Chapter 1

Image modeling with nonlocal spectral graph wavelets

DAVID K. HAMMOND
NeuroInformatics Center
University of Oregon
Eugene, OR, USA
Email: hammond@uoregon.edu

LAURENT JACQUES
ICTEAM Institute, ELEN Department
Université catholique de Louvain
Louvain-la-Neuve, Belgium
Email: laurent.jacques@uclouvain.be

PIERRE VANDERGHEYNST
Institute of Electrical Engineering
Ecole Polytechnique Féderale de Lausanne
Lausanne, Switzerland
Email: pierre.vandergheynst@epfl.ch

local/non-local wavelet transform.

Assuming that natural images may be sparsely represented with the non-local or hybrid local/non-local wavelets yields the image model presented in this work. We examine the marginal statistics of coefficients, and show that they are may be well described by sparse Laplacian probability densities.

In this work we demonstrate the power of the graph wavelet transforms for modeling image content by applying them for image denoising. We examine two methods for image denoising, the scaled Laplacian method yielding a simple thresholding rule, and a method based on ℓ_1 minimization. The former method is based on modeling the coefficients with a Laplacian distribution, after a simple rescaling needed to account for inhomogeneity in the norms of the non-local graph wavelets. With the scaled Laplacian approach, the coefficients are thresholded and the transform is then inverted to give the denoised image. Even though this method operates only a simple operation on each coefficient independently, it gives denoising performance comparable to wavelet methods using much more complicated modeling of the dependencies between coefficients.

The closely related ℓ_1 minimization based algorithm is based on the same underlying non-local graph wavelet transform. Rather than treating each coefficient separately, it proceeds by minimizing a single convex functional containing a quadratic data fidelity term and a weighted ℓ_1 prior penalty for the coefficients. We compute the minimum of this functional using iterative an iterative Forward-Backward splitting procedure based on the proximal methods from convex analysis. We find the ℓ_1 procedure to give improvements over the scaled Laplacian method in perceived image quality, at the expense of additional computational complexity.

1.1.1 Related Work

The idea of exploiting redundancy present in images due to self-similarity has a long history. Much of the original work in fractal image compression was based on explicitly representing image self-similarity, for purposes of efficiently coding the image content [2, 3]. Quite recently, a number of image processing and denoising methods have sought to exploit image self-similarity through patch-based methods. Of particular relevance for this work is the non-local means algorithm for image denoising originally introduced by Buades et al [4].

Many extensions and variations of the original non-local means have been been studied [5, 6, 7]. The basic non-local means algorithm proceeds by first measuring the similarity between pairs of noisy image

patches by computing the ℓ_2 norms of differences of patches. These are then used to compute weights, such that the weight corresponding to two similar patches is high (close to unity), while the weight for two very dissimilar patches is close to zero. Finally, each noisy pixel is replaced by a weighted average of all other pixels, using the aforementioned weights. If a particular patch is highly similar to many other patches in the image, then it will be replaced by structure that is averaged over a large number of regions, resulting in reduction of noise. The weights used in the non-local means algorithm are essentially the same as the edge weights for the non-local image graph used in the current work.

Another approach to using patch similarity for denoising is the collaborative filtering algorithm introduced by Dabov et al [8]. In this approach, similar patches of the noisy image are stacked vertically to form a 3-D data volume. A 3-D wavelet transform is applied, then the stacked volumes are denoised by thresholding. The power of this method comes from the interaction of wavelet thresholding with the redundancy presented across the stacks of similar image patches, resulting in an implicit averaging of similar image structure across the patches.

The spectral graph wavelet transform used in this work relies on tools from spectral graph theory, specifically the use of eigenvectors of the graph Laplacian operator. Several authors have employed the graph Laplacian for image processing before. Zhang and Hancock studied image smoothing using the heat kernel corresponding to the graph laplacian, using a graph whose edge weights depended on the difference of local neighboring windows [9] . Szlam et. al. investigated similar heat kernel based smoothing in a more general context, including image denoising examples using non-local image graphs more similar to the current work [10]. Peyré used non-local graph heat diffusion for denoising, as well as studying thresholding in the basis of eigenvectors of the non-local graph Laplacian [11].

1.2 Spectral Graph Wavelet Transform

The Spectral Graph Wavelet Transform (SGWT) is a framework for constructing multiscale wavelet transforms defined on the vertices of an arbitrary finite weighted graph. The SGWT is described in detail in [1]; a brief description of it will be given here.

Weighted graphs provide an extremely flexible way of describing a large number of data domains. For example, vertices in a graph may correspond to individual people in a social network or cities connected by a transportation network of roads. Wavelet transforms on such weighted graph structures thus have the potential for broad applicability to many problems beyond the image denoising application considered in this

work. Other authors have introduced methods for wavelet-like structures on graphs. These include vertex-domain methods based on n-hop distance [12], lifting schemes [13] and methods restricted to trees [14]. These differ from the SGWT primarily in that they are based in the vertex domain, rather than constructed using spectral graph theory. Other constructions using spectral graph theory have been developed, notably the "diffusion wavelets" of Maggioni and Coifman [15]; the primary difference of the SGWT from their approach is in the simplicity of the SGWT construction, and that the SGWT produces an overcomplete wavelet frame rather than an orthogonal basis.

We consider an undirected weighted graph $\mathcal{G}$ with N vertices. Any scalar valued function $\mathbf{f}$ defined on the vertices of the graph can be associated with a vector in $\mathbb{R}^N$, where the coordinate f_i is the value of $\mathbf{f}$ on the ith vertex. For any real scale parameter $t > 0$, the SGWT will define the wavelets $\psi_{t,n} \in \mathbb{R}^N$ which are centered on each of the vertices n. Key properties of these graph wavelets are that they are zero mean, they are localized around the central vertex n, and they have decreasing support as t decreases.

The SGWT construction is based upon the graph Laplacian operator $\mathbf{L}$, defined as follows. Let $\mathbf{W} \in \mathbb{R}^{N\times N}$ be the symmetric adjacency matrix for $\mathcal{G}$, so that $w_{i,j} \geq 0$ is the weight of the edge between vertices i and j. The degree of vertex i is defined as the sum of the weights of all edges incident to i, *i.e.*, $d_i = \sum_j w_{i,j}$. Set the diagonal degree matrix $\mathbf{D}$ to have $\mathbf{D}_{i,i} = d_i$. The graph Laplacian operator is then defined to be $\mathbf{L} = \mathbf{D} - \mathbf{W}$.

The operator $\mathbf{L}$ should be viewed as the graph analogue of the standard Laplacian operator $-\Delta$ for flat Euclidean domains. In particular, the eigenvectors of $\mathbf{L}$ are analogous to the Fourier basis elements $e^{i\mathbf{k}\cdot\mathbf{x}}$, and may be used to define the graph Fourier transform. As $\mathbf{L}$ is a real symmetric matrix, it has a complete set of orthonormal eigenvectors $\chi_\ell \in \mathbb{R}^N$ for $\ell = 0, \cdots, N-1$ with associated real eigenvectors λ_ℓ. We order these in nondecreasing order, so that $\lambda_0 \leq \lambda_1 \cdots \leq \lambda_{N-1}$. For the graph Laplacian it can be shown the eigenvectors are non-negative, and that $\lambda_0 = 0$. Now for any function $\mathbf{f} \in \mathbb{R}^N$, the graph Fourier transform is defined by

$$\hat{f}(\ell) = \langle \chi_\ell, \mathbf{f} \rangle = \sum_n \chi_\ell^*(n)\mathbf{f}(n). \tag{1.1}$$

We interpret $\hat{f}(\ell)$ as the ℓth graph Fourier coefficient of $\mathbf{f}$. As the χ_ℓ's are orthonormal, it is straightforward to see that $\mathbf{f}$ may be recovered from its Fourier transform by $\mathbf{f} = \sum_\ell \hat{f}(\ell)\chi_\ell$.

The use of the graph Fourier transform for defining the SGWT may be motivated by examining the classical continuous wavelet transform in the Fourier domain. On the real line, classical wavelets are generated by taking a "mother" wavelet $\psi(x)$, and then applying scaling and translation to obtain the wavelet $\psi_{t,a} = \frac{1}{t}\psi((x-a)/t)$ [16]. For a given function f, the wavelet coefficients at scale t and location a are

then given by inner products, *i.e.*, $W_f(t,a) = \int \frac{1}{t}\psi^*((x-a)/t)f(x)dx$. For fixed scale t, this integral may be rewritten as a convolution evaluated at the point a. Letting $\bar{\psi}_t(x) = \frac{1}{t}\psi(-x/t)$, we see $W_f(t,a) = (\bar{\psi}_t \star f)(a)$. Now consider taking the Fourier transform over the a variable. By the convolution theorem this becomes

$$W_f(t,\omega) = \hat{\bar{\psi}}_t(\omega)\hat{f}(\omega) = \hat{\psi}(t\omega)\hat{f}(\omega). \tag{1.2}$$

Thus, the operation of mapping f to its wavelet coefficients $W_f(t,a)$ consists of taking the Fourier transform of f, multiplying by $\hat{\psi}(t\omega)$, and applying the inverse Fourier transform. The key point here is that the operation of scaling by t has been transferred from the original domain to a dilation of the function $\hat{\psi}(t\omega)$ in the Fourier domain. This is significant, as a major problem in adapting the classical wavelet transform to graph domains is the inherent difficulty of defining scaling of a function on an irregular graph.

The SGWT operator for fixed scale t is defined in analogy with the previous discussion. Its specific form is fixed by the choice of a non-negative wavelet kernel $g(x)$, analogous to the Fourier transformed wavelet $\hat{\psi}^*$. This kernel g should behave as a band-pass function, specifically we require $g(0) = 0$ and $\lim_{x\to\infty} g(x) = 0$. For any scale $t > 0$, the SGWT operator $T_g^t : \mathbb{R}^N \to \mathbb{R}^N$ is defined by $T_g^t(\mathbf{f}) = g(t\mathbf{L})\mathbf{f}$. Tools from spectral theory allow definition of taking a real valued function $g(t\cdot)$ of a linear operator $\mathbf{L}$, yielding a linear operator which for symmetric $\mathbf{L}$ may be defined by its action on the eigenvectors. Specifically, we have $g(t\mathbf{L})(\chi_\ell) \equiv g(t\lambda_\ell)\chi_\ell$. By linearity, applying $g(t\mathbf{L})$ to arbitrary $\mathbf{f} \in \mathbb{R}^N$ can be expressed in terms of the graph Fourier coefficients, as

$$T_g^t\mathbf{f} \equiv g(t\mathbf{L})\mathbf{f} = \sum_\ell g(t\lambda_\ell)\hat{f}(l)\chi_\ell. \tag{1.3}$$

Examining this expression, we see that applying T_g^t to $\mathbf{f}$ is equivalent to taking the graph Fourier transform of $\mathbf{f}$, multiplying by the function $g(t\lambda)$, and then inverting the transform. This is in exact analogy to the Fourier domain description of the classical wavelet transform implied by equation (1.2).

