Fast Translation Invariant Multiscale Image Denoising

Meng Li and Subhashis Ghosal

EDICS Category: 3-BBND

M. Li and S. Ghosal are with the Department of Statistics, North Carolina State University, Raleigh, NC, 27695 USA (e-mail: mli9@ncsu.edu; ghoshal@stat.ncsu.edu). Research is partially supported by NSF grand number DMS-1106570.
Fast Translation Invariant Multiscale Image Denoising

Abstract—Translation Invariant (TI) cycle spinning is an effective method for removing artifacts from images. However, for a method using $O(n)$ time, the exact TI cycle spinning by averaging all possible circulant shifts requires $O(n^2)$ time where $n$ is the number of pixels, and therefore is not feasible in practice. Existing literature has investigated efficient algorithms to calculate TI version of some denoising approaches such as Haar wavelet [1]. Multiscale methods, especially those based on likelihood decomposition such as penalized likelihood estimator and Bayesian methods, have become popular in image processing because of their effectiveness in denoising images. As far as we know, there is no systematic investigation of the TI calculation corresponding to general multiscale approaches. In this paper, we propose a Fast Translation Invariant (FTI) algorithm and an Approximate Translation Invariant (ATI) cycle spinning technique, which are applicable to general $d$-dimensional images ($d = 2, 3, \ldots$) with either Gaussian or Poisson noise. The proposed FTI leads to the exact TI estimation but only requires $O(n \log_2 n)$ time. The proposed ATI can achieve almost the same performance as the exact TI estimation, but requires even less time. We achieve this by exploiting the regularity present in the multiscale structure, and give the theoretical justification. The proposed FTI and ATI are generic in that they are applicable on any smoothing techniques based on the multiscale structure. We demonstrate the FTI and ATI algorithms on some recently proposed state-of-the-art methods for both Poisson and Gaussian noised images. Both simulations and real data application confirm the appealing performance of the proposed algorithms. Matlab toolboxes are online accessible to reproduce the results and be implemented for general multiscale denoising approaches provided by the users.

Index Terms—image denoising, multiscale analysis, cycle spinning, translation invariant, Gibbs phenomenon, Gaussian noise, Poisson noise, 2-dimensional image, 3-dimensional image.

I. INTRODUCTION

Reconstruction of images based on noisy observations is often needed in various applications. As the number of pixels $n$ is often extremely large, computational challenges are often overwhelming. Approaches based on wavelet-type transformations and thresholding to draw boundaries have been commonly used [2], [3]. Multiscale methods (or multisresolution analysis in some literature), which decomposes the image in a sequence of refining blocks of pixels to factorize the likelihood function, have been proved to have appealing accuracy and computational efficiency for both count and continuous data types; see [4]–[11]. It is well known that many image denoising approaches may suffer from visual artifacts—the so-called Gibbs phenomena around the neighborhood of discontinuity, for example wavelet based approaches [1]; multiscale methods based on likelihood has the similar issue because of dyadic partition. Cycle spinning is a general technique to remove visual artifacts and improve accuracy in image reconstruction by averaging over shifts. Applying Translation invariant (TI) operation on a given smoothing method by considering all possible circulant shifts is conceptually appealing but is computationally intensive. When accomplished by the naive way which averages all possible “shift-denoise-unshift”, the computation becomes too intensive to be manageable, especially for 3-dimensional images. It becomes essential to take advantage of the multiscale structure to make the computation more efficient. However, unlike the case for wavelets, little or no work has been found to calculate the TI operator efficiently under this multiscale framework.

In this paper, we propose a Fast Translation Invariant (FTI) algorithm and an Approximate Translation Invariant (ATI) technique to accelerate a class of multiscale approaches for both continuously measured and count images. The FTI algorithm is computationally efficient, requiring $O(n \log_2 n)$ time to reconstruct the image; in contrast, the full TI estimation calculated in the naive way requires $O(n^2)$ time. One the other hand, the FTI algorithm only needs slightly longer time than the base method without any cycle spin which is $O(n)$. It is applicable to any smoothing procedure based on the multiscale decomposition. Further, the ATI algorithm can achieve almost the same performance as the FTI algorithm, but requires even less time.

The following paper is organized as follows. Section II introduces the cycle spinning technique and translation invariant operator. Section III details the multiscale likelihood decomposition for both Gaussian and Poisson noised images. Section IV describes the FTI algorithm and the theoretical justification, which is followed by Section V to illustrate the usage of FTI algorithm to applicable methods in the literature. Section VI includes the ATI algorithm and interpretation, while Section VII compares the computing time for the FTI and ATI algorithms. Section VIII investigates the performance of FTI and ATI algorithm to two selected denoising
II. CYCLE SPINNING AND THE TRANSLATION INARIANT (TI) OPERATOR

Cycle spinning has become a common technique to remove visual artifacts and improve the numerical accuracy in image reconstruction [1]. An image denoising method based on averaging over all possible circulant shifts is translation invariant (TI), for example the TI-Haar is the translation invariant version of the original Haar-wavelet [7]. We consider a general d-dimensional image \( X = \{X_i\} \), where the vector index \( i = (i_1, \ldots, i_d) \) and \( i_k = 1, \ldots, N \) for \( k = 1, \ldots, d \). We here consider the images with each dimension to be length \( N \), therefore the total number of pixels is \( n = N^d \). Define the modulation operator for a vector as the element-wise modulation, i.e. for given positive integer \( m, (i \mod m) = (j_1, \ldots, j_d) \) where \( j_k = (i_k \mod m) \) for \( k = 1, \ldots, d \). A circulant shift operator \( S_k \) indexed by \( i' = (i'_1, \ldots, i'_d) \) is defined by

\[
S_k : \mathbb{R}^{N^d} \rightarrow \mathbb{R}^{N^d}, (S_k(X)(i))_i = X((i-i') \mod N). \quad (1)
\]

For simplicity of notations, we use \( X_{i-i'} \) for the modulation \( X_{((i-i') \mod N)} \). We also use the notation \( S_{-i} = S_{i}^{-1} \) interchangeably. For any operator \( G \) to smooth an image \( X \), the induced translation invariant (TI) operator is defined as \( \bar{G} \) by averaging over all possible shifts, which means

\[
\bar{G}(X)_i = \frac{1}{N^d} \sum_{i'} ((S_{-i'} \circ G \circ S_k(X))(i))_i. \quad (2)
\]

If \( G \) is a linear operator, computation of the induced TI operator \( \bar{G} \) is relatively easy, which requires almost the same amount of computation as the original one because of the following lemma.

Lemma II.1. If an operator on \( X \) is defined as \( G(X)_i = \sum_j a_{i,j} X_j \) for some \( \{a_{i,j}\} \) whose value only depends on \((i, j)\), then \( \bar{G}(X)_i = \sum b_{i,j} X_j \) where \( b_{i,j} = N^{-d} \sum_{i'} a_{i+i',j+i'} \).

