Chapter 5
Filtering on Graphs

Abstract Measured data often includes noise. A data point measured in isolation offers little opportunity to tease signal apart from noise. However, this separation of noise from the signal becomes more possible when multiple data points are acquired which have a relationship with each other. A spatial relationship, such as the edge set of a graph, permits the use of the collective data acquisition to make better decisions about the true data underlying each measurement. This process whereby the spatial relationships of the data are used to provide better estimates of the noiseless data is called a *filtering* or a *denoising* process. In this chapter, we outline the assumptions used to justify spatial filtering, describe the equivalent of Fourier analysis on a general graph and discuss how different parameter settings of a small number of variational approaches to filtering lead to a large number of commonly used filters. Although our focus in this chapter is on the filtering of node data (0-cochains), we also discuss how these techniques may be applied to the filtering of edge data (i.e., flows, or 1-cochains) and to the filtering of data associated with higher-dimensional cells.

Data filtering is a common procedure in any kind of data processing and analysis application. In this chapter, we assume that our object of interest is a graph with data $\tilde{s}_i$ assigned to each node v_i and a meaningful neighborhood definition given by the edge set. The standard assumption is that noise has been added to all of the data. In the absence of a data model, there is very little that one can do to extract the signal from the noisy observation. However, data associated with a discrete cell complex has meaningful neighborhood relationships and we may generally assume that the noise has a high spatial frequency. Therefore, the goal of most filtering operations is to remove the high frequency noise, while being careful to preserve the high frequency signal (often modeled as spatial discontinuities).

We begin this chapter by addressing the filtering topic in a traditional context in which the general goal is to produce a filter that removes some frequency range in the data (e.g., a lowpass filter). The focus here will be on Fourier-based techniques in the context of data associated with an arbitrary graph. Following this exposition of

L.J. Grady, J.R. Polimeni, *Discrete Calculus*,
DOI 10.1007/978-1-84996-290-2_5, © Springer-Verlag London Limited 2010

Fourier techniques on a graph, we address the lowpass filtering of data as an energy optimization problem. These optimization-based filtering operations will then be modified to preserve rapid data changes (i.e., discontinuities). Specifically, we may model the filtering process as smoothing noise *within* a region (node set), but not *between* regions. We then proceed to show how to filter via gradient manipulation and discuss nonlocal filtering. These filtering techniques are then generalized beyond the processing of node data to filtering procedures that remove noise in edge data (e.g., flows, traffic) or data associated with higher-dimensional cells. Finally, the chapter ends by showing applications of these filtering techniques.

5.1 Fourier and Spectral Filtering on a Graph

The traditional approach to filtering data sampled on an equally spaced grid in arbitrary dimensions is to apply a digital filter intended to suppress certain frequencies without disrupting others. Digital filtering approaches of this nature comprise an enormous literature which we do not intend to review (a standard text on this subject is Oppenheim and Schafer [297]). Our first goal in this section will be to examine *when* we can apply standard Fourier methods to data defined on a graph.

The Fourier transform was originally developed by Fourier to produce solutions to the diffusion (heat) equation

$$\frac{\partial u}{\partial t} = \nabla^2 u, \tag{5.1}$$

where u is a real-valued function defined on $\mathbb{R}^N$ and $\nabla^2 = \nabla \cdot \nabla$ is the diffusion operator. Specifically, in $\mathbb{R}^N$, the standard Fourier basis constitute the *eigenfunctions* of the Laplacian operator. We would therefore expect that the columns of the Discrete Fourier Transform (DFT) matrix would likewise represent the *eigenvectors* of the Laplacian matrix from discrete calculus, appearing in the diffusion equation (see Chap. 2)

$$\frac{\partial \mathbf{x}}{\partial t} = -\mathbf{L}\mathbf{x}. \tag{5.2}$$

In this section, we will employ s and k as index variables rather than the conventional i and j to avoid confusion with $\mathrm{i} = \mathrm{j} = \sqrt{-1}$. The DFT matrix, $\mathbf{Q}$, has rows s and columns k defined as

$$Q_{sk} = \mathrm{e}^{-\frac{2\pi \mathrm{i}}{N}(s-1)(k-1)}, \tag{5.3}$$

where N is the number of nodes in the graph. The first issue we address is to determine when $\mathbf{Q}$ will comprise the eigenvectors of the Laplacian matrix, analogous to the continuous case. We can proceed to show that $\mathbf{Q}$ will form the eigenvectors of the Laplacian matrix, if and only if the Laplacian matrix is *circulant*. Recall that a **circulant matrix** is defined as a matrix in which each row is a circular shift of the

previous row, i.e., $\mathbf{H}(i, j) = \mathbf{H}(i-1, j-1)$, for $i > 1$, $j > 1$, $\mathbf{H}(i, 0) = \mathbf{H}(i-1, N)$. The general form of a circulant matrix can be seen more easily to follow

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 & \dots & h_N \\ h_N & h_1 & h_2 & \dots & h_{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_2 & h_3 & h_4 & \dots & h_1 \end{bmatrix}. \tag{5.4}$$

Note that a single vector, $\mathbf{h}$, is sufficient to generate a circulant matrix if we set the first column of $\mathbf{H}$ to this vector, i.e., $\mathbf{H}(1, j) = \mathbf{h}_j$. An important aspect of circulant matrices is that a circulant matrix embodies a circular convolution operation represented in matrix form.[1] Given this view of circulant matrices, we can now proceed to connect the DFT to a circulant matrix via the convolution theorem.