Equation (1.3) defines the N spectral graph wavelet coefficients of $\mathbf{f}$ at scale t. We denote these coefficients by $W_\mathbf{f}(t,n)$, *i.e.*, so that $W_\mathbf{f}(t,n) = (T_g^t\mathbf{f})_n$. The wavelets themselves may be recovered by localizing this operator by applying it to a delta impulse. If we set $\delta_n \in \mathbb{R}^N$ to have value 1 at vertex n and zero's elsewhere, then the wavelet centered at vertex n is given by $\psi_{t,n} = T_g^t\delta_n$. It is straightforward to verify that this is consistent both with the previous definition of the coefficients and the desire for the coefficients to be generated by inner products, *i.e.*, that $W_\mathbf{f}(t,n) = \langle \psi_{t,n}, \mathbf{f}\rangle$.

The overall stability of the SGWT is improved by including scaling functions to represent the low frequency content of the signal. This is done by introducing a scaling function kernel h, a low-pass function

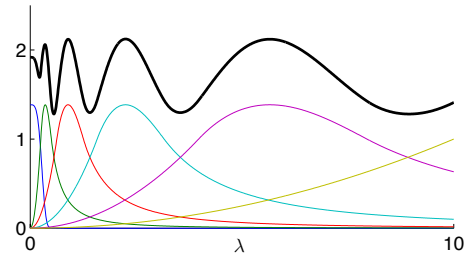


Figure 1.1: Scaling function $h(\lambda)$ (blue curve), wavelet generating kernels $g(t_j\lambda)$, and sum of squares G (black curve), for $J = 5$ scales, $\lambda_{max} = 10$, $K_{\mathrm{lp}} = 20$. Details in Section 1.2.3. Figure adapted from [1].

satisfying $h(0) > 0$ and $h(x) \to 0$ as $x \to \infty$. As with the wavelets, we define the scaling function operator $T_h = h(\mathbf{L})$, scaling functions $\phi_n = T_h\delta_n$, and the scaling function coefficients of f to be given by $T_h\mathbf{f}$.

The above theory describes the SGWT for continuous scales t. In practice, we will discretize t to a finite number of scales $t_1 > \cdots > t_J > 0$, as detailed in Section 1.2.3. Once these are fixed, we shall often abuse notation by referring to the wavelet $\psi_{j,n}$ (or coefficient $W_{\mathbf{f}}(j,n)$) as shorthand for $\psi_{t_j,n}$ (or $W_{\mathbf{f}}(t_j,n)$).

We may then consider the overall transform operator $\mathbf{T} : \mathbb{R}^N \to \mathbb{R}^{(J+1)N}$ formed by concatenating the scaling function coefficients and each of the J sets of wavelet coefficients. This operator can be shown to be a frame, with frame bounds A and B that can be estimated knowing h, g and an upper bound on the spectrum of $\mathbf{L}$. Define $G(\lambda) = h^2(\lambda) + \sum_{j=1}^{J} g^2(t_j\lambda)$. It can then be shown that for any $\mathbf{f} \in \mathbb{R}^N$, the inequalities $A\|\mathbf{f}\|^2 \leq \|\mathbf{T}\mathbf{f}\|^2 \leq B\|\mathbf{f}\|^2$ hold, where $A = \min_{\lambda\in[0,\lambda_{N-1}]} G(\lambda)$, and $B = \max_{\lambda\in[0,\lambda_{N-1}]} G(\lambda)$.

1.2.1 Fast SGWT via Chebyshev Polynomial Approximation

Direct computation of the SGWT by applying equation (1.3) requires explicit computation of the entire set of eigenvectors and eigenvalues of $\mathbf{L}$. This approach scales extremely poorly for large graphs, requiring $O(N^2)$ memory and $O(N^3)$ computational complexity. Direct computation of the SGWT through diagonalizing $\mathbf{L}$ is feasible only for graphs with fewer than a few thousand vertices. This limitation would completely destroy the practical applicability of the SGWT for image processing problems, which routinely involve data with hundreds of thousands of dimensions (*i.e.*, number of pixels).

These concerns motivate development of a fast algorithm for computing the SGWT. The fast algorithm is based on constructing degree m polynomials $p_j(x)$ which approximate the scaled wavelet kernels $g(t_jx)$ for each scale t_j. The approximate wavelet coefficients at scale j, denoted $\widetilde{W}_{\mathbf{f}}(j,n)$, are then given by computing $p_j(\mathbf{L})\mathbf{f}$ instead of $g(t_j\mathbf{L})\mathbf{f}$. An important point is that the approximation need only hold on the spectrum of $\mathbf{L}$. We take the polynomials $p_j(x)$ to be the truncated Chebyshev polynomial expansion for $g(t_jx)$, on the

interval $[0, \lambda_{N-1}]$.

Note that as we may write $\mathbf{L}^n\mathbf{f} = \mathbf{L}(\cdots\mathbf{L}(\mathbf{L}\mathbf{f})\cdots)$, one may compute the application of any polynomial of $\mathbf{L}$ to a vector $\mathbf{f}$ using only matrix-vector multiplication. It is this fact which makes the polynomial approximation algorithm for the SGWT efficient in the important case when the $\mathbf{L}$ is sparse. Details of the algorithm are given in [1]. In addition it is shown there that the computational complexity for the fast SGWT is $O(m|E| + Nm(J+1))$, where J is the number of wavelet scales, m is the order of the polynomial approximation, and $|E|$ is the number of nonzero edges in the underlying graph $\mathcal{G}$. In particular, for classes of graphs where $|E|$ scales linearly with N, such as graphs of bounded maximal degree, the fast SGWT has computational complexity $O(N)$.

1.2.2 SGWT Inverse

A wide class of signal processing applications, including denoising methods described later in this work, involve manipulating the coefficients of a signal in a certain transform, and later inverting the transform. For the SGWT to be useful for more than simply signal analysis, it is important to be able to recover a signal corresponding to a given set of coefficients.

The SGWT is an overcomplete transform, mapping an input vector $\mathbf{f}$ of size N to the $N(J+1)$ coefficients $\mathbf{c} = \mathbf{T}\mathbf{f}$. As is well known, this means that $\mathbf{T}$ will have an infinite number of left-inverses $\mathbf{B}$ s.t. $\mathbf{B}\mathbf{T}\mathbf{f} = \mathbf{f}$. A natural choice is to use the pseudoinverse $\mathbf{L} = (\mathbf{T}^T\mathbf{T})^{-1}\mathbf{T}^T$, which satisfies the minimum-norm property

$$\mathbf{L}\mathbf{c} = \underset{\mathbf{f}\in\mathbb{R}^N}{\arg\min} \|\mathbf{c} - \mathbf{T}\mathbf{f}\|_2 .$$

For applications which involve manipulation of the wavelet coefficients, it is very likely to need to apply the inverse to a a set of coefficients which no longer lie directly in the image of $\mathbf{T}$. The above property indicates that, in this case, the pseudoinverse corresponds to orthogonal projection onto the image of $\mathbf{T}$, followed by inversion on the image of $\mathbf{T}$.

Given a set of coefficients $\mathbf{c}$, the pseudoinverse will be given by solving the square matrix equation $(\mathbf{T}^T\mathbf{T})\mathbf{f} = \mathbf{T}^T\mathbf{c}$. This system is too large to invert directly, but may be solved iteratively using conjugate gradients. The computational cost of conjugate gradients is dominated by matrix-vector multiplication by $\mathbf{T}^T\mathbf{T}$ at each step. As described further in [1], the fast Chebychev approximation scheme described in 1.2.1 may be adapted to compute efficiently the application of either $\mathbf{T}^T$ or $\mathbf{T}^T\mathbf{T}$. We use conjugate gradients, together with the fast Chebyshev approximation scheme, to compute the SGWT inverse.

1.2.3 SGWT design details

The SGWT framework places very few restrictions on the choice of the wavelet and scaling function kernels. Motivated by simplicity, we choose g to behave as a monic power near the origin, and to have power law decay for large x. In between, we set g to be a cubic spline such that g and g' are continuous. Specifically, in this work we used

$$g(x) = \begin{cases} x^2 & \text{for } x < 1, \\ -5 + 11x - 6x^2 + x^3 & \text{for } 1 \leq x \leq 2, \\ x^{-2} & \text{for } x > 2. \end{cases} \tag{1.4}$$

The wavelet scales t_j (with $t_j > t_{j+1}$) are selected to be logarithmically equispaced between the minimum and maximum scales t_J and t_1. These are themselves adapted to the upper bound λ_{max} of the spectrum of $\mathbf{L}$. The placement of the maximum scale t_1 as well as the scaling function kernel h will be determined by the selection of $\lambda_{min} = \lambda_{max}/K_{\text{lp}}$, where K_{lp} is a design parameter of the transform. We then set t_1 so that $g(t_1 x)$ has power-law decay for $x > \lambda_{min}$, and set t_J so that $g(t_J x)$ has monic polynomial behaviour for $x < \lambda_{max}$. This is achieved for $t_1 = 2/\lambda_{min}$ and $t_J = 2/\lambda_{max}$. For the scaling function kernel we take $h(x) = \gamma \exp(-(\frac{x}{0.6\lambda_{min}})^4)$, where γ is set such that $h(0)$ has the same value as the maximum value of g. This set of scaling function and wavelet generating kernels, for parameters $\lambda_{max} = 10$, $K_{\text{lp}} = 20$, and $J = 5$, are shown in Figure 1.1.

1.3 Non-Local Image Graph

The non-local image graph is a weighted graph with number of vertices N equal to the number of pixels in the original image. This association assumes some labeling of the two-dimensional pixels, they may for example be labeled in raster-scan order. We will define the edge weights so that the weights will lie between 0 and 1, and will provide a measure of the similarity between image patches.

The number of pixels N for any reasonable sized image will be so large that we cannot reasonably expect to explicitly diagonalize the corresponding graph Laplacian. As we wish to use the SGWT defined on the non-local graph, we must produce a graph which is sparse enough to be handled by the fast Chebychev polynomial approximation scheme.

Given a patch radius K, we let $\mathbf{p}_i \in \mathbb{R}^{(2K+1)^2}$ be the $(2K+1) \times (2K+1)$ square patch of pixels centered on pixel i. For pixels within distance K of the image boundary, patches may be defined by extending the

image outside of its original boundary by reflection. Set $d_{i,j} = \|\mathbf{p}_i - \mathbf{p}_j\|$ to be the norm of the differences of patches. Following the weighting used in much of the NL-means literature, we set the edge weights for the (unsparsified) non-local image graph to be $w_{i,j} = \exp(-d_{i,j}/2\sigma_p^2)$.