Local smoothing approaches such as kernel-based filters belong to this class. When the kernel function is chosen as \( K_h(\cdot, \cdot) \) with bandwidth \( h \), the coefficients \( a_{i,j} = K_h(i,j) \). The commonly used running mean (RM) filter is a special case of kernel-based approaches with the kernel function \( K_{RM}(i,j) = (2h+1)^{-d} I(||i-j||_\infty \leq 2h + 1) \); here \( I(\cdot) \) is the indicator function and the norm \( ||\cdot||_\infty \) is defined as the maximum absolute values of the vector argument. We can see that kernel-based approaches are automatically translation invariant using circulant boundaries, i.e. the input matrix (or generally array) values outside the bounds are computed by implicitly assuming the input is periodic. In practice, the readers can use the Matlab function \texttt{imfilter} with the boundary option to be “circular” for TI filters.

However, linear operators tend to oversmooth images and may lack the sharpness at the boundaries between features in the images. It excludes a lot of more refined approaches including wavelet based methods and its extension [1], [6] and multiscale likelihood based methods [5], [9], [11]. Wavelet-based approaches decompose an \( L^2 \) function using the wavelet basis functions, which can be interpreted as a multiscale representation of the target function. It has been widely used in signal and image processing, benefiting from its near-optimality in a minimax sense and practically efficient algorithms [2], [3]. In general, the TI operator is challenging to compute since it requires to denoise all the shifted images while there are \( n = N^d \) possible shifts in total. In the existing literature, computationally efficient TI algorithm has been studied only for the case of Harr wavelets and some corresponding variants [1], [7], which requires \( O(n \log_2 n) \) computing time.

In spite of effectiveness and computational efficiency of likelihood-based multiscale denoising methods, little is known about efficient computation of its TI operator. The goal of the present paper is to provide efficient algorithms for computing TI operators for general multiscale denoising methods.

III. MULTISCALE LIKELIHOOD REPRESENTATION

Multiscale likelihood representation (MLR) decomposes the entire image in a sequence of refining blocks of pixels to factorize the likelihood function. Next we shall introduce this multiscale framework using 2-dimensional (2D) images, although it is applicable to 3-dimensional (3D) or any \( d \)-dimensional images with \( d = 2, 3, \ldots \). Higher dimensional images many arise if the color or some other features are also taken into account for smoothing. For simplicity of notations, we assume the length of each dimension \( N = 2^L \) for some integer \( L \), and thus the total number of pixels \( n = N^d = 2^{Ld} \). For simplicity of notations, if an image (2D, 3D or generally \( d \)-dimensional) has the same length for each dimension then we call that length the size of the image.

A. 2D images

Starting with the original observed image (called level \( L \)), we combine a group of 4 neighboring pixels into one block by summing them together, resulting a coarser level of image size \( 2^{L-1} \). The block formed by this process is known as the parent. The four neighboring pixels forming the group are called children, and the formed
structure in this way is called a parent-child group. Continuing this grouping process until the grand sum (a scalar) is obtained, we get a multiscale representation consisting of levels \( l = L, L - 1, \ldots, 1, 0 \). Formally, the different scales of an image \( X = \{X_{j,k}\} \) are defined as follows. In the \( l \)'th scale of the image, the parent \((j, k)\)'th block pixel is split into 4 children of block-pixels at the \((l + 1)\)'th scale, which can be formulated as

\[
X_{l,(j,k)} = \sum_{j'=2j-1}^{2j-1} \sum_{k'=2k-1}^{2k} X_{l+1,(j',k')} \quad (3)
\]

where \( l = 0, 1, 2, \ldots, L - 1 \) and \( j, k = 1, \ldots, 2^l \). Here \( X_{L,(j,k)} = X_{(j,k)} \) and when \( l = 0 \), \( X_{0,(1,1)} \) is the summation of all the entire image. Define the local collapse operator to be \( H \) which sums over every size 2 block, then the \( l \)'th scale of the image \( X \) is obtained by

\[
X_l = H^{L-l}X, \quad (4)
\]

where \( X_l \) has size \( 2^l \) and \( l = 0, \ldots, L \). We call it the multiscale representation of \( X \) by collecting all the \( X_l \)'s, i.e. \( \{X_l : l = 0, \ldots, L\} \); and \( X_l \) is observation at the \( l \)'th level (or scale).

While \( X_{l,(j,k)} \) is the observation of the pixel \((j, k)\) at level \( l \), we use \( X^*_{l,(j,k)} \) to denote the vector of its children group \( \{X_{l+1,(2j-1,2k-1)}, X_{l+1,(2j-1,2k)}, X_{l+1,(2j,2k-1)}, X_{l+1,(2j,2k)}\} \). We can generally assume that the observation follows a parametric family independently, i.e. \( X_{l,j} \sim \mathcal{P}(\theta_{l,j}) \) up to unknown parameters \( \theta_{l,j} \). The entire image \( X \) can be viewed as following the joint distribution \( \mathcal{P}(\Theta) \). A multiscale statistical model is then given by the factorization of the statistical model for the entire image into the following:

\[
P(X|\Theta) = P_l(X_{0,(1,1)}; \theta_{0,(1,1)})
\times \prod_{l=0}^{L-1} \prod_{j=1}^{2^l} \prod_{k=1}^{2^l} \mathcal{P}_2(X^*_{l,(j,k)}|X_{l,(j,k)}, \theta_{l,(j,k)}),
\quad (5)
\]

where \( \mathcal{P}_1 \) is the distribution of the grand sum \( X_{0,(1,1)} \) and \( \mathcal{P}_2 \) denotes the conditional distribution of the children pixels \( X_{l,(j,k)} \) given the parent \( X_{l,(j,k)} \) with unknown parameters \( \theta_{l,(j,k)} \), which are typically transformations of the original set of parameters \( \{\theta_{l,j}\} \) instead of themselves.

This multiscale decomposition holds for various types of models. We next use the two commonly used models—Gaussian noised images and Poisson noised images for demonstration.