Before we investigate when a graph structure gives rise to a circulant Laplacian, we must first prove that the DFT vectors are eigenvectors of any circulant matrix.

Theorem 5.1 *The columns of the DFT matrix,* $\mathbf{Q}$*, are eigenvectors of any circulant matrix,* $\mathbf{H}$*.*

This theorem can be proved by first considering a lemma. Define a *shift matrix* as the circulant matrix, $\mathbf{S}$, generated by the row vector $\mathbf{v} = [0, 1, 0, 0, \dots, 0]^\mathsf{T}$. In other words, $\mathbf{S}$ is the identity matrix with each row undergoing a circular shift one entry below the diagonal. While the transformation effected by $\mathbf{S}$ is a delay, its transpose $\mathbf{S}^\mathsf{T}$ represents a shift forward and is a matrix with non-zero entries above the diagonal.

Lemma 5.1 *Vector* $\mathbf{q}_k$*, the kth column of the DFT matrix* $\mathbf{Q}$*, is an eigenvector of* $\mathbf{S}$ *with eigenvalue* $\lambda = \mathrm{e}^{\frac{2\pi \mathrm{i}}{N}(k-1)}$.

Proof The effect of the shift matrix applied to the column vector $\mathbf{q}_k$ will be to produce a new vector, $\tilde{\mathbf{q}}_k = \mathbf{S}\mathbf{q}_k$ such that $\tilde{\mathbf{q}}_k[s] = \mathbf{q}_k[s-1]$ for $s < N$ and $\tilde{\mathbf{q}}_k[N] = \mathbf{q}_k[1]$. However, since the columns of the DFT matrix represent equal partitions of the unit circle [297], this shift can be accomplished by multiplying each entry by $\lambda = \mathrm{e}^{\frac{2\pi \mathrm{i}}{N}(k-1)}$. Therefore,

$$\tilde{\mathbf{q}}_k[s] = \mathbf{q}_k[s-1] = \mathrm{e}^{-\frac{2\pi \mathrm{i}}{N}(s-2)(k-1)} = \mathrm{e}^{-\frac{2\pi \mathrm{i}}{N}(s-1)(k-1)}\, \mathrm{e}^{\frac{2\pi \mathrm{i}}{N}(k-1)} = \mathbf{q}_k[s]\, \mathrm{e}^{\frac{2\pi \mathrm{i}}{N}(k-1)},$$

giving the lemma. □

Now we are prepared to give the proof for Theorem 5.1.

[1] Note that, while circulant matrices represent *circular* convolution, **Toeplitz matrices**, which comprise a distinct class of matrices, represent *linear* convolution. A thorough treatment of these matrices is available in [172].

Proof Let us restrict our attention to proving that the kth column of the DFT matrix $\mathbf{Q}$, $\mathbf{q}_k$, is an eigenvector of any circulant matrix $\mathbf{H}$. Note that row s of $\mathbf{H}$ is generated by the first row of $\mathbf{H}$, $\mathbf{g}$ (which is a flipped and rotated version of $\mathbf{h}$), multiplied $s-1$ consecutive times by the shift matrix, which can be represented as $\mathbf{S}^{s-1}$, i.e., $\mathbf{H}_s = \mathbf{S}^{s-1}\mathbf{g}$. Furthermore, assign the inner product of the generator $\mathbf{g}$ of $\mathbf{H}$ with the kth column of the DFT matrix to the scalar α, i.e., $\mathbf{g}^\mathsf{T}\mathbf{q}_k \equiv \alpha$. Now,

$$\mathbf{H}_s^\mathsf{T}\mathbf{q}_k = (\mathbf{S}^{s-1}\mathbf{g})^\mathsf{T}\mathbf{q}_k = \frac{\alpha}{\lambda^{s-1}} = \alpha\, \mathrm{e}^{-\frac{2\pi \mathrm{i}}{N}(s-1)(k-1)} = \alpha \mathbf{q}_k[s].$$

Therefore, $\mathbf{H}\mathbf{q}_k = \alpha\mathbf{q}_k$. □

Corollary 5.1 *The eigenvalues of* $\mathbf{H}$ *are the Fourier transform of the generating vector* $\mathbf{h}$, *i.e.*, $\lambda = \mathbf{Q}\mathbf{h}$.

Corollary 5.2 (The *Convolution Theorem*) *Since convolution is a finite, linear operation, it may be represented by a matrix. Furthermore, since convolution applies the same kernel to every location, the matrix representing convolution is circulant. Therefore, the convolution of any two signals* $\mathbf{h}$ *and* $\mathbf{x}$ *may be computed by the Fourier transform, i.e.,* $\tilde{\mathbf{x}} = \mathbf{H}\mathbf{x} = \frac{1}{N}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\mathsf{T}\mathbf{x}$, *where the diagonal matrix of eigenvalues* $\mathbf{\Lambda}$ *is equal to* $\mathrm{diag}(\lambda)$.