We will let $\mathbf{W}^{\text{nl}}$ denote the adjacency matrix for the sparsified non-local image graph. Our sparsification procedure will form $\mathbf{W}^{\text{nl}}$ by systematically setting most of the edge weights $w_{i,j}$ to zero. One reasonable way to do this would be by hard thresholding, *i.e.*, to fix $\eta > 0$ and set $\mathbf{W}^{\text{nl}}_{i,j} = w_{i,j}$ if $w_{i,j} \geq \eta$, or zero otherwise. However, we have observed that hard thresholding results in very large variations in the degrees of the vertices of the resulting sparsified graph. There is no simple way to control the minimum degree for the vertices, so it may happen that for a certain value of η some vertices have very large degree while other vertices may not have any incident edges at all.

Instead, we proceed by keeping a fixed minimum number M of edges incident to each vertex, as follows. For each patch i, we sort the elements $w_{i,j}$ in decreasing order. In other words, we find $j_1, j_2, \cdots, j_N$ so that $w_{i,j_1} \geq w_{i,j_2} \cdots \geq w_{i,j_N}$. We wish to keep the edges $(i, j_1), (i, j_2), \cdots (i, j_M)$. For convenience, denote this set of the M neighbors with largest edge weights by $\mathcal{N}(i) \equiv \{j_1, \cdots j_M\}$. Note that we exclude the possibility of retaining i as its own neighbor, *i.e.*, $i \notin \mathcal{N}(i)$.

Once these vertex sets $\mathcal{N}(i)$ are computed for each $1 \leq i \leq N$, $\mathbf{W}^{\text{nl}}$ is given by

$$\mathbf{W}^{\text{nl}}_{i,j} = \begin{cases} w_{i,j} \text{ if } i \in \mathcal{N}(j) \text{ or } j \in \mathcal{N}(i) \\ 0 \text{ otherwise} \end{cases}$$

After this procedure, each vertex of $\mathbf{W}^{\text{nl}}$ will have at least M nonzero edges incident to it, and the total number of nonzero edges will be bounded by $2NM$. Note that due to the symmetrization of the matrix, it is possible for vertices to have more than M incident nonzero edges.

For convenience in the sequel, we denote by $\mathcal{D}(i)$ the ordered set of M patch distances $\{d_{i,j} \text{ for } j \in \mathcal{N}(i)\}$. As the function $\exp(-\frac{\cdot}{2\sigma_p})$ determining $w_{i,j}$ from $d_{i,j}$ is decreasing, the elements of $\mathcal{D}(i)$ consist of the M smallest patch distances from the patch at vertex i. Note that the sets $\mathcal{N}(i)$ and $\mathcal{D}(i)$ do not depend on the variance parameter σ_p.

The parameter σ_p is fixed by specifying the desired mean of all of the nonzero elements of $\mathbf{W}^{\text{nl}}$. Given a target mean $\mu_{\text{NL}} \in (0,1)$, σ_p is set so that

$$\frac{1}{\#\{\mathbf{W}^{\text{nl}}_{i,j} \neq 0\}} \left(\sum_{i,j} \mathbf{W}^{\text{nl}}_{i,j} \right) = \mu_{\text{NL}}$$

We chose to determine σ_p this way, rather than simply fixing a numerical value for it, in order to allow automatic selection of an appropriate σ_p as the input image and other parameters are changed (such as the patch size K). In practice the value $\mu_{\mathrm{NL}} = 0.75$ gives good results.

1.3.1 Acceleration of computation with seeds

Naive computation of the sets $\mathcal{N}(i)$ and $\mathcal{D}(i)$ could be done simply by looping over i, computing the entire set of patch distances $d_{i,j}$, sorting them, and retaining the smallest M. This is computationally intensive. In this section we describe an accelerated algorithm for more efficiently computing these sets. We note that a number of other authors have explored various strategies for reducing the computational cost of the closely related non-local means filter. Representative approaches include dimensionality reduction of raw patches via PCA [17], and structured organization of patches into a cluster tree [18].

Our "seeds acceleration" algorithm proceeds by exploiting the geometry of the patch space $\mathbb{R}^{(2k+1)^2}$ to get lower bounds on distances between patches. These lower bounds will depend on the selection of a set of seed patches to which all distances will be computed. By using the lower bounds to show on the fly that certain patch distances will not appear amongst the M smallest, we will cull the need to compute them. We note that the seeds acceleration algorithm is not an approximate algorithm, and will produce exactly the same edge and distance sets as the naive algorithm (unless there are ties in the distances, which may be broken differently).

Let $\mathcal{S} = \{\mathbf{s}_1, \cdots, \mathbf{s}_F\}$ be a set of F seeds, with each $\mathbf{s}_k \in \mathbb{R}^{(2K+1)^2}$. We first compute and store the NF values $\xi_{i,k} = \|\mathbf{p}_i - \mathbf{s}_k\|$. We then employ the triangle inequality, which shows $\|\mathbf{p}_i - \mathbf{s}_k\| \leq \left\|\mathbf{p}_i - \mathbf{p}_j\right\| + \left\|\mathbf{p}_j - \mathbf{s}_k\right\|$, and $\left\|\mathbf{p}_j - \mathbf{s}_k\right\| \leq \left\|\mathbf{p}_j - \mathbf{p}_i\right\| + \|\mathbf{p}_i - \mathbf{s}_k\|$. These in turn imply

$$\left|\left(\|\mathbf{p}_i - \mathbf{s}_k\| - \left\|\mathbf{p}_j - \mathbf{s}_k\right\|\right)\right| \;\leq\; \|\mathbf{p}_i - \mathbf{p}_j\| \;\equiv\; d_{i,j} \tag{1.5}$$

Using this inequality for all $k \leq J$ shows

$$\max_k(|\xi_{i,k} - \xi_{j,k}|) \;\leq\; d_{i,j}$$

Denote this quantity $\overline{d}_{i,j} = \max_k(|\xi_{i,k} - \xi_{j,k}|)$. Computing $\overline{d}_{i,j}$ for fixed i, j is very fast (complexity $O(F)$) as it requires no floating point operations other than absolute value and comparison. Crucially, it is significantly faster than directly computing $d_{i,j}$.

We now consider computation of all of the index and distance sets $N(i)$ and $D(i)$. The naive algorithm without using seeds would be Algorithm 1 in Table 1.1.

Algorithm 1: Naive Algorithm	**Algorithm 2:** Seeds acceleration : Implicit sorting with heap ($\mathcal{H}$) and pruning by lower bounds

for $i \in \{1, \cdots, N\}$ **do**
  for $j \in \{1, \cdots, N\} \setminus \{i\}$ **do**
  Compute $d_{i,j}$
  $L(j) \leftarrow d_{i,j}$
  end for
  Sort the list L in increasing order,
  keeping track of indices
  $\mathcal{N}(i) \leftarrow$ first M indices
  $\mathcal{D}(i) \leftarrow$ first M distance values
end for
	Select F seeds $\mathcal{S} = \{\mathbf{s}_k : 1 \leq k \leq F\}$
	Precompute the $\{\xi_{i,k} : 1 \leq i \leq N, 1 \leq k \leq F\}$
	for $i \in \{1, \cdots, N\}$ **do**
	  $\mathcal{H} \leftarrow \emptyset$
	  for $j \in \{1, \cdots, N\} \setminus \{i\}$ **do**
	  if $\lvert\mathcal{H}\rvert < M$ **then**
	  Compute $d_{i,j}$
	  $\mathcal{H} \leftarrow \mathcal{H} \cup \{d_{i,j}\}$
	  else
	  Compute $\overline{d}_{i,j}$
	  if $\overline{d}_{i,j} \leq \max\{\mathcal{H}\}$ **then**
	  Compute $d_{i,j}$
	  $\mathcal{H} \leftarrow M$ smallest values of $\mathcal{H} \cup \{d_{i,j}\}$
	  end if
	  end if
	  end for
	  Sort $\mathcal{H}$ in increasing order, keeping track of indices
	  $\mathcal{N}(i) \leftarrow$ first M indices
	  $\mathcal{D}(i) \leftarrow$ first M distance values
	end for

Table 1.1: Naive and seeds accelerated algorithms

This is wasteful, though, as the entire list L is sorted even though only the M smallest elements are needed. Instead, we can only keep track of the smallest M elements while looping through j. This can be done efficiently using a heap data structure. We are now in a position to use seeds to prune unnecessary computation. If the computed lower bound ensures that $d_{i,j}$ will be larger than the current heap maximum, then its computation may be suppressed. This yields the accelerated Algorithm 2 in Table 1.1.

The performance of the accelerated algorithm is highly dependent on the selection of the seed points $\mathbf{s}_k$. For any patch $\mathbf{p}_i$, the lower bounds (1.5) on $d_{i,j}$ given by seed $\mathbf{s}_k$ will be more informative the closer $\mathbf{s}_k$ is to $\mathbf{p}_i$. This implies that a set of seed points will be good if the seeds are near as many patches as possible.

		Lena	Barbara	Boats	House
maxmin	ρ	5.2 %	4.9 %	12.5 %	14.3 %
	runtime	89.07 s	87.77 s	111.67 s	121.02 s
random	ρ	6.1 %	6.0 %	13.1 %	14.9 %
	runtime	92.71 s	92.31 s	112.12 s	122.44 s
naive	runtime	425.16 s	430.84 s	434.08 s	425.87 s

Table 1.2: Performance of seeds acceleration algorithm, averaged over 3 trials for $K = 5$ and $M = 10$. Runtimes (in seconds) were measured on a parallel implementation using 4 Intel Xeon 2.0 GHz cpu cores.

On the other hand, if two seeds are close to each other, the information they provide in their lower bounds is redundant. This implies the contrasting criterion that the seeds should be far from each other.

Motivated by these two considerations, we introduce a maxmin heuristic for selecting seed points. We select the first seed s_1 to be the patch corresponding to the very middle of the image. We then compute and store the N values $\xi_{i,1} = \|\mathbf{p}_i - \mathbf{s}_1\|$. The next seed $\mathbf{s}_2$ is picked as the point furthest from $\mathbf{s}_1$, *i.e.*, $s_2 = \arg\max_i \xi_{i,1}$. At each successive step, the $k+1^{\text{th}}$ seed is picked to be the point maximizing the distance to the set $\{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_k\}$, *i.e.*,

$$\mathbf{s}_{k+1} \;=\; \arg\max_i \left(\min_{k' \leq k} \xi_{i,k'} \right).$$

This procedure is iterated until all F seeds are computed. Note that the precomputed distances $\xi_{i,k}$ are produced during the execution of this heuristic.