1) Gaussian noised images: The model for Gaussian noised images assume independent Gaussian noises with \( X_{(j,k)} \sim \text{Normal}(\mu_{(j,k)}, \sigma_0^2) \). We use \( \mu_{l,(j,k)} \) to denote the mean of \( X_{l,(j,k)}, l = 0, 1, \ldots, L \). The entire image can be viewed as \( P(X|\mu, \sigma_0) \). A multiscale statistical model is then given by the factorization of the statistical model for the entire image into the following:

\[
P(X|\mu, \sigma_0) = \mathcal{N}(X_{0,(1,1)}; \mu_{0,(1,1)}, 4\sigma_0^2)
\times \prod_{l=0}^{L-1} \prod_{j=1}^{2^l} \prod_{k=1}^{2^l} \mathcal{N}(X^*_{l,(j,k)}; \frac{1}{4}X_{l,(j,k)}1_4 + \xi_{l,(j,k)}, 4^{L-l}\Sigma_0),
\quad (6)
\]

where \( \mathcal{N} \) is the probability density function of the (multivariate) Gaussian distribution, \( 1_4 = (1, 1, 1, 1)^T \), and \( \Sigma_0 = I - 1_4(1_4^T1_4)^{-1}1_4^T = I - \frac{1}{4}1_41_4^T \) such that the sum the children is equal to their parent so that we can preserve the total exposure of the original image. Here the parameterization is given by

\[
\begin{align*}
\mu_{l+1,(2j-1,2k-1)} &= \frac{1}{4} \mu_{l,(j,k)} + \xi_{l+1,(2j-1,2k-1)}; \\
\mu_{l+1,(2j-1,2k)} &= \frac{1}{4} \mu_{l,(j,k)} + \xi_{l+1,(2j-1,2k)}; \\
\mu_{l+1,(2j,2k-1)} &= \frac{1}{4} \mu_{l,(j,k)} + \xi_{l+1,(2j,2k-1)}; \\
\mu_{l+1,(2j,2k)} &= \frac{1}{4} \mu_{l,(j,k)} + \xi_{l+1,(2j,2k)}.
\end{align*}
\quad (7)
\]

The reparameterization of the means emphasizes that we shall re-assign the weights of four children by \( \xi_{l,(j,k)} \) based on differences with \( \frac{1}{4} \mu_{l,(j,k)} \).

Let \( [x] \) is the ceiling function meaning the smallest integer not less than \( x \), then the \((j,k)\)'th pixel in the original image (or the \( L \)'th scale) is a child pixel of the \(((j/2), (k/2))\)'th pixel at the \((L - 1)\)'th scale, and generally a descendant pixel of the \(((j/2^{L-1}), (k/2^{L-1}))\)'th pixel at the \( l \)'th scale. Therefore, the parameter of interest is obtained by aggregating information across various scales and parent-child groups by

\[
\begin{align*}
\xi_{j,k} &= \sum_{l=1}^{L} \frac{1}{4^{L-l}} \xi_{l,([j/2^{L-1}],[k/2^{L-1}])}; \\
\mu_{j,k} &= \xi_{j,k} + \frac{1}{4} \mu_{0,(1,1)}.
\end{align*}
\quad (8)
\]

Let \( H^* \) to be the operator which divides (not replicates) each element of the input matrix (or array) evenly to a children block with size 2. Then the reparameterization step in equation (7) can be represented by the following matrix form

\[
\xi_{l} = \mu_{l} - H^* \mu_{l-1},
\quad (9)
\]

for \( l = 1, \ldots, L \), and \( \mu_{l} = H^{L-l} \mu \). Note that \( \xi_{l} \) has size \( 2^l \), and we let \( \xi_{0} = \mu_{0,(1,1)} \) in particular. Therefore, the aggregation step in equation (8) using a matrix form for all pixels:

\[
\mu = \sum_{l=0}^{L} (H^*)^{L-l} \xi_{l}.
\quad (10)
\]
2) Poisson noised images: The model for Poisson noised images assume independently Poisson observations with $X_{i,j} \sim \text{Poisson}(\lambda_{i,j})$. We use $\lambda_{i,j}$ for the mean of $X_{i,j}$, $l = 0, \ldots, L$. The entire image can be viewed as $P(X | \Lambda)$. A multiscale statistical model is then given by the factorization of the statistical model for the entire image into the following:

$$P(X | \Lambda) = P_1(X_{0,0,1}; \lambda_{0,0,1}) \times \prod_{l=0}^{L-1} \prod_{j=1}^{2^l} \prod_{k=1}^{2^l} P_2(X_{l,j,k}^*, | X_{l,j,k}, \rho_{l,j,k}^*),$$

(11)

where $P_1$ is the Poisson distribution, and $P_2$ is the multinomial distribution with the splitting probabilities $\rho_{l,j,k}^*$. Here the parameterization is given by

$$\begin{cases}
\lambda_{l+1,2j-1,2k-1} = \lambda_{l,j,k} \rho_{l,1,2j-1,2k-1}, \\
\lambda_{l+1,2j-1,2k} = \lambda_{l,j,k} \rho_{l,1,2j-1,2k}, \\
\lambda_{l+1,2j,2k-1} = \lambda_{l,j,k} \rho_{l,1,2j,2k-1}, \\
\lambda_{l+1,2j,2k} = \lambda_{l,j,k} \rho_{l,1,2j,2k}.
\end{cases}$$

(12)

The formulation of the model, parameterization and aggregation for Poisson noised images follow similar changes. Note that the definitions of $H$ and $H^*$ still hold for general $d$-dimensional images, therefore the parameter representation and aggregation for Gaussian noised images (equation (9) and (10)) and Poisson noised images (equation (14) and (15)) are applicable without any modification.

C. General denoising approaches

The multiscale structure allows to consider each parent-child group independently. For a parent-child group, whose size is 2 ($2d$), the total number of pixels is $2^d$ for $d$-dimensional matrix, numerous approaches are available in the literature to estimate the corresponding parameters. We shall consider a general class of denoising operator $G(\cdot)$ as follows. For any image $X$ with multiscale representation $\{X_l, l = 0, 1, \ldots, L\}$, our proposed fast TI technique considers any denoising operator $G(\cdot)$ satisfying the following general properties:

C1. $G(\cdot) = \{G_l(\cdot; \alpha_l) : l = 0, 1, \ldots, L\}$, where $G_l(\cdot; \alpha_l)$ is the denoising procedure operated on $X_l$ by each parent-child groups (size 2 blocks) independently and $\alpha_l$ is some tuning parameters;

C2. The tuning parameters $\alpha_l$ are selected based on the original image only without any cycle spinning, which can be considered given for our purpose;

C3. The resulting operator $G(\cdot)$ aggregates $G_l(\cdot; \alpha_l)$ across all the scales by summation in the form of (10) or (15), i.e.

$$G(X) = \sum_{l=0}^{L} (H^*)^{L-l} G_l(H^{L-l} X; \alpha_l).$$

(19)
Those conditions are not restrictive in practice. Condition C1 indicates that $G(\cdot)$ is a multiscale smoothing operator, which is based on the multiscale representation $X_t$ and parent-child groups. Condition C2 means the selected tuning parameters will be shared by different cycle spinning; this is reasonable since tuning parameters are typically related to smoothness level which is intuitively to be the same regardless how the image is shifted. In Condition C3, both Gaussian and Poisson models are applicable to the mentioned additive aggregation.

The resulting class of estimators is quite general [5], [9], [11]–[15, to name a few]. In fact, maximum likelihood estimators with appropriate penalties and Bayesian estimators with independent priors on each parent-child group but the priors of each scale depend on the same smoothing parameters are members of this class. Section V illustrates some selected methods which are applicable to our proposed algorithms. Since the tuning parameters $\alpha_l$ are selected using the original image, therefore we consider $\alpha_l$ as given. For notational simplicity, we use $G_l(\cdot)$ for $G_l(\cdot; \alpha_l)$.