Note that the factor of $\frac{1}{N}$ is included to satisfy Parseval's (or Plancherel's) Theorem which effectively states that $\frac{1}{\sqrt{N}}\mathbf{Q}$ is a unitary operator, i.e., that $\frac{1}{N}\mathbf{Q}\mathbf{Q}^\mathsf{T} = \mathbf{I}$.

Contrary to the usual interpretation of the Fourier transform as a decomposition into signal frequencies, we view the Fourier transform here as being tightly coupled to the concept of *shift invariance* of the graph.

> Specifically, a graph is called **shift invariant** if there exists a permutation of the node ordering such that the Laplacian matrix representing the graph is circulant. Therefore, any shift-invariant graph has the DFT basis as the eigenvectors of its Laplacian matrix.

We may now examine the graphs that are shift-invariant. One requirement for shift-invariance of a graph is that the graph must be *regular*. Recall that a regular graph is any graph such that every node has the same number of neighbors (degree). Common examples of shift-invariant graphs are infinite lattices, cycles and fully-connected graphs, as depicted in Fig. 5.1. On all of these shift-invariant graphs, we can solve the diffusion equation (5.2) by decomposition of the initial state onto the DFT basis and evolution of each component independently in time. Although regularity is a necessary condition for a graph to be shift-invariant, not all regular graphs are also shift-invariant graphs.

Treatments of the Fourier transform for specific lattices may now be viewed as special cases of shift-invariant lattices. For example, DuBois [116] treats the Fourier

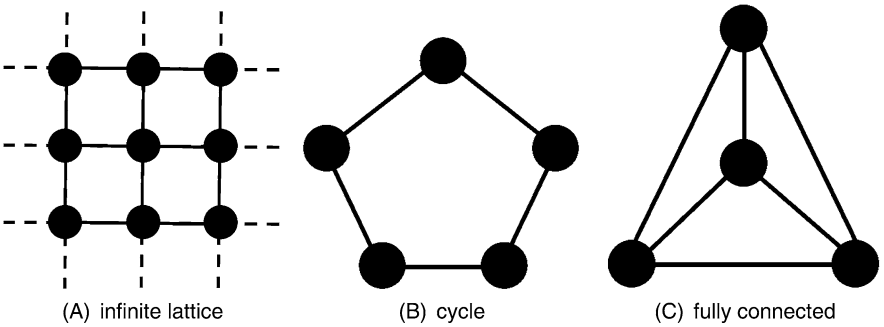

Fig. 5.1 Examples of shift-invariant lattices. A shift-invariant lattice is represented by a *circulant* Laplacian (and adjacency) matrix, which permits the decomposition of signals defined on the nodes of these graphs onto the Fourier basis. (**A**) Infinite (or wrapping) lattice. An infinite lattice is the standard assumption that justifies use of the Fourier transform, even on finite lattices. (**B**) A cycle graph. (**C**) A fully connected graph

transform on a "quincunx" lattice. A second example of a special Fourier transform treatment for a specific lattice is given by the literature on Fourier descriptors, which gives applications of Fourier transforms on cycle graphs [88, 256, 417]. In the signal processing literature, the signal is typically a one-dimensional signal that varies over time. In this context, a shift-invariant system is considered *time-invariant* and the corresponding theory of such signal analysis is called **LTI system theory**, where LTI stands for Linear and Time-Invariant.

Standard signal processing techniques based on the Fourier transform may be applied to functions defined on a shift-invariant graph. A somewhat surprising consequence of this analysis is that Fourier-based filtering *has nothing to do with the graph topology* as long as the graph is shift-invariant, i.e., the signal at every node is treated equally. However, despite this remarkable fact, the evolution of processes closely associated with the Fourier basis, such as the diffusion equation, will depend on the topology of the graph, since the eigenvalues of shift-invariant graphs change with graph topology.

5.1.1 Graphs that Are Not Shift-Invariant

Although the Fourier transform can be used to analyze signals on a wide range of graphs that arise in practice, not all useful graph structures are shift-invariant. Furthermore, a graph that is seemingly regular can lose shift-invariance for two common reasons: (i) The graph has a dynamically changing or adaptive connectivity; (ii) Although the topology is shift-invariant, the weighting is not. The first situation occurs in surface processing (computer graphics) and space-variant vision in which the graph represents a nonuniform domain. The second situation occurs commonly in image processing, in which the lattice has an inhomogeneous weighting to reflect changes in image properties (e.g., object boundaries).

In principle, filtering is also straightforward to implement on shift-variant graphs that do not possess the regularity necessary for Fourier-based filtering. As before, we can simply calculate the eigenvectors, $\mathbf{Q}$, for the Laplacian matrix $\mathbf{L}$ of any graph. The only drawback with a shift-variant graph is that these eigenvectors are not the convenient columns of the DFT matrix (5.3), nor will the eigenvectors be an orthonormal set of complex exponentials. Therefore, it is necessary to calculate the eigenvectors explicitly for each graph, and the familiar Fast Fourier Transform is not available for efficiently projecting a signal into frequency space.