We have found the performance of the seed acceleration algorithm using seeds computed the above maxmin heuristic to be slightly better than using the same number of randomly selected seeds. Even using randomly selected seeds, however, provides a marked improvement over the naive algorithm.

The performance of the seeds acceleration can be quantified by the ratio ρ of patch distances $d_{i,j}$ computed by the accelerated algorithm divided by the number of patch distances computed by the naive algorithm. Alternatively, one may directly examine the reduction in run time. The overall run time will not be reduced as much as the number of patch distance computations, reflecting the additional overhead of computing the lower bounds $\overline{d}_{i,j}$. Given ρ, a quick computational complexity estimation provides that Algorithm 2 runs in $O\big(NM(2K+1)^2 + (N-M)NF + \rho\,(N-M)N(2K+1)^2\big)$. Considering that the proportionality constant are the same in Algorithms 1 and 2, the ratio of computational time is about $\rho + \frac{F}{(2K+1)^2} + \frac{M}{N}$. Table 1.2 approximately follows this rough estimation.

In practice, on 256×256 clean test images with patch radius $K = 5$, the seeds acceleration algorithm using 15 seeds allows reduction of run time by a factor of 3-8 compared to the naive algorithm for computing the sparsified adjacency matrix, as shown in Table 1.2. We note that the performance improvement is very dependent on the structure of the input image. In particular, when testing on images of random noise, computation time is not reduced at all! This makes sense as the lower bounds are useful only if seeds are near where "most" of the image patches are, for random images the patches are too distributed.

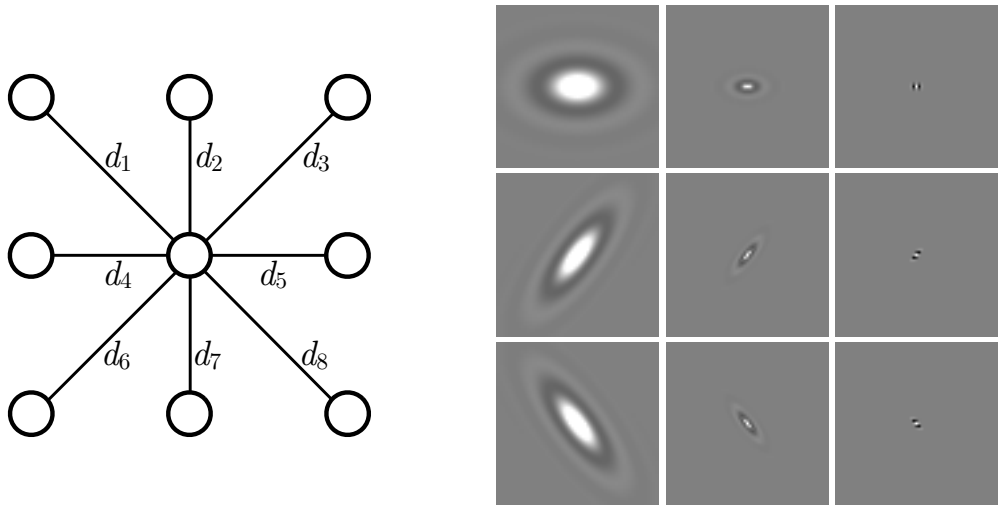
1.4 Hybrid local/non-local image graph

As the spectral graph wavelet transform is parametrized by the underlying weighted graph, great flexibility is available for influencing the wavelets by choosing the design of the graph. While much of the emphasis of this paper is on construction of the non-local image graph, one may also construct local graph wavelets simply by using a connectivity graph that includes only local connections vertices. Given both the non-local adjacency $\mathbf{W}^{\mathrm{nl}}$ and a local adjacency $\mathbf{W}^{\mathrm{loc}}$, we may form a hybrid local/non-local graph simply by linearly interpolation, *e.g.*, $\mathbf{W}^{\mathrm{hyb}}(\lambda) = \lambda \mathbf{W}^{\mathrm{loc}} + (1-\lambda)\mathbf{W}^{\mathrm{nl}}$. Here λ smoothly parametrizes the degree of locality present in the graph, so that at $\lambda = 0$ the graph is purely non-local, while at $\lambda = 1$ it is purely local. The structure of the SGWT makes it extremely easy to explore the effects of such a hybrid connectivity. As long as both the local and non-local adjacencies are sufficiently sparse to allow the fast Chebyshev transform to work efficiently, no other changes need be made to any other parts of the SGWT machinery.

1.4.1 Oriented Local Connectivity

It is well known that for image processing applications, oriented wavelet filters are much more effective than spatially isotropic filters. This property arises because much of the significant content in images is highly oriented, such as edges. This observation motivates the design of a local adjacency graph that would give oriented filters when used with the SGWT. The standard 4-point local connectivity, with equal edge weights in all directions, gives rise to spectral graph wavelets which are isotropic, and thus unlikely to be highly efficient as a sparsity basis.

If oriented filters are used, it is natural to have a set of orientations which are evenly distributed on $[0, \pi]$. We will generate oriented local graph wavelets by choosing a local connectivity which is itself oriented. The design of these oriented local connectivities will be determined by requiring that the associated graph Laplacian approximate a continuous *oriented* second derivative operator.



(a) (b)

Figure 1.2: (a) 8 nearest neighbor connectivity. The complete graph is produced by reproducing these edges across the entire image. (b) Spectral graph wavelets computed using local oriented adjacencies $\mathbf{W}^{\text{loc}}_{\theta,\delta}$. Shown are scaling functions and wavelets from selected scales for transform with $J = 20$ wavelet scales, $S = 3$ orientations,using $\delta = 0.44$. Top row, $\theta = 0$, middle row, $\theta = \pi/3$, bottom row, $\theta = 2\pi/3$. Left column, scaling function; middle column, $j = 10$, right column, $j = 20$.

We describe the target continuous operator in terms of both a dominant orientation parameter θ, and a second parameter $\delta \in [0,1]$ specifying the degree of anisotropy, or "orientedness". For any angle θ, let the perfectly oriented oriented second derivative in the θ direction be D^2_θ, so that $(D^2_\theta f)(x,y) \equiv \frac{d^2}{d\varepsilon^2} f(x + \varepsilon\cos\theta, y + \varepsilon\sin\theta)|_{\varepsilon=0}$ for any (x,y) in the plane. For any $\delta < 1$, the target operator $D(\theta,\delta)$ will include some fraction of the second derivative in the perpendicular direction. Specifically, we define

$$D(\theta,\delta) = \tfrac{1+\delta}{2} D^2_\theta + \tfrac{1-\delta}{2} D^2_{\theta+\pi/2}.$$

This operator can be expressed in terms of partial derivatives in the standard coordinate directions. A straightforward calculation shows

$$D(\theta,\delta)f \;=\; \tfrac{1}{2}(1+\delta\cos 2\theta)\, f_{xx} \;+\; (\delta\sin 2\theta)\, f_{xy} \;+\; \tfrac{1}{2}(1-\delta\cos 2\theta)\, f_{yy}. \tag{1.6}$$

We choose the local connectivity $\mathbf{W}^{\text{loc}}_{\theta,\delta}$ to be the 8-point connectivity with weights such that the graph Laplacian approximates $D(\theta,\delta)$. The local connectivity will be completely specified by the values of the weights of any vertex to its 8 nearest neighbors. For convenience, these may be labeled $d_1, d_2, \cdots, d_8$, as shown in Figure 1.2 (a). As we are considering undirected graphs, we must have equality of edge weights

for edges in opposite directions, *e.g.*, $d_1 = d_8$, $d_2 = d_7$, $d_3 = d_6$, $d_4 = d_5$. We may consider the first four weights as components of the vector $\mathbf{d} \in \mathbb{R}^4$.

We may match the discrete graph Laplacian implied by the choice of $\mathbf{d}$ to the continuous operator $D(\theta,\delta)$ by looking at the local Taylor expansion. We now consider f to be a continuous function which is sampled on the image pixel lattice. Let f_i for $0 \le i \le 8$ denote the value of f at the vertices shown in Figure 1.2, and let x_i and y_i be the integer offsets for vertex i, *i.e.*, $(x_1,y_1) = (-1,1)$, $(x_2,y_2) = (0,1)$, etc. If Δ is the grid spacing, using the Taylor expansion to second order shows

$$f_i - f_0 \;=\; \Delta\,(x_i f_x + y_i f_y) \;+\; \tfrac{1}{2}\Delta^2\,(x_i^2 f_{xx} + 2x_i y_i f_{xy} + y_i^2 f_{yy}) \;+\; o(\Delta^3).$$

By above, applying the graph Laplacian to f at the center vertex provides

$$\sum_i d_i(f_i - f_0) \;=\; \Delta^2(f_{xx}\sum_i \tfrac{1}{2}\,d_i x_i^2 \;+\; f_{xy}\sum_i d_i x_i y_i \;+\; f_{yy}\sum_i \tfrac{1}{2}\,d_i y_i^2) + o(\Delta^3). \tag{1.7}$$

We may take the grid spacing to be unity, *i.e.*, $\Delta = 1$. Equating the coefficients of f_{xx}, f_{xy} and f_{yy} in (1.6) and (1.7) yields

$$\begin{cases} d_1 + d_3 + d_4 & = \frac{1}{2}(1+\delta\cos 2\theta) \\ \quad -d_1 + d_3 & = \frac{1}{2}\delta\sin 2\theta \\ d_1 + d_2 + d_3 & = \frac{1}{2}(1-\delta\cos 2\theta) \end{cases} \quad\Leftrightarrow\quad \mathbf{M}\,\mathbf{d} \;=\; \mathbf{v}(\theta,\delta), \tag{1.8}$$

where we set $\mathbf{M} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ -1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$ and $\mathbf{v} = (1/2,0,1/2)^T + \frac{\delta}{2}(\cos\theta/2, \sin\theta/2, -\cos\theta/2)^T$. This is an underdetermined linear system for the four free components of $\mathbf{d}$. A simple procedure for calculating $\mathbf{d}$ would be to use least-squares. However, the graph weights must be non-negative. Including this constraint yields a convex optimization program for $\mathbf{d}$:

$$\mathbf{d}^*(\theta,\delta) = \arg\min_{\mathbf{d}} \;\|\mathbf{d}\|^2 \text{ subject to } \mathbf{M}\mathbf{d} = \mathbf{v}(\theta,\delta), \text{ and } d_i \ge 0 \tag{1.9}$$

For δ close to 1, the program (1.9) fails to be feasible for angles θ away from aligned with the coordinate axes. We have found that the program is feasible for all θ whenever $\delta < \widetilde{\delta} \approx 0.44$. We solve the program (1.9) numerically using the MATLAB cvx toolbox [19]. Using these values of $\mathbf{d}(\theta,\delta)$ to construct the weighted graph for the entire image yields the local connectivity $\mathbf{W}^{\mathrm{loc}}_{\theta,\delta}$.