IV. FAST TRANSLATION INVARIANT (FTI) ALGORITHM FOR MULTISCALE METHODS

In this section, we shall introduce the Fast Translation Invariant (FTI) algorithm for any operator $G(\cdot)$ defined in Section III-C. We first detail the algorithm for 2D images in Section IV-A and then introduce the general algorithm for $d$-dimensional images in Section IV-B using compact notations. As discussed in Section VII, the proposed algorithm can calculate the full TI estimation in $O(n \log_2 n)$ time.

A. FTI algorithms for 2D images

- **Step 1: Obtaining the TI Decomposition table (D-table).** We use $R_{l,(s,t)}$ to denote a size $2^l$ matrix where $l = 1, \ldots, L$ and $s, t = 0, \ldots, 2^{L-l} - 1$. Let $R_{l,(0,0)} = X$. For $j = L, \ldots, 2$ and $s, t = 0, \ldots, 2^{L-j} - 1$, $R_{j,(s,t)}$ is generated

$$
R_{j-1,(2s,2t)} = C_{0,0} R_{j,(s,t)};
R_{j-1,(2s,2t+1)} = C_{0,1} R_{j,(s,t)};
R_{j-1,(2s+1,2t)} = C_{1,0} R_{j,(s,t)};
R_{j-1,(2s+1,2t+1)} = C_{1,1} R_{j,(s,t)};
$$

where the operator $C_i = H \circ S_i$ for $i \in S_1$. We call it the D-table by collecting all $R_{l,(s,t)}$'s, i.e.

$$
\text{D-table} : \{ R_{l,(s,t)} : l = 1, \ldots, L; s, t = 0, \ldots, 2^{L-l} - 1 \}.
$$

- **Step 2: Obtaining the full TI table (α-table).** For $j = 1, \ldots, L$, and $s, t = 0, \ldots, 2^{L-j} - 1$, let

$$
\begin{align*}
\alpha_{j,(s,t)}^{(0,0)} &= S_{0,0}^{-1} G_j S_{0,0} R_{j,(s,t)}; \\
\alpha_{j,(s,t)}^{(1,0)} &= S_{0,1}^{-1} G_j S_{0,1} R_{j,(s,t)}; \\
\alpha_{j,(s,t)}^{(0,1)} &= S_{1,0}^{-1} G_j S_{1,0} R_{j,(s,t)}; \\
\alpha_{j,(s,t)}^{(1,1)} &= S_{1,1}^{-1} G_j S_{1,1} R_{j,(s,t)}.
\end{align*}
$$

We call it the α-table by collecting all these α’s, i.e.

$$
\text{α-table} : \{ \alpha_{l,(s,t)}^{(0,0)}, \alpha_{l,(s,t)}^{(0,1)}, \alpha_{l,(s,t)}^{(1,0)}, \alpha_{l,(s,t)}^{(1,1)} : l = 1, \ldots, L; s, t = 0, \ldots, 2^{L-l} - 1 \}.
$$

- **Step 3: Full TI estimation.** For $j = 1, \ldots, L$, and $s, t = 0, \ldots, 2^{L-j} - 1$, let

$$
\gamma_{j,(s,t)} = \frac{1}{4} \left( \alpha_{j,(s,t)}^{(0,0)} + \alpha_{j,(s,t)}^{(1,0)} + \alpha_{j,(s,t)}^{(0,1)} + \alpha_{j,(s,t)}^{(1,1)} \right).
$$

Let $\beta_{0,(1,1)} = X_{0,(1,1)}$. Start from $j = 0$ and let

$$
\beta_{j+1,(s,t)} = \gamma_{j+1,(s,t)} + \frac{1}{4} \left\{ S_{0,0}^{-1} H^* \beta_{j,(2s,2t)} + S_{0,1}^{-1} H^* \beta_{j,(2s,2t+1)} + S_{1,0}^{-1} H^* \beta_{j,(2s+1,2t)} + S_{1,1}^{-1} H^* \beta_{j,(2s+1,2t+1)} \right\}.
$$

After exhausting all $s, t$ for one level, then set $j = j + 1$ and repeat. Stop when reaching $j = L$. The final estimate is $\beta_{L,(0,0)}$.

B. FTI algorithm for $d$-dimensional images

The TI algorithm introduced in Section IV-A is extendable to 3D or more generally $d$-dimensional images ($d = 2, 3, \ldots$) as follows.

- **Step 1: Obtaining the TI Decomposition table (D-table).** Let $R_{l,s}$ denote a size $2^l$ matrix where $l = 1, \ldots, L$, and $s \in \mathbb{S}_L - 1$. Let $R_{L,0} = X$. For $j = L, \ldots, 2$, let $R_{j-1,2s+2t} = C_{s'} R_{j,s}$ for all $s \in \mathbb{S}_L - j$ and $s' \in \mathbb{S}_1$ where the operator $C_{s'} = H \circ S_{s'}$ for $s' \in \mathbb{S}_1$. The D-table is defined as: \{ $R_{l,s} : l = 1, \ldots, L; s \in \mathbb{S}_{L-1}$ \}.

- **Step 2: Obtaining the full TI table (α-table).** For $j = 1, \ldots, L$, and $s \in \mathbb{S}_{L-j}$, let $\alpha_{j,s} = S_{s'}^{-1} G_j S_{s'} R_{j,s}$ for all $s \in \mathbb{S}_{L-j}$ and $s' \in \mathbb{S}_1$. Then the α-table is defined as \{ $\alpha_{j,s} : l = 1, \ldots, L; s \in \mathbb{S}_{L-1}, s' \in \mathbb{S}_1$ \}.

- **Step 3: Full TI estimation.** For $j = 1, \ldots, L$, and $s \in \mathbb{S}_{L-j}$, let $\gamma_{j,s} = \sum_{s' \in \mathbb{S}_1} \alpha_{j,s'/2^d}$. Let $\beta_{0,1} = X_{0,1}$. Start from $j = 0$ and let

$$
\beta_{j+1,s} = \frac{1}{2^d} \sum_{s' \in \mathbb{S}_1} S_{s'}^{-1} H^* \beta_{j,2s+1} + \gamma_{j+1,s}.
$$

After exhausting all $s$ for one level, then set $j = j + 1$ and repeat. Stop when reaching $j = L$. The final estimate is $\beta_{L,0}$.
C. Theory and Interpretation of the FTI algorithm

We call two images have the same composition if they are identical to each other after some cycle spinning. The first two steps to obtain the \( \alpha \)-table is actually to obtain all unique compositions after the application of the smoothing operator \( G \) for all multiscale levels of the observed image. Therefore, the recovery of Full TI estimation is to look at the \( \alpha \)-table and then aggregate the selected pieces appropriately, which is given by Step 3.

