2\|\mathbf{R}_j \mathbf{y}\|^2 \leq \|\mathbf{\Phi} \mathbf{R}_j \mathbf{y}\|^2 \leq 2^L \|\mathbf{R}_j \mathbf{y}\|^2. \tag{A.17}$$

We notice that

$$\sum_{j=1}^{n} \|\mathbf{R}_j \mathbf{y}\|^2 = \mathbf{y}^T \left(\sum_{j=1}^{n} \mathbf{R}_j^T \mathbf{R}_j \right) \mathbf{y} = \mathbf{y}^T \mathbf{D} \mathbf{y} \tag{A.18}$$

and therefore

$$2\mathbf{y}^T \mathbf{D} \mathbf{y} \leq \|\mathbf{c}^{SA}\|^2 \leq 2^L \mathbf{y}^T \mathbf{D} \mathbf{y}. \tag{A.19}$$

As the minimum value of the diagonal matrix $\mathbf{D}$ equals 1, and the maximum value equals n, we obtain that

$$2\|\mathbf{y}\|^2 \leq \|\mathbf{c}^{SA}\|^2 \leq 2^L n \|\mathbf{y}\|^2. \tag{A.20}$$

REFERENCES

[1] I. Ram, M. Elad, and I. Cohen, "Redundant Wavelets on Graphs and High Dimensional Data Clouds," *IEEE Signal Processing Letters*, vol. 19, no. 5, pp. 291–294, 2012.

[2] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Modeling and Simulation*, vol. 4, no. 2, pp. 490–530, 2006.

[3] I. Ram, M. Elad, and I. Cohen, "Image processing using smooth ordering of its patches," *IEEE Trans. Image Processing*, vol. 22, no. 7, pp. 2764 – 2774, 2013.

[4] M. Elad, J. L. Starck, P. Querre, and D. L. Donoho, "Simultaneous cartoon and texture image inpainting using morphological component analysis (mca)," *Applied and Computational Harmonic Analysis*, vol. 19, no. 3, pp. 340–358, 2005.

[5] G. Plonka, "The Easy Path Wavelet Transform: A New Adaptive Wavelet Transform for Sparse Representation of Two-Dimensional Data," *Multiscale Modeling & Simulation*, vol. 7, p. 1474, 2009.

[6] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[7] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 2272–2279.

[8] G. Yu, G. Sapiro, and S. Mallat, "Solving inverse problems with piecewise linear estimators: from gaussian mixture models to structured sparsity," *Image Processing, IEEE Transactions on*, no. 99, pp. 1–1, 2010.

[9] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[10] ——, "Image restoration by sparse 3d transform-domain collaborative filtering," in *SPIE Electronic Imaging*, vol. 6812, 2008.

[11] A. Danielyan, V. Katkovnik, and K. Egiazarian, "BM3D frames and variational image deblurring," *IEEE Trans. Image Processing*, vol. 21, no. 4, pp. 1715–1728, 2012.

[12] I. Ram, M. Elad, and I. Cohen, "Generalized Tree-Based Wavelet Transform ," *IEEE Trans. Signal Processing*, vol. 59, no. 9, pp. 4199–4209, 2011.

[13] M. Shensa, "The discrete wavelet transform: Wedding the a trous and mallat algorithms," *IEEE Trans. Signal Processing*, vol. 40, no. 10, pp. 2464–2482, 1992.

[14] G. Beylkin, "On the representation of operators in bases of compactly supported wavelets," *SIAM J. Numer. Anal.*, vol. 29, no. 6, pp. 1716–1740, 1992.

[15] T. H. Cormen, *Introduction to algorithms*. The MIT press, 2001.

[16] S. Mallat, *A Wavelet Tour of Signal Processing, The Sparse Way*. Academic Press, 2009.

[17] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets. Time-Frequency Methods and Phase Space*, vol. 1. Springer-Verlag, 1989, pp. 286–297.

[18] J. Fowler, "The redundant discrete wavelet transform and additive noise," *IEEE Signal Processing Letters*, vol. 12, no. 9, pp. 629–632, 2005.

[19] M. Elad, P. Milanfar, and R. Rubinstein, "Analysis versus synthesis in signal priors," *Inverse problems*, vol. 23, no. 3, p. 947, 2007.

[20] S. Ghael, A. Sayeed, and R. Baraniuk, "Improved wavelet denoising via empirical wiener filtering," in *Proc. SPIE*, vol. 3169, 1997, pp. 389–399.