It is straightforward to use the eigenvectors of the Laplacian matrix of a shift-variant graph for filtering in a manner analogous to standard Fourier-based filtering. Specifically, **spectral filtering** could be implemented by:

1. Computing the eigenvectors of the Laplacian matrix, $\mathbf{Q}$;
2. Projecting the data onto the eigenvector space, $\tilde{\mathbf{x}} = \mathbf{Q}\mathbf{x}$;
3. Modifying the frequency components, which are the values of $\tilde{\mathbf{x}}$, to generate $\hat{\mathbf{x}}$ such that the desired filter is achieved; then
4. Backprojecting $\hat{\mathbf{x}}$ via $\mathbf{Q}^{-1}\hat{\mathbf{x}}$ to obtain the filtered signal with the modified frequency components.

Although this procedure is straightforward for applying spectral filtering on an arbitrary graph, it is computationally expensive (and memory intensive) to calculate and apply the full set of eigenvectors of a shift-variant Laplacian matrix. Fortunately, good alternatives exist to explicit calculation of the eigenvectors of a shift-variant graph. The filtering approach adopted by Taubin [371, 372] begins by observing that applying a diffusion equation with a forward Euler method will have the effect of reducing high frequencies. By phrasing the filtering as a difference

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} - \lambda \mathbf{L}\mathbf{x}^{[k]} = \mathbf{x}^{[k]} - \lambda \mathbf{Q}^{\mathsf{T}} \mathbf{\Lambda} \mathbf{Q}\mathbf{x}^{[k]}, \tag{5.5}$$

where $\mathbf{x}^{[k]}$ represents the outcome of the kth iteration, the high frequencies will be subtracted out during the update step, but the low frequencies will largely remain at the same magnitude. The parameter λ controls the speed of the filtering during the iterations, but care must be taken to avoid instability by setting λ small enough to avoid violating the CFL conditions.[2] The iteration represented by (5.5) is commonly called **Laplacian smoothing** in finite element analysis and computer graphics. However, Taubin observed that there was a "shrinking bias" in the context of mesh filtering, as a result of the fact that the low frequencies will eventually also dissipate, leaving only the constant eigenvector (i.e., after enough iterations, all of the nodes would have the same values). A similar effect is observed in image processing in which the diffusion process of (5.5) eventually yields a constant gray value of the image intensities. To counteract this shrinking bias, Taubin preserved the low frequencies while damping the high frequencies by alternating diffusion steps of the form in (5.5) with iterations of the form

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} + \mu \mathbf{L}\mathbf{x}^{[k]}, \tag{5.6}$$

[2]Note that 'λ' is often used to represent an eigenvalue (e.g., Lemma 5.1). We follow Taubin's notation for his λ–μ algorithm by using 'λ' as a parameter when discussing Taubin's algorithm.

where the parameter $\mu > \lambda$, which has the effect of replacing the relatively well-preserved low frequencies that were lost during the smoothing step (5.5) without replacing the high frequencies that were almost completely removed. When $\mu \approx \lambda$, a sharp, "wall filter", is obtained for the low frequencies, while setting $\mu \gg \lambda$ has the effect of a slow filter roll-off.

An immediate concern with Taubin's algorithm as described is that the appropriate values of the λ and μ parameters vary with the graph size, since the magnitude of the eigenvalues of the Laplacian will change. Taubin's resolution of this problem is to employ the symmetric, normalized Laplacian from Chap. 2, $\tilde{\mathbf{L}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$ (where $\mathbf{D}$ is the diagonal matrix of node degrees). This normalized Laplacian is identical to a scaled or reweighted $\mathbf{L}$ for a shift-invariant graph, but has the additional property that the eigenvalues lie in the range $[0, 2]$, which was shown in [81]. Consequently, the values of the parameters λ and μ can remain relatively fixed in order to achieve a particular filtering characteristic with respect to the symmetric normalized Laplacian. Taubin's "λ–μ" filter is very useful in practice due to its simplicity, efficiency and intuitive behavior. However, this algorithm does behave differently from the full spectral filtering approach (i.e., projecting directly onto the eigenvectors of the Laplacian matrix), since Taubin's approach only implicitly employs the spectrum of eigenvalues of the graph. For example, in the case of a shift-invariant graph, the full spectral filtering implementation of a lowpass "wall" filter would be to (i) project the signal onto the eigenvectors (i.e., take a forward DFT), then (ii) set the high frequency components to zero, and finally (iii) project the signal onto the transpose of the eigenvectors (i.e., take an inverse DFT). This procedure would be the same regardless of the distribution of the eigenvalues of the graph, e.g., for any of the shift-invariant graphs such as those in Fig. 5.1. However, the diffusion-based approach would proceed at a much faster rate for, e.g., the fully-connected graph than the ring graph in Fig. 5.1, resulting in a need to adjust the values of λ and μ to achieve with Taubin's λ–μ method the same effect as the full spectral-based method.