For any given cycle spinning indexed by \( \ell \), it has two effects to a multiscale denoising approach. First, the multiscale representation of \( X \) may have different compositions. Second, when two level images have the same composition, the smoothing operator \( G_l(\cdot) \) can produce different outputs since the grouping of the pixels may be different. These two effects can be formalized by the following lemma.

**Lemma IV.1.** For a given vector index \( \ell \) for general dimensional image, let \( \phi(\ell, p) = ([\ell/2^p-1] \mod 2) \) for an integer \( p \), \( \ell_p = [\ell/2^p] \). Then for \( l = 1, 2, \ldots, L \),

\[
H_l(S_l X) = S_l(G(C_\phi(\ell,l) \circ \cdots \circ C_\phi(1,l)) X). \tag{27}
\]

In addition, the smoothed image for the \( l \)th level \( G_l(H_l(S_l X)) \) is given by

\[
S_l[\phi(\ell,l+1)] G_l(S_l[\phi(\ell,l+1)]((C_\phi(\ell,l) \circ \cdots \circ C_\phi(1,l)) X)]. \tag{28}
\]

**Remark IV.2** (Interpretation of the D-table). For given index \( \ell \), \( \phi(\ell, p) \) is the \( p \)th digit in the binary representation of \( \ell \). For the binary series \( \{\phi(\ell,1), \ldots, \phi(\ell, L-1)\} \), note that \( (C_\phi(\ell,L-1) \circ \cdots \circ C_\phi(1,l)) X \) corresponds to the \( t_\ell \)th element of \( R_{t_\ell} \), where \( t_\ell \) is the decimal number using the sequence as binary digits. In other words, for any \( \ell \in \mathbb{S}_{L-1} \), we have

\[
R_{t_\ell} = (C_\phi(\ell,L-1) \circ \cdots \circ C_\phi(1,l)) X, \tag{29}
\]

where \( t_\ell = \phi(\ell,1) \times 2^{L-1} + \cdots + \phi(\ell, L-1) \times 2^0 \). Therefore, the D-table gives all possible compositions of the multiscale representation of \( X \) due to an arbitrary cycle spinning, according to equation (27). Note that for each \( \ell \), the D-table only has \( 2^{L-1} \) elements which is a lot less than the number of possible cycle spinnings \( 2^{L \times L} \) when \( L \) close to \( L \), and this partially explains why the calculation using the naive way to obtain the Full TI operator is redundant. Since higher levels (larger values of \( L \)) have more pixels, the proposed algorithm is particularly useful.

**Remark IV.3** (Interpretation of the \( \alpha \)-table). For any given level \( \ell \), according to equation (29) and the definition of \( \alpha \)-table, we have \( \alpha_{t_\ell}^{\phi(\ell,L-1+1)} \) is

\[
S_{\phi(\ell,L-1+1)} (C_\phi(\ell,L-1) \circ \cdots \circ C_\phi(1,l)) X. \tag{30}
\]

Consequently, by equation (28), the level estimation is

\[
G_l(H^{L-1}_l(S_l X) = S_l G_l(S_l X)] = S_{l-1} \alpha_{l-1}^{\phi(\ell,L-1+1)} \tag{31}
\]

Therefore the \( \alpha \)-table contains all the matrices or d-dimensional arrays we need to recover \( G(S_l X) \) for any \( \ell \) and thus the resulting Full TI estimation.

The following theorem guarantees that the resulting estimate obtained by the proposed TI algorithm is equal to the one obtained in a naive way.

**Theorem IV.4.** The estimate obtained from Step 3 equation (25) or generally equation (26) is the Full TI estimate, i.e.

\[
\beta_{L,(0,0)} = \frac{1}{m} \sum_{l \in \mathbb{S}_L} (\phi \circ G \circ S_l X). \tag{32}
\]

V. Illustration for Applying FTI Algorithm to Existing Methods

The proposed algorithm is applicable to general multiscale methods defined in Section III-C, to produce the induced TI smoothing operator. We here select some of the available methods for illustration.

For any multiscale based approach, \( G_l \) operates on \( X_l \) by each parent-child group, which is a size 2 block. Therefore, it is sufficient to focus on one single parent-child group, say the \( (j,k) \)th group at the \( (l+1) \)th scale \( X^*_{l,j,k} \) (the corresponding parameters are \( \theta_{l,j,k} \)) in equation (5) to introduce each denoising approach.

1. **Complexity penalized likelihood estimator** [5] estimated the parameters by maximizing the likelihood function with a complexity penalty, i.e. \( G_{l+1}(X^*_{l,j,k}) =: \hat{\theta}_{l,j,k} \) is defined as

\[
\arg \min \{- \log p(X^*_{l,j,k} | X_{l,j,k}, \theta_{l,j,k}) + 2\lambda \cdot \text{pen}(\theta_{l,j,k})\}, \tag{33}
\]

where \( \lambda \) is a tuning parameter and \( \text{pen}(\theta_{l,j,k}) \) is the proposed complexity penalty (equation (8) in [5]). This method is applicable for both Gaussian and Poisson noised images.

2. **Bayesian estimator.** Bayesian procedures can estimate the parameters using the posterior expectation (or other loss criteria such as the posterior median), where the posterior distribution which is obtained from a prior distribution and the likelihood function, i.e.

\[
G_{l+1}(X^*_{l,j,k}) =: \hat{\theta}_{l,j,k} = E(\theta_{l,j,k} | X^*_{l,j,k}).
\]

Various priors have been proposed for both Poisson and Gaussian noised images (recall that the parameters are \( \rho \)'s for Poisson noised images and \( \xi \)'s for Gaussian noised images):

- 1D Poisson signal: [12] used mixture of beta(1,1) as the prior;
- 1D Poisson signal: [13] used beta(\(\alpha, \alpha\) as the prior;
• 2D Poisson images: [15] used Dirichlet($\alpha, \alpha, \alpha$) as the prior for 2D images, and further used a hyper-prior for $\alpha$;
• 2D and 3D Poisson images: [9] used the Dirichlet distribution as the prior, and further used a Chinese restaurant process (CRP) which is a one-parameter family of distributions on partitions that helps create ties among parameters. This prior was extended to 3D and colored Poisson noised images in [10];
• 2D Gaussian images: [14] used a multivariate Gaussian (mean 0) and discussed the usage of mixture multivariate Gaussian distributions as the prior;
• 2D and 3D Gaussian images: [11] used a multivariate Gaussian (mean 0) and further used a CRP to create ties among $\xi$'s.

All those mentioned methods essentially are to specify the smoothing operator $G_l$, and then application of the proposed TI algorithm in Section IV-A or Section IV-B will lead to the corresponding Full TI operator.

Among all those approaches, we shall select two recently proposed approaches — Bayesian CRP for Poisson [9] and Gaussian noised images [11], because the incorporation of these structures via the CRP prior has been proved to incredibly useful for image reconstruction and showed to be state-of-the-art by the authors; and the two methods also have Matlab toolbox available online.