For completeness, we show several images of the spectral graph wavelets computed using $\mathbf{W}^{\mathrm{loc}}_{\theta,\delta}$ in Figure 1.2(b). The important point of these local oriented wavelets is not necessarily that we expect them to be highly effective for image representation on their own, but rather that they allow the introduction of oriented filter character in a manner compatible with the SGWT construction.

1.4.2 Unions of Hybrid Local/Non-Local SGWT frames

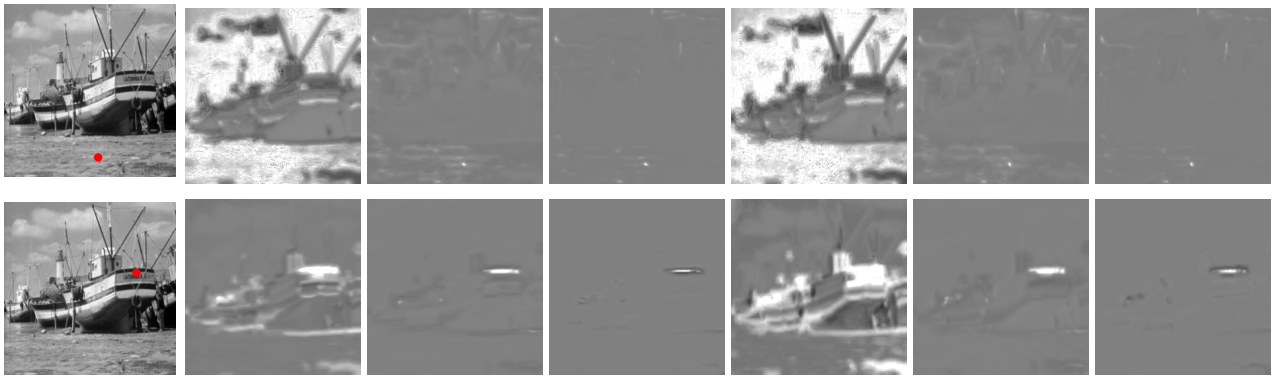
The SGWT generates a frame of wavelets and scaling functions from a single adjacency matrix. Using a hybrid adjacency with an oriented local connectivity would give a single frame where all of the wavelets had the same directionality. Using such a frame for image restoration tasks is likely to introduce a bias towards orientations in this specific direction. We would prefer a wavelet transform that samples orientations uniformly on $[0,\pi]$. A straightforward way to do this is to form the union of multiple SGWT frames corresponding to hybrid adjacencies with uniformly sampled orientations. Let $\mathbf{T}^{\theta,\delta} : \mathbb{R}^N \to \mathbb{R}^{N(J+1)}$ be the SGWT operator generated using the hybrid adjacency matrix $\mathbf{W}^{\text{hyb}}_{\theta,\delta} \equiv \lambda \mathbf{W}^{\text{loc}}_{\theta,\delta} + (1-\lambda)\mathbf{W}^{\text{nl}}$. Denote by S the number of orientation directions to sample, and set $\theta_k = \frac{k-1}{S\pi}$ for $1 \leq k \leq S$. We then define the overall wavelet transform operator $\mathbf{T} : \mathbb{R}^N \to \mathbb{R}^{NS(J+1)}$ by $\mathbf{T}\mathbf{f} = ((\mathbf{T}^{\theta_1,\delta}\mathbf{f})^T, (\mathbf{T}^{\theta_2,\delta}\mathbf{f})^T, \cdots, (\mathbf{T}^{\theta_S,\delta}\mathbf{f})^T)^T$.

While in principle separate scaling function and wavelet kernels h and g could be used for the separate component SGWT frames, we use the same kernels as well as the same sampled scales t_j for each of the S sub-frames. As the selection of scales is determined by the spectrum of the graph Laplacian (as described in Section 1.2), choosing the scales uniformly only makes sense if the maximal eigenvalues of the S graph Laplacians formed from each of the $\mathbf{W}^{\text{hyb}}_{\theta_k,\delta}$ are similar. In practice we find this to be the case. We note finally that the union of frames transformation is itself a frame. If each of the component frames satisfy the frame bounds A and B, then the union of frames transformation will have frame bounds SA and SB.

We remark that the hybridization with λ may also be described directly with the graph Laplacian operators, i.e. we may write $\mathbf{L}^{\text{hyb}}_{\theta_k,\delta} = \lambda \mathbf{L}^{\text{loc}}_{\theta_k,\delta} + (1-\lambda)\mathbf{L}^{\text{nl}}$. One issue with this parametrization with λ is that, depending on the parameters of the graph construction, the operator norm of $\mathbf{L}^{\text{nl}}$ may be very different in magnitude than that of the $\mathbf{L}^{\text{loc}}_{\theta_k,\delta}$'s. This implies, for example, that taking $\delta = 0.5$ will not really give an equal mixing of local and nonlocal character. For convenience, we introduce the parameter λ'. We wish the effective nonlocal contribution to be proportional to λ', and the effective local contribution to be proportional to $1-\lambda'$. We enforce this by determining λ so that

$$
\begin{aligned}
(1-\lambda)\,\|\mathbf{L}^{\text{nl}}\| &= (1-\lambda')C \\
\lambda\,\|\mathbf{L}^{\text{loc}}\| &= \lambda' C
\end{aligned}
$$

for some constant C. In the above we have suppressed the dependence on the angle θ_k for $\|\mathbf{L}^{\text{loc}}\|$, in practice we find there to be little dependence on k and we may just take $\|\mathbf{L}^{\text{loc}}\| = \max_k \|\mathbf{L}^{\text{loc}}_{\theta_k,\delta}\|$. Solving the above



original | s.f., $\theta = 0$ | $j = 3$, $\theta = 0$ | $j = 6$, $\theta = 0$ | s.f., $\theta = \pi$ | $j = 3$, $\theta = \pi$ | $j = 6$, $\theta = \pi$

Figure 1.3: Images of the graph wavelets with hybrid local/non-local adjacency, for boats image. Shown are scaling functions and wavelets from selected scales for transform with $J = 20$ wavelet scales, $S = 2$ orientations,using $\delta = 0.5$, for two different wavelet centers. Red dots on original image indicate wavelet center locations.

set of equations gives

$$\lambda = \frac{\|\mathbf{L}^{\mathrm{nl}}\|}{(\frac{1-\lambda'}{\lambda'})\|\mathbf{L}^{\mathrm{loc}}\| + \|\mathbf{L}^{\mathrm{nl}}\|}$$

Finally, for convenience of notation we note the following. Coefficients of the graph wavelet transforms formed from a single SGWT frame may be indexed by scale parameter j and location index n, while coefficients of the hybrid local/non-local wavelet transforms require an additional orientation index k. In the sequel, we will often replace the scale and orientation indices by a single multi-index β associated to one choice of (t_j, θ_k). This allows us to refer to the wavelet $\psi_{\beta,n}$ and coefficient $x_{\beta,n}$ at (scale and/or orientation) band β, position n, independent of the specific form of the transform.

1.4.3 Image Graph Wavelets

As an illustrative example of what the resulting hybrid local/non-local image graph wavelets look like, we display several in Figure 1.3 computed for the well known "boats" test image. These wavelets were computed using the same parameters as are later used in our denoising applications. A relatively small value of $\lambda' = 0.15$ was used, so they are mostly non-local spiced with a small pinch of local character. Note that the wavelets really are non-local - this is especially evident with the scaling functions which show support on parts of the image very distant from the wavelet center. It is also evident that the support for both the scaling functions and wavelets consists of image regions that are similar in structure to the center patch. Note for the top row where the center patch is on the ground how the support of the scaling functions are

mostly ground and similar blank sky, while for the bottom row the supports consist of regions with strong horizontal structure similar to the patch center. It can also be verified that for increasing j (corresponding to decreasing scale t_j), the wavelets are increasingly localized. For even smaller scales (not pictured), this localization proceeds further.

1.5 Scaled Laplacian Model

Coefficients of natural signals in common localized bases (*e.g.*, (bi)orthogonal wavelets, wavelet frames, steerable wavelets, curvelets, ...) typically exhibit sparse behavior. This has commonly been modeled by using peaked, heavy-tailed distributions as prior models for coefficients. A canonical example of this is the Laplacian distribution $p(x) = \frac{1}{2s}\exp(\frac{-|x|}{s})$. Laplacian models have been very commonly used in the literature to describe the marginal statistics of wavelet coefficients of images, *e.g.*, [20, 21, 22].

Typically, the parameter s is allowed to depend on the wavelet scale. Including this is necessary to allow the variance of the signal coefficients to depend on scale. This effect arises from the power spectral properties of the original image signal (typically showing larger power at low frequencies) . It may also arise from normalization of the wavelets themselves, as for many overcomplete wavelet transforms the norms of the wavelets may depend on the wavelet scale.

Unlike classical wavelet transforms which are translation invariant at each scale (*i.e.*, all wavelets at a particular subband are translates of a single waveform), the spectral graph wavelets centered at different vertices, for a particular wavelet scale t, do not all have the same norm. As the coefficients $c_{\beta,n} = \langle \psi_{\beta,n}, f \rangle$ are linear in the norm of each wavelet, it is problematic to model the coefficients at one scale with a single Laplacian density. Allowing the Laplacian parameter s to vary independently at each vertex location n would give a model with too many free parameters that we would not be able to fit.

These considerations motivate the scaled Laplacian model, where we model each coefficient as Laplacian with a parameter that is proportional to the wavelet norm. This proportionality constant is allowed to depend on the wavelet band β, giving one free parameter for the model at each wavelet band. The model is

$$\mathrm{p}_{\beta,n}(x) = \frac{1}{2\alpha_j \sigma_{\beta,n}} \exp\left(\frac{-|x|}{\alpha_\beta \sigma_{\beta,n}}\right)$$

where $\sigma_{\beta,n} = \left\|\psi_{\beta,n}\right\|$ is the norm of the wavelet at band β and vertex n.