Taubin's λ–μ algorithm is defined by the following steps.

1. Choose values for λ and μ parameters.
2. Construct the symmetric normalized Laplacian matrix, $\tilde{\mathbf{L}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$.
3. Given an initial $\mathbf{x}^{[0]}$, solve K iterations, alternating between the diffusion step and the unshrinking step, i.e.,

$$\mathbf{x}^{[2k+1]} = \mathbf{x}^{[2k]} - \lambda\tilde{\mathbf{L}}\mathbf{x}^{[2k]}, \tag{5.7}$$

$$\mathbf{x}^{[2k+2]} = \mathbf{x}^{[2k+1]} + \mu\,\tilde{\mathbf{L}}\mathbf{x}^{[2k+1]}. \tag{5.8}$$

Taubin's method computes only a lowpass filter directly. However, by subtracting the lowpass filtered signal from the original signal, it is possible to generate a highpass filtered signal. By making further combinations of highpass and lowpass filters, Taubin's method may be used to generate bandpass filters of signals on a graph.

5.1.2 The Origins of High Frequency Noise

Now that we have given a precise meaning to "high-frequency" signals on an arbitrary graph, we would like to pause to examine the origins of the common assumption that noise is predominantly contained in high frequencies. The high-frequency character of noise can be justified from two different viewpoints: as an acquisition model and as a model of independent, identically distributed (i.i.d.) noise.

To see the acquisition model viewpoint, consider a linear acquisition matrix, $\mathbf{H}$, a true signal $\mathbf{x}$, an additive noise vector $\boldsymbol{\nu}$, an ideal observed signal $\mathbf{f} = \mathbf{H}\mathbf{x}$ and a noisy observed signal $\mathbf{y}$, such that

$$\mathbf{y} = \mathbf{H}(\mathbf{x} + \boldsymbol{\nu}) = \mathbf{f} + \mathbf{H}\boldsymbol{\nu}. \tag{5.9}$$

Phrased in this way, our goal in filtering is to recover either the ideal observed signal $\mathbf{f}$ or the true signal $\mathbf{x}$. However, for either goal, it is helpful to remove high-frequency noise that corrupts the observation.

We know from the last section that if our graph is shift-invariant and the acquisition matrix $\mathbf{H}$ is shift-invariant, then we can analyze the effects of $\mathbf{H}$ via the DFT and, in particular, the eigenvalues of the acquisition matrix are comprised of the DFT of any single row of $\mathbf{H}$. In general, if our graph is shift-invariant then we can consider the acquisition matrix to be shift-invariant since this shift-invariance simply means that the signal acquisition device operates the same everywhere. A typical acquisition model is that each node performs a weighted sum of the underlying data in a small region about the point. However, since the DFT of a small summation kernel such as a box kernel or a Gaussian kernel concentrates more power in the low frequency range (where a greater extent of the spatial window increasingly concentrates power in the low frequencies), then the high-frequency content of the true signal $\mathbf{x}$ will be suppressed and we may assume that residual high-frequency content in the observation $\mathbf{y}$ is contributed by the noise vector $\boldsymbol{\nu}$. Similarly, if we try to recover the true signal $\mathbf{x}$ via inversion of the acquisition matrix $\mathbf{H}$ (i.e., deconvolution), then the high-frequencies of the observation $\mathbf{f} + \mathbf{H}\boldsymbol{\nu}$ will be amplified and it would be better to filter out the high frequencies in the observed data before inversion.

The other viewpoint for motivating the removal of high frequencies in the observed data is that if the noise term $\boldsymbol{\nu}$ is i.i.d. across the samples then the expected value for the difference of the values of $\boldsymbol{\nu}$ over an edge will be zero. Consequently, a reasonable model is to assume that large gradients are noise and attempt to remove them.

Despite the reasonable assumptions underpinning the motivation for removal of high frequency noise, most real data do contain some high frequency content that we do *not* want removed. A common procedure for keeping the high frequency content in the signal is to assume that the high frequency content generally takes the form of discontinuities in which the data jumps from one smoothly varying region to another. In the context of image processing, this assumption leads to an image model where individual objects have a smoothly varying intensity but that neighboring objects may have an arbitrarily large jump in intensity. Of course, this assumption

is not true for all kinds of data (e.g., a textured object within an image), but it works well for many kinds of real data and refocuses the filtering operation on the detection of discontinuity locations, while applying lowpass filtering everywhere else.

Despite the usefulness of the above lowpass model for data filtering, it is important to recognize that some noise contains significant power in the low frequencies, which is equivalent to the presence of spatial correlations (called "pink noise"). Additionally, filtering can also be applied to problems in which the purpose of the filtering operation is to correct observation or acquisition *error*, which may have any frequency content. Finally, not all data domains are shift-invariant. However, for a shift-variant data domain, the above assumptions may also imply a high-frequency noise model. Although the signal defined on the shift-variant graph may contain high-frequency noise, the eigenvectors of the graph Laplacian corresponding to high-frequency may behave differently from the standard high-frequency eigenvectors employed in Fourier analysis. However, even in the case of a shift-variant data domain, the techniques in this chapter will allow for the removal of high-frequency components of the observed data, even if we allow discontinuities across some edges.[3]

5.2 Energy Minimization Methods for Filtering

Many filtering methods may be viewed as procedures to minimize an energy. In this section, we review several energy minimization models and show how these models lead to commonly used filtering algorithms. We begin this section by describing the basic energy minimization models before proceeding to describe methods for filtering in the presence of implicit or explicit discontinuities.