Note that the Bayesian CRP method requires a tunning step. For example, when applied to Gaussian noised images, there are two smoothing parameters $M$ and $\tau$ at each scale to determine the tieing structure via the CRP probability allocation and the prior distribution of distinct values in a parent-child group. In [11], the smoothing parameters $(M, \tau)$ can be selected by maximizing the marginal likelihood; in addition, the global parameter $\sigma^2_G$ can be estimated either based on a selected background with 0 mean or the completely data-driven approach used in that paper. When applied to Poisson noised images, we just need to tune $M$ at each scale.

VI. APPROXIMATE TRANSLATION INVARIANT (ATI) ALGORITHM

Note that in the $l$th scale of the image, two pixels are possibly enforced to be a “family” when their distance at each axis direction does not exceed $2^l$. This leads to the fact that estimators based on the multiscale likelihood representation typically are influenced more by the finer scales rather than the coarser scales. It makes sense since we would not believe that two pixels with distance $2^l$ (for example, $l = 5$), which are far from each other could be possibly tied together with each other. The coarser scales are mainly used to align “parent” and “children” appropriately, instead of estimating “parent” parameters accurately. This observation indicates that an approximate version of the TI algorithm up to some scale may improve the performance of denoising as much as the Full TI algorithm. In addition, we have the following result.

Lemma VI.1. The estimation using all shifts indexed by $S_k$ leads to a TI version of a multiscale smoothing operator for the $l$th scale, where $l \geq L - k + 1$.

Therefore, the collection of shifts in $S_k$ will give all possible estimates due to circulant shift operators for the last $k$ scales of the image. Note that $S_k$ only has $2^k$ elements, which is much less compared to the total number of shifts $2^{dl}$ especially when $k$ is small such as $k = 1$. This observation plays the key role when simplifying the calculation of TI estimation to make the computation scalable. For simplicity of notation, we have the definition as follows.

Definition VI.2. We define the set of shifts indexed by $S_k$ as the k-TI shifts, and the induced TI operator as k-TI operator. In particular, 0-TI is the original operator without any cycle spinning, and L-TI is the Full TI operator.

In fact, the proposed ATI algorithm has another interpretation indicated by the following Lemma VI.3. Let the parameters at the $l$th scale be $T_l(\theta)$ which is given by equation (9) for Gaussian noised images and equation (14) for Poisson noised images. We define a denoising operator $G_l$ at level $l$ to be effectively null if $G_l(X_l) = T_l(X)$. It means the operator $G_l$ does not denoise the observations for level $l$, which is reasonable for small $l$’s (coarse levels). Many methods satisfy this when the involved tunning parameters are extreme in some sense; for example:

• Complexity penalized likelihood estimator in [5] when $\lambda = 0$ at level $l$;
• Bayesian CRP method for Gaussian noised images in [11] when $M, \tau \rightarrow \infty$ at level $l$;
• Bayesian CRP for Poisson noised images in [9], [10] when $M \rightarrow \infty$ at level $l$.

Lemma VI.3. For a multiscale denoising operator $G = \{G_l : l = 1, \cdots, L\}$ where $G_l$ is effectively null for $l \leq L - k$, then the full TI estimation is equal to the $k$-TI estimation.

When $k < L$, the computation of $k$-TI is more efficient than Full TI since only a portion of matrices in the D-table and $\alpha$-table will be used. It is easy to modify the Full TI algorithm to the $k$-TI version, therefore we only briefly describe here for completeness.

Step 1. D-table. The D-table of $k$-TI is

\[
\{R_{l,s} : l = 1, \cdots, L - k - 1, s \in S_{L-k}\} \\
\cup \{R_{l,s} : l = L - k, \cdots, L, s \in S_{L-L}\}.
\]
• Step 2. α-table. The α-table of k-TI is
\[ \{\alpha_{l,s}^k : l = 1, \ldots, L - k - 1, s \in S_{L-k} \} \cup \{\alpha_{l,s}^k : l = L - k, \ldots, L, s \in S_{L-l} \} \]
(35)

• Step 3. k-TI estimation. For \( j = 1, \ldots, L - k - 1 \), let \( \gamma_j, s = \alpha_{l,s}^k \); for \( j = L - k, \ldots, L \), and \( s \in S_{L-j} \), let \( \gamma_j, s = \sum_{s' \in S_{l}} \alpha_{l,s'}^k / 2^d \). Let \( \beta_{0,1} = X_{0,1} \). Start from \( j = 0 \) and let
\[
\beta_{j+1,s} = \frac{1}{2^d} \sum_{s' \in S_{l}} S_{s'}^{-1} H^* \beta_{j,2s+1} + \gamma_{j+1,s}. \quad (36)
\]
After exhausting all \( s \) for one level, then set \( j = j + 1 \) and repeat. Stop when reaching \( j = L \). The final estimate is \( \beta_{L,0} \).

Following the same argument used in Theorem IV.4, we can see that the resulting estimate is the k-TI estimate, i.e.
\[
\beta_{L,(0,0)} = \frac{1}{n} \sum_{i \in S_k} ((S_{-i} \circ G \circ S_{1}) X). \quad (37)
\]

VII. COMPUTING TIME

The total computing time is determined by three parts: multiscale decomposition (MD), tuning, estimation at each scale and aggregation. In the following table, note that \( n = 2^d L \) and \( 1 \leq k \leq L = \frac{1}{2} \log_2 n \). Table I shows the summary of the required computing time for a general operator \( G \).

The time for tuning as \( c(n) \) is a function of \( n \), whose order depends on the concrete optimization algorithms that are used. Most smoothing operators either do not require tuning or it could be conducted in \( c(n) = O(n) \) time.

The \( l \)th level of the observation has \( 2^d l \) pixels and thus requires \( 2^{d(l-1)} \) calculations to obtain the multiscale decomposition for the next level. Therefore, the total time for the multiscale decomposition (MD) step is \( O(\sum_{l=2}^{L} 2^{d(l-1)}) = O(2^d L) = O(\log_2 n) \). Similarly the aggregation step, which is the inverse procedure of the decomposition step in some sense, requires \( O(n) \) time as well. We consider the calculation for each parent-child group as the unit, and denote the corresponding time as \( c \). For the \( l \)th level, the size of the corresponding level observation is \( 2^l \), which means that the number of groups with size 2 is \( 2^{d(l-1)d} \), where \( l = 1, 2, \ldots, L \). Therefore the total time for smoothing the entire image is \( \sum_{l} c \times 2^{d(l-1)d} = O(n) \).

The naive TI approaches considers all possible \( n \) shifts, which means the total calculation is \( n \) times \( O(n) \) in addition to the tuning step. Therefore it requires \( c(n) + O(n^2) \) time.