This model implies that at each band, the marginal distribution of the coefficients rescaled by $1/\sigma_{\beta,n}$ should have a Laplacian distribution. Support for the appropriateness of the scaled Laplacian model can be

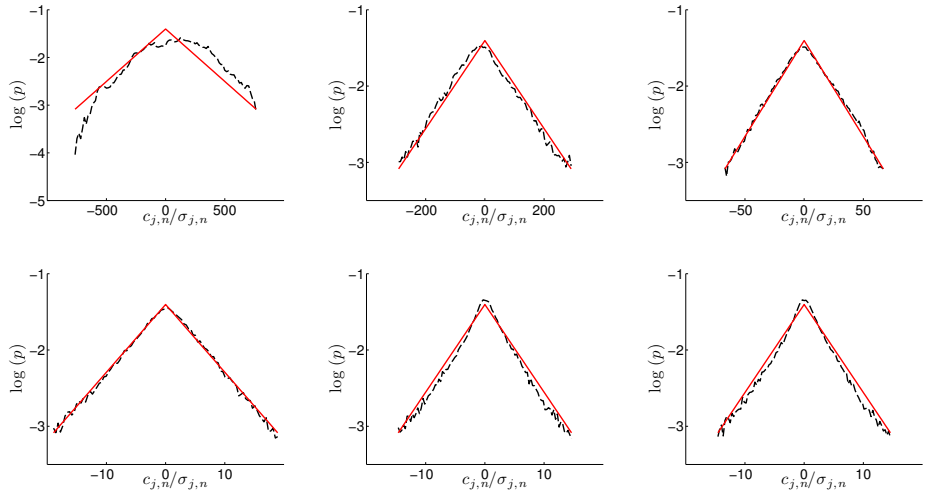


Figure 1.4: Log-histograms of rescaled graph wavelet coefficients, from from the purely non-local transform with 6 wavelet scales, for clean "boats" image. From top left to bottom right are wavelet scales j=1 through j=6. Solid red lines show best Laplacian fit.

found by examining the marginal distributions of rescaled coefficients. We find these marginal distributions are qualitatively well fit by a Laplacian, as shown in Figure 1.4. These are shown for the purely non-local graph wavelet transform, but we observe very similar results for the hybrid local/non-local transforms.

1.6 Applications to Image Denoising

As an illustration of the power of the non-local and hybrid graph wavelets for modeling image content, we study the problem of removing noise from natural images. In this work we consider additive Gaussian white noise. We fix our notation as follows. We let $\mathbf{x}, \mathbf{y}$ and $\mathbf{n}$ denote the (unknown) clean image, noisy image and noise respectively. This is consistent with the additive noise model $\mathbf{y} = \mathbf{x} + \mathbf{n}$, where $\mathbf{n}$ is a zero-mean white Gaussian noise with variance σ_I^2 in the pixel domain. Our denoising algorithms are described in wavelet space. Let $\mathbf{T}$ be the graph wavelet operator, which will be either the SGWT with non-local Laplacian, or a union of SGWT frames for the hybrid local/non-local adjacencies described in 1.4.2. We will use the notation $\mathbf{c}$, $\mathbf{d}$, and $\mathbf{e}$ to refer to the wavelet coefficients of the clean image, noisy image and noise process, respectively, so that $\mathbf{c} = \mathbf{T}\mathbf{x}$, $\mathbf{d} = \mathbf{T}\mathbf{y}$ and $\mathbf{e} = \mathbf{T}\mathbf{n}$.

1.6.1 Construction of non-local image graph from noisy image

For a realistic image denoising problem, we cannot measure the true non-local image graph as computing it would require the availability of the clean image data. The question of how to estimate the non-local image graph is a key concern for our method, as well as for many other works using non-local methods, as the effectiveness of the overall non-local image denoising procedure is highly dependent on the quality of the estimate underlying non-local graph. This presents a chicken-and-egg problem, as denoising the image demands a good quality non-local graph, but computing the graph depends on good estimates of the clean image patches.

We have found that directly applying the denoising procedures described in this section using the non-local graph computed directly from noisy image patches leads to very poor denoising performance. This necessitates the development of some method for robustly estimating the non-local graph in the presence of noise. While we have not satisfactorily solved this problem, we still wish to demonstrate the potential of the non-local graph wavelet methods for capturing image structure. In this work, we sidestep this issue by using some different image denoising method, dubbed the "predenoiser", to first estimate the image. We then compute the non-local image graph from this "predenoised" image. Once the non-local image graph is computed, the predenoised image is discarded and we then proceed purely with our graph wavelet techniques. In this work, we use the Gaussian Scale Mixture model of Portilla et. al. [23] to perform the predenoising.

While incorporating predenoising yields a highly effective overall denoising algorithm, it introduces some question of how much the performance is depending on the effectiveness of the estimation of the non-local graph. To address this, we also examine denoising under the condition where the non-local graph is given by an "oracle", *i.e.*, computed from the original clean image. While this does not yield a true denoising methodology, as it requires access to the original clean image, it does provide an upper bound on the performance of the overall method.

1.6.2 Scaled Laplacian Thresholding

Our first denoising algorithm will consist of taking the forward graph wavelet transform, applying a spatially varying soft thresholding operation to the wavelet coefficients, and then inverting the transform. We derive the soft thresholding rule as a Bayesian MAP estimator, assuming the signal coefficients follow the scaled Laplacian model described in 1.5.

Given an input noisy image, we must first estimate the parameters α_β at each wavelet band β. Below we describe a simple estimator for α_β given a set of noisy coefficients $d_{\beta,n}$ arising from an image corrupted with additive Gaussian noise of known variance σ_I.

In the wavelet domain, the noise model is $d_{\beta,n} = c_{\beta,n} + e_{\beta,n}$, where the noise coefficient $e_{\beta,n} = \langle \psi_{\beta,n}, \mathbf{n} \rangle$. Assuming signal and noise are independent we have

$$\mathbb{E}[d_{\beta,n}^2] \;=\; \mathbb{E}[c_{\beta,n}^2] \;+\; \mathbb{E}[e_{\beta,n}^2] \;=\; 2\alpha_\beta^2\sigma_{\beta,n}^2 \;+\; \sigma_I^2\sigma_{\beta,n}^2.$$

Summing over n and rearranging gives

$$\alpha_\beta^2 = \frac{1}{2\sum_n \sigma_{\beta,n}^2}\left(\sum_n \mathbb{E}[d_{\beta,n}^2] - \sigma_I^2 \sum_n \sigma_{\beta,n}^2\right).$$

Replacing $\sum_n \mathbb{E}[d_{\beta,n}^2]$ by the plug-in estimate $\sum_n d_{\beta,n}^2$ gives a formula for α_β. Note that for actual data it is possible for this method to estimate a negative value for α_β^2, in this case we set $\alpha_\beta = 0$.

The scaled Laplacian thresholding rule is derived by assuming the independence of wavelet coefficients of both the signal and noise at different spatial locations. Under this independence assumption we may consider the posterior probability of the desired coefficient given the observed coefficient,

$$\mathtt{p}(c_{\beta,n} \mid d_{\beta,n}) \;=\; \mathtt{p}(d_{\beta,n} \mid c_{\beta,n})\, \mathtt{p}(c_{\beta,n}) \;/\; \mathtt{p}(d_{\beta,n})$$

separately for each location n and band β. The noise present in $c_{\beta,n}$ is zero mean Gaussian with variance $\sigma_I^2\sigma_{\beta,n}^2$, while the desired signal is zero mean Laplacian with parameter $\alpha_\beta\sigma_{\beta,n}$. These together imply that

$$\mathtt{p}(c_{\beta,n} \mid d_{\beta,n}) \;=\; C \exp\big(-\tfrac{1}{2\sigma_I^2\sigma_{\beta,n}^2}(d_{\beta,n} - c_{\beta,n})^2 - \tfrac{|c_{\beta,n}|}{\alpha_\beta\sigma_{\beta,n}}\big),$$

where $C > 0$ does not depend on $c_{\beta,n}$. Computing the argmax of $\mathtt{p}(c_{\beta,n} \mid d_{\beta,n})$ is given by

$$c_{\beta,n}^{\mathrm{MAP}}(d_{\beta,n}) \;=\; \arg\min_x\, (d_{\beta,n} - x)^2 \;+\; \tfrac{2\sigma_I^2}{\alpha_\beta}\sigma_{\beta,n}|x|. \tag{1.10}$$

Solving this shows $c_{\beta,n}^{\mathrm{MAP}} = S_{\tau_{\beta,n}}(d_{\beta,n})$ with $\tau_{\beta,n} = \frac{\sigma_I^2}{\alpha_\beta}\sigma_{\beta,n}$; where $S_\tau(y) = (|y| - \tau)_+ \operatorname{sign}(y)$ is the well known soft-thresholding operator, with $(\lambda)_+ = \frac{1}{2}(\lambda + |\lambda|)$ for $\lambda \in \mathbb{R}$. Once the estimated coefficients $\mathbf{c}^{\mathrm{MAP}}$ are computed, the denoised image is recovered by applying the inverse wavelet transform, described in Section 1.2.

Application of the scaled Laplacian model requires knowledge of the norms $\sigma_{\beta,n}$ of the wavelet coefficients. This poses a somewhat challenging problem given the structure of how the SGWT is computed, as the wavelets are not explicitly formed in memory. While the wavelet norms may be computed naively

by applying the SGWT to delta impulses, and looping over all image locations, this procedure would be extremely slow. Instead, we estimate the $\sigma_{\beta,n}$ by computing the variances of the wavelet coefficients of pseudorandomly generated white noise. Let $\mathbf{e}^{(k)} = \mathbf{T}\mathbf{n}^{(k)}$, where $\mathbf{n}^{(k)}$ is a sample of zero mean unit white Gaussian noise in the image domain. We estimate $\sigma_{\beta,n}$ by

$$\sigma^*_{\beta,n} = \frac{1}{K}\sum_{k=1}^{K} e^{(k)}_{\beta,n}$$

The numerical results presented in this work used $P = 100$ samples.

1.6.3 Weighted ℓ_1 Minimization

While the scaled Laplacian thresholding may be cleanly motivated by statistical modeling, it is limited in that each wavelet coefficient is processed independently of all the others. As a related alternative denoising algorithm, we develop a variational weighted ℓ_1 minimization procedure which allows for coupling between different wavelet coefficients. The coefficients of the estimated image are recovered as the minimum of a convex functional consisting of the sum of a quadratic error term in the image domain and a weighted ℓ_1 penalty on the wavelet coefficients.