5.2.1 The Basic Energy Minimization Model

In the previous section, a traditional view of denoising was taken in which the goal was to remove the high frequencies (noise) while preserving the low frequencies (signal). A different approach to accomplishing the same goal is to view the desired denoised signal, $\mathbf{x}$, as a minimum of the energy[4]

$$\mathcal{E}_{\text{BEM}}[x] = \int x \Delta x \, \mathrm{d}t = \int \|\nabla x\|_2^2 \, \mathrm{d}t, \tag{5.10a}$$

[3] Note that in image processing the term *edge* is used to mean discontinuity (e.g., "edge detection"). However, since the context of this entire book is the analysis/processing of graphs (complexes) and data defined on graphs, we reserve the word *edge* to refer strictly to a 1-cell (i.e., we use *edge* in the sense of graph theory).

[4] The term *energy* is used throughout the book to represent an objective function which is optimized to produce a useful application-specific solution. In this case, the solution represents the filtered (denoised) signal. Although the term *energy* is not generally intended to have a physical relationship to energy, note that the energy described in (5.10b) is actually the power dissipation for an electric circuit (when $\mathbf{x}$ represents the electrical potentials at every node), as given in Chap. 3.

$$\mathcal{E}_{\text{BEM}}[\mathbf{x}] = \mathbf{x}^{\mathsf{T}}\mathbf{L}\mathbf{x} = (\mathbf{A}\mathbf{x})^{\mathsf{T}}(\mathbf{A}\mathbf{x}). \tag{5.10b}$$

This energy model returns a high energy for a signal $\mathbf{x}$ dominated by high-frequency components and returns a low energy for a signal $\mathbf{x}$ dominated by low-frequency components. Therefore, finding a signal that minimizes this energy will produce a low-frequency signal. However, by connecting the frequency components of $\mathbf{x}$ with the norm of the gradient, we may generalize this model to measure the gradients with any p-norm.

$$\mathcal{E}_{\text{BEM}}[x] = \int \|\nabla x\|_p^p \, \mathrm{d}t, \tag{5.11a}$$

$$\mathcal{E}_{\text{BEM}}[\mathbf{x}] = \mathbf{1}^{\mathsf{T}}|\mathbf{A}\mathbf{x}|^p = \sum_{e_{ij}\in\mathcal{E}} |x_i - x_j|^p, \tag{5.11b}$$

where the parameter p controls the norm of the energy functional and the summation in (5.11b) is over every edge in the graph. Note that the conditions for a norm are violated when $p < 1$ (specifically the triangle inequality), therefore we will employ the term *p-norm* to refer to

$$\|\mathbf{x}\|_p = \left(\sum_i |\mathbf{x}_i|^p\right)^{\frac{1}{p}}, \tag{5.12}$$

even though we allow $0 < p < 1$. We refer to this energy as the **Basic Energy Model**. The Basic Energy Model pervades this entire book. In future chapters, we see that many common algorithms can be viewed as instances of the Basic Energy Model with different values of p and different interpretations of the variable $\mathbf{x}$. In this chapter, the Basic Energy Model leads to mean, median, mode and minimax filters as well as Laplacian smoothing and anisotropic diffusion. In Chap. 6, the Basic Energy Model leads to several clustering methods in the literature, including max-flow/min-cut, random walks, geodesic clustering, watersheds, spectral clustering, normalized cuts and the isoperimetric partitioning algorithm. Going further, in Chap. 7 we see how the same Basic Energy Model leads to the Laplacian Eigenmaps manifold learning technique. We believe that the unification of so many content extraction (data processing) methods into one functional is a major contribution of this work.

The trivial minimum of the Basic Energy Model given by a constant $\mathbf{x}$ is a useless filtering of a noisy signal. However, a noisy signal may have its energy reduced iteratively without reducing the energy to the undesirable global minimum. There are two standard methods for iteratively minimizing the Basic Energy Model—iterative minimization of the model for each node individually and gradient descent.

5.2.1.1 Iterative Minimization

We begin by describing how to iteratively minimize the Basic Energy Model for each node. If we were given a solution, $\mathbf{x}^{[k]}$, at iteration k, we can fix the solution

everywhere except for one node v_i and consider finding the update which would minimize $\mathcal{E}_{\text{BEM}}[x_i]$ for that node. By definition of (5.11b), the minimum is given by

$$x_i = \text{argmin}\left\{ \sum_{e_{ij} \in \mathcal{E}} |x_i - x_j|^p \right\}. \tag{5.13}$$

When $p = 2$, x_i is assigned to have the *mean* value of its neighbors, while $p = 1$ causes x_i to be assigned the *median* value of its neighbors. Likewise, $p = 0$ assigns x_i to be the *mode* value of its neighbors, and as $p \to \infty$, x_i approaches the *minimax* value of its neighbors. Therefore, if the initial estimate $\mathbf{x}^{[0]} = \mathbf{s}$, *the mean, median, mode and minimax filters can all be viewed as operations designed to incrementally minimize* (5.11b) under different p-norms. Note that many implementations of these filters also include a self-connected edge at each node such that the node's value is included in the mode, mean, median or minimax calculation.