The FTI algorithm takes advantage of the multiscale structures. The TI decomposition table \( \{ R_{l,s} : l = 1, \ldots, L; s \in S_l \} \) requires \( O(n) \) times to calculate for each \( l \), since the size of \( R_{l,s} \) is \( 2^l \) and there are \( 2^{L-l} \) of these at each dimension which leading to \( O(2^{dL}) = O(n) \) calculations. Therefore the total time to obtain the TI decomposition table is \( O(n) \times L = O(n \log_2 n) \). This calculation applies for the estimation and aggregation, which also require \( O(n \log_2 n) \) time.

The FTI algorithm addresses \( L \) matrices with the same size as the original observation separately, while the TI algorithm only does this for the last several fine levels, say the last \( k \) fine levels. For the other levels, ATI just addresses smaller matrices whose sizes decrease by \( 1/2 \) each time, which is negligible when considering the order of the time. Therefore the total time of \( k \)-TI is \( O(nk) \), \( k = 1, \ldots, L = \frac{1}{2} \log_2 n \).

VIII. EXPERIMENTS FOR THE EFFECT OF SHIFTS

We investigate the numerical performance of the Full TI estimation and more generally \( k \)-TI estimation for \( k = 0, 1, \ldots, L \), to the selected denoising methods. We use simulated 2D and 3D SheppLogan phantom (various sizes) with Gaussian noise (standard deviation \( \sigma = 0.1 \)) and apply the Bayesian CRP method of [11]; we use a Saturn image (maximum intensity is 1) with Poisson noise and apply the Bayesian CRP method of [9]. We also investigate the performance of \( k \)-TI algorithms on real astronomical images from the Chandra X-ray Observatory satellite (Figure 3).

Figure 1 plots the estimation with various TI levels for 2D phantom and Saturn images. We can see that 0-TI estimates (subplot (e) and (i)) have obvious artifacts. 1-TI improves the smoothed images a lot, at least visually; and Full TI estimates (subplot (h) and (l)) lead to much better outputs. We also notice that 4-TI estimates are very close to Full TI estimates here.

In addition to this visual comparison, we conduct 100 simulations to investigate the performance of \( k \)-TI estimates numerically, using both mean square errors (MSE) and computing time. Figure 2 shows that the circulate shifts increase the accuracy of estimation and the MSE can be decreased by almost one half when the Full TI estimation is used, compared to the original estimation. In the plots for Time (subplot (d), (e) and (f)), we can see the time used by our algorithm is linear in the TI level \( k \), i.e. \( O(kn) \); while the naive shift-denosing-unshift approach requires \( O(4^k n) \) time (2D image) and \( O(8^k n) \) time (3D image). Therefore, the Full TI estimation has much better accuracy than the original estimation and is also computationally feasible. We also can observe that \( k \)-TI estimation with \( k = 3 \) or \( k = 4 \) have almost the same accuracy as the Full TI estimation but requires less time. The investigation of the effect of shifts to the accuracy becomes more important for 3D images where computing time is a main concern for most
TABLE I
SUMMARY OF COMPUTING TIME FOR A GENERAL OPERATOR G APPLIED ON THE IMAGE X WHICH HAS n PIXELS IN TOTAL. THE FIRST ROW IS THE TIME REQUIRED BY APPLYING G ON THE ORIGINAL IMAGE WITHOUT ANY CYCLE SPINNING.

<table>
<thead>
<tr>
<th>Method</th>
<th>Tuning</th>
<th>MD</th>
<th>Estimation</th>
<th>Aggregation</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Without CS</td>
<td>c(n)</td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(n)</td>
<td>c(n)</td>
</tr>
<tr>
<td>2. Naive TI</td>
<td>c(n)</td>
<td>O(n^2)</td>
<td>O(n^2)</td>
<td>O(n^2)</td>
<td>c(n)</td>
</tr>
<tr>
<td>3. FTI</td>
<td>c(n)</td>
<td>O(n log 2 n)</td>
<td>O(n log 2 n)</td>
<td>O(n log 2 n)</td>
<td>c(n)</td>
</tr>
<tr>
<td>4. k-TI</td>
<td>c(n)</td>
<td>O(n^k)</td>
<td>O(n^k)</td>
<td>O(n^k)</td>
<td>c(n)</td>
</tr>
</tbody>
</table>

where the last step is obtained by substituting k = j - i'. Therefore, the TI operator \( \tilde{G}(X) \) is

\[
\frac{1}{N^d} \sum_{i'} \sum_{k} a_{i'+i',k+i'} X_k = \frac{1}{N^d} \sum_{k} \sum_{i'} a_{i'+i',k+i'} X_k, \tag{38}
\]

which is \( \sum b_{i,k} X_k \) by the definition of \( b_{i,k} \).

**Proof of Lemma IV.1.** We first show equation (27) by induction. For an arbitrary index \( i \), the image \( H(S_i X) \) has the same composition with \( H(S_{\lfloor i/2 \rfloor} X) \), or equivalently \( C_{\phi(i,1)} X \). Let \( r_1 = \lfloor i/2 \rfloor \), then it is easy to see

\[
H(S_i X) = S_{\lfloor i/2 \rfloor}(H(S_{\lfloor i/2 \rfloor} X)) = S_{ri}(C_{\phi(i,1)} X), \tag{39}
\]

which shows the result when \( l = 1 \). Assume that equation (27) holds for \( l < L \), then for \( l + 1 \),

\[
H^{l+1}(S_i X) = H(S_{ri}(C_{\phi(i,l)} \cdots C_{\phi(i,1)} X)) = S_{ri/2}(H(S_{\lfloor i/2 \rfloor} X) \cdots C_{\phi(i,1)} X), \tag{40}
\]

Noting that \( \lfloor i/2 \rfloor = \lfloor \lfloor i/2 \rfloor/2 \rfloor = \lfloor i/2^l \rfloor = i_{l+1} \), and \( (i_{l+1}) \mod 2 = \phi(i,l+1) \), equation (27) follows. This completes the proof according to induction.

Note that the operator \( G_l(S_l X) \) has the same composition as \( G_l(X) \) if \( (i_{mod 2}) = 0 \) according to Condition C1. Therefore, we have

\[
G_l(S_i X) = S_{l-\phi(i,l+1)}(G_l(S_{\lfloor i/2 \rfloor} X)) =: S_{l-\phi(i,1)}(G_l(S_{\phi(i,1)} X)). \tag{41}
\]

Combined with equation (27), we have

\[
G_l(H^l(S_i X)) = G_l(S_{l-\phi(i,l+1)}(G_l(S_{\phi(i,1)} X)) = S_{l-\phi(i,1)} G_l(S_{\phi(i,1)}(C_{\phi(i,1)} \cdots C_{\phi(i,1)} X)), \tag{42}
\]

Noting that \( \phi(i,1) = \phi(i,l+1) \) and \( S_{l-\phi(i,l+1)} = S_{l} \), equation (28) follows.