This weighted ℓ_1 functional is obtained from considering the minimization problems from (1.10). Formally, these NB uncoupled minimizations are the solution to

$$\arg\min_{\mathbf{c}} \sum_{\beta,n}(d_{\beta,n} - c_{\beta,n})^2 \; + \sum_{\beta,n} 2\tau_{\beta,n}\,|c_{\beta,n}|.$$

The first term in the above expression is the sum of squares of the portion of the wavelet coefficients assigned to the noise process. We arrive at our ℓ_1 program by replacing this term by the sum of squares of the estimated noise in the image domain, where the recovered image is given by $\mathbf{T}^T\mathbf{c}$. Specifically, this yields

$$\mathbf{c}^* \;=\; \arg\min_{\mathbf{c}} \; \|\mathbf{y} - \mathbf{T}^T\mathbf{c}\|^2 \;+\; \|\mathbf{c}\|_{\tau,1}, \tag{1.11}$$

where $\|\mathbf{c}\|_{\tau,1} = \sum_{\beta,n} 2\tau_{\beta,n}|c_{\beta,n}|$ stands for the τ-weighted ℓ_1-norm of the coefficients $\mathbf{c}$. After solving this minimization problem, the denoised image is given by $\mathbf{x}^* = \mathbf{T}^T\mathbf{c}^*$.

The objective function in (1.11) is convex in $\mathbf{c}$, and thus admits a unique minimum. We compute the minimum using the Forward-Backward (FB) splitting algorithm described by Combettes and Wajs [24]. We note that the resulting procedure is very closely related to Iterative Soft Thresholding (IST) [25, 26, 27]. For completeness, we give a brief description below.

We apply the FB algorithm writing the objective function $f(\mathbf{c})$ in (1.11) as the sum $f = f_1 + f_2$, where both $f_1(\mathbf{c}) \equiv \|\mathbf{c}\|_{\tau,1}$ and $f_2(\mathbf{c}) \equiv \|\mathbf{y} - \mathbf{T}^T\mathbf{c}\|^2$ are convex, and f_2 is differentiable. The FB splitting is a general iterative method for solving convex optimization problems of this type. This algorithm uses the proximal operator [28], which takes a general convex function φ and returns a new function prox_φ defined by $\text{prox}_\varphi z = \arg\min_x \varphi(x) + \frac{1}{2}\|x - z\|^2$. The FB splitting computes a series of iterates which use the component functions f_1 and f_2 only through computation of ∇f_2 and prox_{f_1}. Given the algorithm parameters γ^{fb} and λ^{fb}, the FB splitting computes

$$\mathbf{c}_{n+1} = \mathbf{c}_n + \lambda^{\text{fb}}\big(\text{prox}_{\gamma^{\text{fb}} f_1}\left[\mathbf{c}_n - \gamma^{\text{fb}}\nabla f_2(\mathbf{c}_n)\right] - \mathbf{c}_n\big).$$

It is proven in [24] that for $\lambda^{\text{fb}} \in [0,1]$ the above iterates $\mathbf{c}_n$ converge to the minimum of $f(\mathbf{c})$, subject to some technical conditions relating the Lipschitz norm of f_2 and the maximum allowable value of γ^{fb}.

The essential reason the FB algorithm is applicable to our objective functions is that the computation of $\text{prox}_{\gamma^{\text{fb}} f_1}$ reduces to simple soft thresholding in each component. A straightforward calculation shows that

$$\big(\text{prox}_{\gamma^{\text{fb}} f_1}\,\mathbf{c}\big)_{\beta,n} \;=\; S_{\gamma^{\text{fb}}\tau_{\beta,n}}(c_{\beta,n}).$$

For the examples presented in this work, we used the parameters $\gamma^{\text{fb}} = 0.1$ and $\lambda^{\text{fb}} = 0.8$. We ran the FB algorithm for 20 iterations, which was enough to observe empirical convergence of the objective function.

1.6.4 Denoising results

We have performed a number of numerical experiments in order to demonstrate the efficacy of the graph wavelet methods for both image denoising and deconvolution problems. We also explore the change in performance of our denoising methods when the local orientation component of the hybrid local/non-local is removed. Our test images were all 256×256 pixels in size, with greyscale values ranging between 0 and 255. In all our experiments, reconstructed image qualities have been assessed by the Peak Signal-to-Noise Ratio (PSNR) defined as

$$\text{PSNR}(x, x^*) \;=\; 10\log_{10}\frac{255^2}{N^{-1}\|x - x^*\|^2},$$

where $x \in \mathbb{R}^N$ and $x^* \in \mathbb{R}^N$ stand for the original and the reconstructed image.

The extreme flexibility of the methods used in this work give rise to a number of parameters for detailing the specifics of both the SGWT and the graph construction. For convenience, we list them here. For all of our denoising examples, we have used the SGWT with $J = 20$ scales, with $K_{\text{lp}} = 200$ and polynomial order

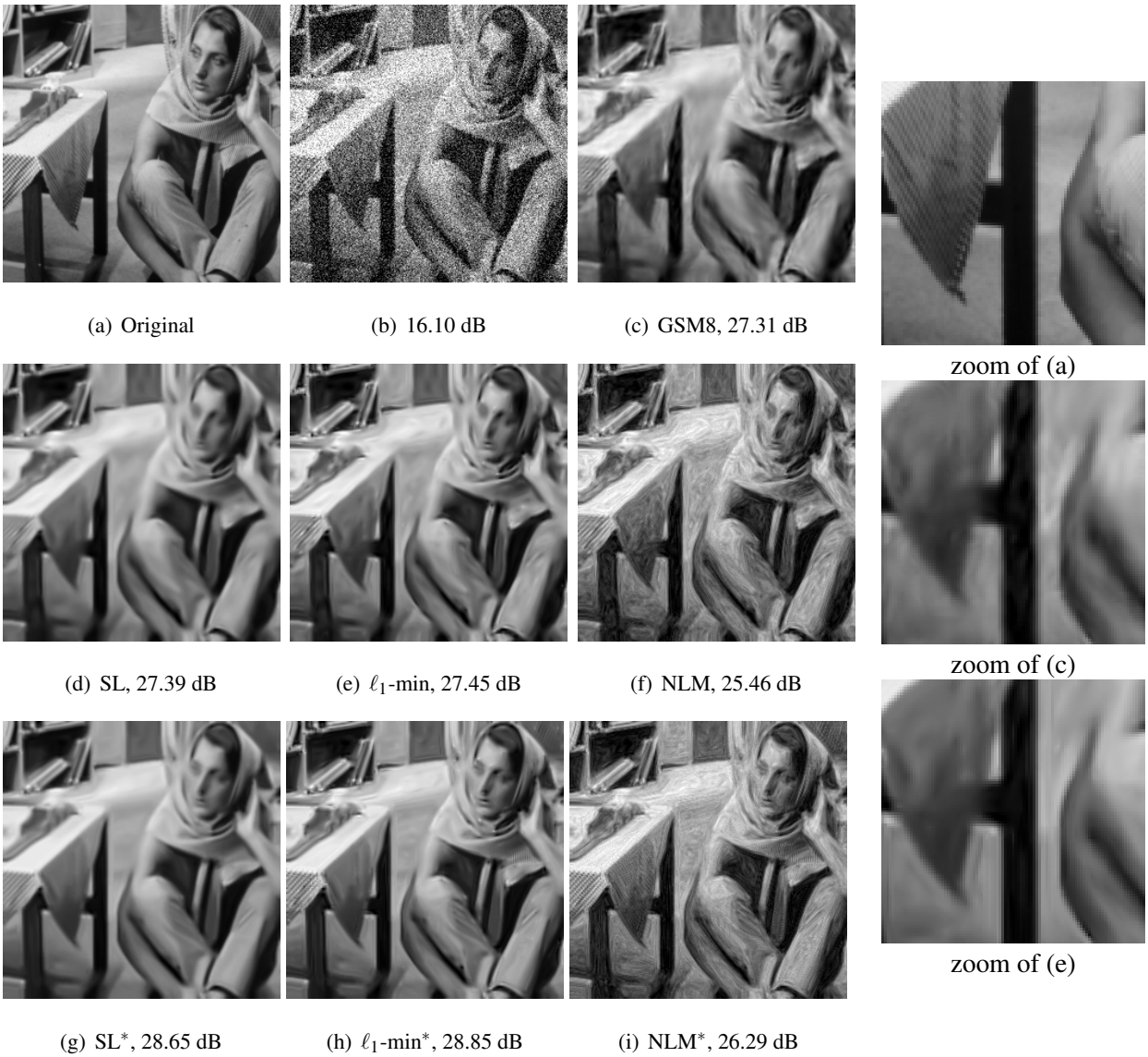
		Noisy	GSM8	SL†	ℓ_1-min†	SL	ℓ_1-min	NLM	SL*	ℓ_1-min*
barbara	$\sigma = 20$	22.12	<u>30.78</u>	29.74	29.48	30.70	30.75	30.07	31.42	**31.47**
	$\sigma = 40$	16.10	27.31	26.38	26.18	27.37	<u>27.46</u>	25.46	28.62	**28.85**
	$\sigma = 80$	10.08	23.89	23.05	23.14	24.08	<u>24.11</u>	20.02	25.93	**26.36**
boat	$\sigma = 20$	22.12	<u>30.11</u>	28.82	28.86	29.54	29.68	29.14	30.27	**30.36**
	$\sigma = 40$	16.10	<u>26.68</u>	25.39	25.65	26.28	26.41	24.90	27.36	**27.82**
	$\sigma = 80$	10.08	<u>23.58</u>	22.63	22.91	23.36	23.24	19.42	24.98	**25.54**
lena	$\sigma = 20$	22.12	<u>31.39</u>	30.22	29.43	31.14	31.13	30.38	**31.80**	31.77
	$\sigma = 40$	16.10	27.69	26.38	26.20	27.69	<u>27.71</u>	25.49	28.74	**28.91**
	$\sigma = 80$	10.08	24.27	23.00	23.23	<u>24.43</u>	24.35	19.93	26.00	**26.29**
house	$\sigma = 20$	22.12	**<u>32.25</u>**	30.71	30.50	32.01	31.97	30.09	31.72	31.99
	$\sigma = 40$	16.10	<u>29.08</u>	27.97	27.57	28.91	28.77	24.97	29.28	**29.53**
	$\sigma = 80$	10.08	<u>25.85</u>	24.73	24.73	25.82	25.62	19.57	26.83	**27.25**

Table 1.3: Performance of proposed denoising algorithm and variants, reported by PSNR for 4 standard 256×256 test images. In above, GSM8 : 8-band GSM method of Portilla et. al., used for predenoising; SL† and ℓ_1-min†: variants of SL and ℓ_1-min where non-local graph computed without predenoising; SL: Scaled Laplacian thresholding; ℓ_1-min : weighted ℓ_1 minimization; NLM : non-local means using same non-local graph as SL and ℓ_1-min ; SL* and ℓ_1-min*: oracle variants. In bold, the highest PSNR per row. Underlined, the best PSNR amongst the non-oracle methods.

$m = 80$ for the fast SGWT transform. The nonlocal image graph was constructed using 11×11 patches (*i.e.*, patch radius $K = 5$), $M = 10$, and $\mu_{NL} = 0.75$. For the hybrid local/non-local graph, we used $S = 2$ orientation directions, with $\delta = 0.5$, and normalized hybridization parameter $\lambda' = 0.15$.