Minimization of the energy functional given in the Basic Energy Model of (5.11b) for p-norms given by $p = 0$, $p = 1$, $p = 2$, and $p \to \infty$, results in the commonplace mode, median, mean, and minimax filters, respectively. Consequently, these filters can all be generalized to arbitrary graphs with any neighborhood structure.

Instead of minimizing the Basic Energy Model by optimizing the solution for each node independently, we can also find the solution via gradient descent (see Appendix B). Since gradient descent is typically used to optimize the Basic Energy Model only for the cases of $p = 2$ or $p = 0$, our discussion will be limited to these cases. When $p = 2$, the Basic Energy Model takes the form

$$\mathcal{E}_{\text{BEM}}[\mathbf{x}] = \mathbf{1}^{\mathsf{T}} |\mathbf{A}\mathbf{x}|^2 = \mathbf{x}^{\mathsf{T}} \mathbf{A}^{\mathsf{T}} \mathbf{A} \mathbf{x} = \mathbf{x}^{\mathsf{T}} \mathbf{L} \mathbf{x}, \tag{5.14}$$

with gradient

$$\frac{\partial \mathcal{E}_{\text{BEM}}[\mathbf{x}]}{\partial \mathbf{x}} = \mathbf{L}\mathbf{x}. \tag{5.15}$$

Therefore, the iterative update to perform gradient descent of an initial noisy signal according to the Basic Energy Model when $p = 2$ would be

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} - \lambda \mathbf{L} \mathbf{x}^{[k]}. \tag{5.16}$$

We saw this filtering algorithm before in (5.5) when it was called Laplacian smoothing. In this context, the *time derivative* of $\mathbf{x}$ was set equal to the negative gradient of the Basic Energy Model, i.e., $\partial \mathbf{x}/\partial t = -\partial \mathcal{E}/\partial \mathbf{x}$. By establishing this equality, a forward Euler solution of the diffusion equation

$$\frac{\partial \mathbf{x}}{\partial t} = -\mathbf{L}\mathbf{x}, \tag{5.17}$$

performs gradient descent on the Basic Energy Model where the time step of the forward Euler operation is represented by λ.

The classic nonlinear anisotropic diffusion model formulated for image processing by Perona and Malik [306] may be viewed as a descent algorithm that minimizes the Basic Energy Model (5.11b) when $p = 0$. In this section, we follow Black et al. [38] to describe this relationship.

Consider the Basic Energy Model for $p = 0$, i.e.,

$$\mathcal{E}_{\text{BEM}}[\mathbf{x}] = \mathbf{1}^{\mathsf{T}}|\mathbf{A}\mathbf{x}|^0 = \sum_{e_{ij}} |x_i - x_j|^0. \tag{5.18}$$

Recall that the value of the ℓ_0-norm $|\cdot|^0$ is zero when the argument is zero and is unity otherwise. Although Black et al. [38] showed that the nonlinear anisotropic diffusion work of Perona and Malik unintentionally approached this energy indirectly, similar filtering models were directly considered at about the same time (see [50, 144, 146]). These models were motivated by the desire to represent implicit discontinuities in the reconstruction. Unfortunately, the $p = 0$ energy in (5.18) is both non-convex and non-differentiable. The non-differentiable aspect of the energy may be overcome by replacing $|\cdot|^0$ with an *M-estimator*, which is an error function used in statistics for which the penalty of large errors increases very slowly for higher error values. M-estimators were used to derive several different methods for generating weights in Chap. 4. For purposes of exposition here, we may approximate the ℓ_0-norm $|\cdot|^0$ via the Welsch function from Chap. 4

$$\rho_\alpha(z) \propto 1 - \exp\left(-\frac{z^2}{\alpha}\right), \tag{5.19}$$

where α is a parameter that controls the approximation to $|\cdot|^0$, such that the approximation fidelity improves as $\alpha \to \infty$. Figure 4.3 in Chap. 4 gives an example of the Welsch function.