**Proof of Theorem IV.4.** By noting that the operator \( (H^*)^{l-1} \circ S_j = S_{2^l-1-j} \circ (H^*)^{l-1} \) for any \( j \) and applying equation (31), we have

\[
(H^*)^{l-1} G_l(H^{l-1} S_i X) = S_{2^l-i-l} (H^*)^{l-1} G_l(S_{\phi(i,L-l+1)} X), \tag{43}
\]
Fig. 1. Smoothed images with various translation invariant levels. Then phantom image (size 512) has Gaussian noise (standard deviation 0.1); (a), (b) and (e)–(h) are the true image, observation, 0-TI, 1-TI, 4-TI and Full TI estimates. The Saturn image (size 256; maximum intensity is 1) has Poisson noise; (c), (d) and (i)–(l) are the true image, observation, 0-TI, 1-TI, 4-TI and Full TI estimates.

Since $i = 2^{L-l}i_{L-l} + (i \mod 2^{L-l})$, we have

$$S_{-i}[((H^*)^{L-l}G_i(H^{L-l}S_iX)] = S_{-(i \mod 2^{L-l})}(H^*)^{L-l}\alpha_{i,t_{i,j}}^{L-l+1}.$$  \hspace{1cm} (44)

Consequently,

$$S_{-i}G(S_iX) = S_{-i} \sum_{l=0}^{L} (H^*)^{L-l}G_i(H^{L-l}S_iX)$$

$$= \sum_{l=0}^{L} S_{-i}[(H^*)^{L-l}G_i(H^{L-l}S_iX)]$$

$$=(H^*)^{L}(H^LX) + \sum_{l=1}^{L} S_{-(i \mod 2^{L-l})}(H^*)^{L-l}\alpha_{i,t_{i,j}}^{L-l+1}.$$  \hspace{1cm} (45)

Therefore,

$$\frac{1}{n} \sum_{i,t_{i,j} \in S_L} ((S_{-i} \circ G \circ S_i)X) - (H^*)^{L}(H^LX)$$

$$= \frac{1}{n} \sum_{i,t_{i,j} \in S_L} \sum_{l=1}^{L} S_{-(i \mod 2^{L-l})}(H^*)^{L-l}\alpha_{i,t_{i,j}}^{L-l+1}$$

$$= \sum_{l=1}^{L} \frac{1}{n} \sum_{i,t_{i,j} \in S_L} S_{-(i \mod 2^{L-l})}(H^*)^{L-l}\alpha_{i,t_{i,j}}^{L-l+1}.$$  \hspace{1cm} (46)

For fixed $l = 1, \ldots, L$, for the same value of $(i \mod 2^{L-l})$, we have the same value of $t_{i,j}$, but $\phi(i, L-l+1)$ can take all values in $S_1$ with the same possibilities, if
we consider all shifts in $S_L$. Therefore,
\[
\frac{1}{n} \sum_{i \in S_L} S_{-i \mod 2^{L-1}} (H^*)^{L-l} \alpha_{i,t_{k,l}}
\]
\[
= \frac{1}{n} \sum_{i \in S_L} S_{-i \mod 2^{L-1}} (H^*)^{L-l} \gamma_{i,t_{k,l}}
\]
\[
= \frac{1}{2d(L-l)} \sum_{i \in S_{L-1}} S_{-i} (H^*)^{L-l} \gamma_{i,t_{k,l}}.
\] (47)

On the other hand, notice that
\[
\beta_{j+1,a} = \frac{1}{2d} \sum_{j \in S_1} S_{-j} H \beta_{j,2a+j} + \gamma_{j+1,a}.
\] (48)

Therefore $\gamma_{i,t_{k,l}}$ contributes to $\beta_{L,0}$ additively. Using the definition of $t_{k,l}$, one can easily confirm that the contributions of $\gamma_{i,t_{k,l}}$ match the right hand side of equation (47). This completes the proof.

Proof of Lemma VI.1. Following equation (45), the estimation of the $l$th scale only depends on the value of $i \mod 2^{L-l}$ and $\phi(i, L-l+1)$. Therefore, the average of all shifts for the $l$th scale of the image is the same the average of all shifts in $S_{L-l+1}$ or any $S_k$ such that $k \geq L-l+1$. In other words, the shifts in $S_k$ will produce the TI estimation for the $l$th scale when $l \geq L-k+1$.

Proof of Lemma VI.3. Here we illustrate the proof when using equation (9) for $T_l$; one can easily see that the
Averaged spectral channel

Full TI

2-TI

0-TI

Fig. 3. Effect of k-TI algorithms when applying Bayesian CRP on the Chandra image. (a) is a single spectral channel (the 66th), while (b) is the average across all 256 channels. (c)-(f) are the smoothed single spectral channel images when applying 0-TI, 1-TI, 2-TI and the Full TI algorithm to the Bayesian CRP for Poisson noised images.

proof will follow the same argument when using equation (14) for \( T_i \).

Let \( G^{(L-k)}(X) \) be the aggregated estimate from scale 1 to scale \((L-k)\), i.e.

\[
G^{(L-k)}(X) = \sum_{l=0}^{L-k} (H^*)^{L-l} G_l(H^{L-l}X). \tag{49}
\]

When \( G_l \) is effectively null for \( l \leq L-k \), then \( G_l(X_i) = T_l(X) = X_i - H^*X_{i-1} \), and thus \((H^*)^{L-l} G_l(X_i) = (H^*)^{L-l}X_i - (H^*)^{L-l+1}X_{i-1} \). When \( l = 0 \), \( G_l(X_i) = X_0 \), which is \( X_{0,(1,1)} \). Therefore,

\[
G^{(L-k)}(X) = \sum_{l=1}^{L-k} (H^*)^{L-l} (X_i - H^*X_{i-1}) + X_0 = G_l(X_i) = (H^*)^kH^{k}X. \tag{50}
\]

For any circulant shift indexed by \( i \), the aggregated effects up to the \((L-k)\)th scale is

\[
(S_{-i} \circ G^{(L-k)} \circ S_i)X = (S_{-i} \circ (H^*)^k \circ H^{k} \circ S_i)X. \tag{51}
\]

Consider any shift indexed by \( i' \) such that \( i' = i + 2^k p \) for some integer vector \( p \), i.e. \( i' = i \pmod{2^k} \). Noting that \( S_{-p'} \circ (H^*)^k = S_{-k} \circ H^{k} \circ S_i \) and \( H^{k} \circ S_{-p'} = S_{-p} \circ H^{k} \circ S_i \), we have

\[
(S_{-i'} \circ (H^*)^k \circ H^{k} \circ S_{-i'})X = (S_{-i'} \circ (H^*)^k \circ H^{k} \circ S_i)X. \tag{52}
\]

Therefore, the operation of averaging all possible circulant shifts can be equivalently obtained by the one averaging all the \( k \)-TI shifts when considering scale 0 to scale \( L-k \). Note that \( k \)-TI shifts already lead to the TI estimation for scales from \((L-k+1)\) to \( L \), according to Lemma VI.1. Therefore, all the \( k \)-TI shifts lead to the full TI operator. \( \square \)

REFERENCES