Results for the Scaled Laplacian thresholding (SL) and weighted ℓ_1 minimization (ℓ_1-min) algorithms on four standard test images are shown in Table (1.3). Three different levels of noise were tested, with standard deviation $\sigma = 20$, 40 and 80. The variants using the oracle to compute the nonlocal graph, as well as variants not using predenoising (*i.e.*, computing the nonlocal graph directly from the noisy image) are also shown. We include comparison against the 8-band GSM used for predenoising, as well as the result of non-local means applied using the same non-local graph used in the SL and ℓ_1-min cases.

We first note that the PSNR performance of both the SL and ℓ_1-min methods are competitive with the 8-band GSM, providing higher performance in a few cases, but typically within 0.2 dB. While the similarity in denoising performance may seem to suggest that the graph wavelet methods are simply reproducing the



(a) Original (b) 16.10 dB (c) GSM8, 27.31 dB

(d) SL, 27.39 dB (e) ℓ_1-min, 27.45 dB (f) NLM, 25.46 dB

(g) SL*, 28.65 dB (h) ℓ_1-min*, 28.85 dB (i) NLM*, 26.29 dB

Figure 1.5: Denoising results. Non-local Means* is for Non-local Means with oracle graph.

predenoised image, the results presented really do represent a distinct methodology. This can be seen by examining images of the results, in particular the visual quality of the GSM predenoiser and the ℓ_1-min method are different. In Figure 1.5, carefully comparing images (c) and (e) shows the presence of unsightly ripple artifacts in the GSM image that are smooth in the ℓ_1-min image, such as near the table edges.

Clearly the good performance of the graph wavelet methods does depend on calculating a good estimate of the non-local graph. As shown in Table 1.3, the SL and ℓ_1-min methods calculated using the nonlocal graph computed directly from the noisy image show poorer performance, typically losing over 1dB. In

	$\sigma = 20$	$\sigma = 40$	$\sigma = 80$
ℓ_1-min	0.133	0.117	0.116
oracle ℓ_1-min	-0.164	-0.159	-0.067

Table 1.4: PSNR($\lambda' = 0.15$) - PSNR($\lambda' = 0$), averaged over 16 test images under different noise conditions.

contrast, the oracle methods which use the perfect nonlocal graph show extremely good performance. This is especially noticeable for the high noise ($\sigma = 80$) case, exceeding the GSM method by more that 2 dB.

While the PSNR performance of the SL and ℓ_1-min methods using predenoising are very similar, their visual qualities are different. The SL method results seem oversmoothed compared to the ℓ_1-min results, which preserve sharper edges and generally appear to have more high frequency content. Interestingly, the PSNR of the ℓ_1-min method is consistently better than SL for the oracle case, indicating that the ℓ_1-min method is more sensitive to the quality of the non-local graph.

The non-local graph used in our methods has weights that are computed the same way as the weights used in the non-local means method for image denoising. Both the non-local graph wavelet methods and non-local means can be described using the same underlying non-local graph structure. It is thus interesting to compare against the results of non-local means denoising, using exactly the same non-local graph. Given the adjacency matrix $\mathbf{W}$, the non-local means proceeds by replacing each pixel of the noisy image by the weighted average of its graph neighbors. Recall that $\mathbf{W}$ is defined with zeros on the diagonal; the non-local means denoising result is

$$\mathbf{x}^*_{\text{NLM}}(j) = \frac{\mathbf{y}(j) + \sum_i w_{i,j}\mathbf{y}(i)}{1 + \sum_i w_{i,j}} \tag{1.12}$$

We show the result of this non-local means in figure 1.5, and note that the performance of the graph wavelet methods is much better. It should be noted that the non-local means results shown here are not the best results that can be obtained with the non-local means algorithm. The graph construction here has been optimized to give good results for the graph wavelet methods, we have observed that different choices for the graph sparsification and μ_{NL} can improve the non-local means performance. However, the dramatic difference in performance when using the same graph emphasizes that the graph wavelet methods are really doing something beyond simple non-local means.

In order to assess the usefulness of introducing orientation through the local/non-local hybridization described in Section 1.4, we have studied the change in performance when the local oriented adjacency is removed (*i.e.*, setting $\lambda' = 0$). Results averaged over a larger set of 16 test images are shown in Table 1.4.

Including the oriented local component provides a moderate improvement on average when the non-local graph is computed via predenoising. Interestingly, including the local component actually degrades performance when the oracle graph is used. This suggests that the usefulness of the local component is to provide some regularization of the imperfectly estimated non-local graph. For the oracle case, as the non-local graph is perfectly estimated, the local component is not needed.

1.7 Conclusions

We have introduced a novel image model based on describing the wavelet coefficients under a new non-local image graph wavelet transform. The underlying non-local image graph has vertices associated with the image pixels, and edge weights which measure the similarity of different image patches. We constructed an overcomplete frame of wavelets on this graph using the spectral graph wavelet transform, and showed that the nonlocal graph wavelet coefficients of images are well modeled by a scaled Laplacian probability model. We also detailed a way for building local oriented wavelets with the spectral graph wavelet transform, enabling the construction of hybrid local/non-local graph wavelets.

As a demonstration of the power of the image graph wavelets, we have developed two related methods for image denoising based on the scaled Laplacian model. A straightforward Bayesian MAP inference in the wavelet domain leads to the scaled Laplacian thresholding method. The second weighted ℓ_1 method uses the scaled Laplacian prior as part of a global convex objective function, leading to improved recovery of higher frequency image features.

There are many opportunities for future work using the non-local image graph wavelets. At present, we are actively investigating the use of non-local image graph wavelets for other image processing problems. Both image deconvolution and image super-resolution problems may be cast in a convex variational framework as the ℓ_1-min method. An additional important question is how to improve the estimation of the non-local graph, hopefully to avoid the somewhat awkward use of the predenoising method.

1.8 Acknowledgements

DH is funded by the US DoD TATRC Research Grant # W81XWH-09-2-0114

LJ is funded by the Belgian Science Policy (Return Grant, BELSPO, Belgian Interuniversity Attraction Pole IAP-VI BCRYPT).

Bibliography

[1] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied Computational Harmonic Analysis*, 2010.

[2] M. F. Barnsley, "Fractal functions and interpolation," *Constructive Approximation*, vol. 2, pp. 303–329, 1986.

[3] A. E. Jacquin, "Image coding based on a fractal theory of iterated contractive image transformations," *IEEE Transactions on Image Processing*, vol. 1, no. 1, pp. 18–30, Jan 1992.

[4] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 490–530, 2005.

[5] G. Gilboa and S. Osher, "Nonlocal linear image regularization and supervised segmentation," *Multiscale Modeling & Simulation*, vol. 6, no. 2, pp. 595–630, 2007.

[6] A. Buades, B. Coll, and J.-M. Morel, "Nonlocal image and movie denoising," *International Journal of Computer Vision*, vol. 76, pp. 123–139, 2008.

[7] L. Pizarro, P. Mrzek, S. Didas, S. Grewenig, and J. Weickert, "Generalised nonlocal image smoothing," *International Journal of Computer Vision*, vol. 90, pp. 62–87, 2010.

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

[9] F. Zhang and E. R. Hancock, "Graph spectral image smoothing using the heat kernel," *Pattern Recogn*, vol. 41, no. 11, pp. 3328–3342, Jan 2008.

[10] A. D. Szlam, M. Maggioni, and R. R. Coifman, "Regularization on graphs with function-adapted diffusion processes," *J. Mach. Learn. Res.*, vol. 9, pp. 1711–1739, June 2008.

[11] G. Peyré, "Image processing with nonlocal spectral bases," *Multiscale Modeling & Simulation*, vol. 7, no. 2, pp. 703–730, 2008.

[12] M. Crovella and E. Kolaczyk, "Graph wavelets for spatial traffic analysis," *INFOCOM 2003*, vol. 3, pp. 1848 – 1857 vol.3, Jan 2003.

[13] M. Jansen, G. P. Nason, and B. W. Silverman, "Multiscale methods for data on graphs and irregular multidimensional situations," *Journal of the Royal Statistical Society: Series B*, vol. 71, no. 1, pp. 97–125, 2009.

[14] A. B. Lee, B. Nadler, and L. Wasserman, "Treelets - an adaptive multi-scale basis for sparse unordered data," *Annals of Applied Statistics*, vol. 2, pp. 435–471, 2008.

[15] R. R. Coifman and M. Maggioni, "Diffusion wavelets," *Applied and Computational Harmonic Analysis*, vol. 21, pp. 53–94, Jan 2006.

[16] I. Daubechies, *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1992.

[17] T. Tasdizen, "Principal neighborhood dictionaries for nonlocal means image denoising," *Image Processing, IEEE Transactions on*, vol. 18, no. 12, pp. 2649 –2660, 2009.

[18] T. Brox, O. Kleinschmidt, and D. Cremers, "Efficient nonlocal means for denoising of textural patterns," *Image Processing, IEEE Transactions on*, vol. 17, no. 7, pp. 1083 –1092, 2008, same purpose as Tasdizen paper.

[19] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 1.21," http://cvxr.com/cvx, Oct. 2010.

[20] S. Mallat, "A theory for multiresolution signal decomposition: the Wavelet representation," *IEEE Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, 1989.

[21] P. Moulin and J. Liu, "Analysis of multiresolution image denoising schemes using generalized Gaussian and complexity priors," *IEEE Transactions on Information Theory*, vol. 45, pp. 909–919, 1999.

[22] H. Rabbani, "Image denoising in steerable pyramid domain based on a local Laplace prior," *Pattern Recognition*, vol. 42, no. 9, pp. 2181 – 2193, 2009.

[23] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of Gaussians in the wavelet domain," *IEEE Transactions on Image Processing*, vol. 12, pp. 1338–1351, 2003.

[24] P. Combettes and V. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Model Sim*, vol. 4, no. 4, pp. 1168–1200, Dec 2005.

[25] I. Daubechies, M. Defrise, and C. D. Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Comm. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, 2004.

[26] M. Elad and M. Elad, "Iterative-shrinkage algorithms," in *Sparse and Redundant Representations*. Springer New York, 2010, pp. 111–136.

[27] J. Bioucas-Dias and M. Figueiredo, "A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration," *Image Processing, IEEE Transactions on*, vol. 16, no. 12, pp. 2992 –3004, 2007.

[28] J. Moreau, "Fonctions convexes duales et points proximaux dans un espace hilbertien," *CR Acad. Sci. Paris Ser. A Math*, vol. 255, pp. 2897–2899, 1962.