Using the Welsch function as an approximation to $|\cdot|^0$ gives us an approximation for the Basic Energy Model energy for the case $p = 0$ as

$$\mathcal{E}_{\text{BEM}}[\mathbf{x}] = \mathbf{1}^{\mathsf{T}}\rho(\mathbf{A}\mathbf{x}), \tag{5.20}$$

where the error function $\rho(\cdot)$ is assumed to operate on the individual components of any vector or matrix input. We may take an initial solution for $\mathbf{x}$, $\mathbf{x}^{[0]} = \mathbf{g}$ and apply gradient descent to reduce the energy (see Appendix B). The gradient of (5.20) is given by

$$\frac{\partial \mathcal{E}_{\text{BEM}}[\mathbf{x}]}{\partial \mathbf{x}} = 2\alpha \mathbf{A}^{\mathsf{T}} \operatorname{diag}(\bar{\rho}(\mathbf{A}\mathbf{x}))\mathbf{A}\mathbf{x}, \tag{5.21}$$

where

$$\bar{\rho}(z) = \exp\left(-\frac{z^2}{\alpha}\right). \tag{5.22}$$

Applying the gradient descent operation to iteratively update $\mathbf{x}$ gives the iteration

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} - \lambda \mathbf{A}^{\mathsf{T}} \operatorname{diag}(\bar{\rho}(\mathbf{A}\mathbf{x}^{[k]}))\mathbf{A}\mathbf{x}^{[k]} = \mathbf{x}^{[k]} - \lambda \mathbf{L}^{[k]}\mathbf{x}^{[k]}, \tag{5.23}$$

where the Laplacian operator $\mathbf{L}^{[k]}$ is updated at each iteration to reflect the changing edge weights given by $\mathbf{G}^{-1} = \operatorname{diag}(\bar{\rho}(\mathbf{A}\mathbf{x}^{[k]}))$. This update rule is exactly the iteration given by Perona and Malik to describe anisotropic diffusion, in which the weights were given by $\bar{\rho}(z)$. In this way, the nonlinear anisotropic diffusion of Perona and Malik is naturally interpreted as an *approximation* to the unweighted Basic Energy Model when $p = 0$, while gradient descent on the Basic Energy Model for $p = 2$ corresponds to a *linear* diffusion smoothing process.

5.2.2 Extended Basic Energy Model

The Basic Energy Model defined in (5.11b) has the distinct problem that the energy is trivially minimized by a trivial solution in which $x_i = \text{constant}$. Therefore, we may extend the Basic Energy Model by viewing the Basic Energy Model as a *smoothness term* to which we may add a *data attachment term* that causes the original data to push back against the smoothness, phrasing the solution as a balance between the model and the measured data. This allows for a non-trivial solution and provides a mechanism for incorporating prior information into the solution in the form of regularization. The generalized formulation may be given as

$$\mathcal{E}_{\text{EBEM}}[x] = \int \|\nabla x\|_p^p \, \mathrm{d}t + \lambda \int |x - s|^p \, \mathrm{d}t, \tag{5.24a}$$

$$\begin{aligned}\mathcal{E}_{\text{EBEM}}[\mathbf{x}] &= \mathbf{1}^{\mathsf{T}}|\mathbf{A}\mathbf{x}|^p + \lambda \mathbf{1}^{\mathsf{T}}|\mathbf{x} - \mathbf{s}|^p \\ &= \sum_{e_{ij}} |x_i - x_j|^p + \lambda \sum_{v_i} |x_i - s_i|^p,\end{aligned} \tag{5.24b}$$

where $\mathbf{s}$ represents the observed (noisy) data and the regularization parameter λ acts to trade off between fidelity of the denoised signal to the original data and smoothness of the denoised signal. Although this new energy model appears substantially different from the Basic Energy Model, we may view the new energy given in (5.24b) as a variant of the Basic Energy Model in which every node $v_i \in \mathcal{V}$ with value x_i has a "phantom" neighbor $u_i \in \mathcal{P}$ with value s_i such that the phantom node u_i is connected only to v_i by an edge weighted with value λ. These phantom nodes are sometimes called **dongle nodes** or **t-links** ("terminal" links in [56]). Therefore, this new energy model may be viewed as an example of the Basic Energy Model with additional nodes which are fixed to Dirichlet boundary conditions (see Chap. 2 for definition and Appendix B for optimization). Since the new energy model may be viewed as equivalent to the Basic Energy Model with an extended node set (where the new nodes are fixed as boundary conditions), we refer to the energy described in (5.24b) as the ***extended* Basic Energy Model**.

The primary advantages of the Extended Basic Energy Model are that the energy is minimized by a nontrivial solution, and that the adjustable influence of the regularization term allows explicit control over the contribution of the filtering. In the case of $p = 2$, an intuitive interpretation exists for the Extended Basic Energy Model. We previously saw that when $p = 2$, gradient descent on the Basic Energy Model could be interpreted as a diffusion process on the noisy measured signal which is solved with a forward Euler method. However, if we solve the diffusion process with a *backward* Euler method, then the diffusion time is $1/\lambda$ [104]. To see this connection, consider again the linear diffusion equation

$$\frac{\partial \mathbf{x}}{\partial t} = -\mathbf{L}\mathbf{x}, \tag{5.25}$$

solved using a backward Euler method [312]

$$\frac{\mathbf{x}^{[k+1]} - \mathbf{x}^{[k]}}{\Delta t} = -\mathbf{L}\mathbf{x}^{[k+1]}, \tag{5.26}$$

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} - \Delta t\, \mathbf{L}\mathbf{x}^{[k+1]}, \tag{5.27}$$

$$(\Delta t\, \mathbf{L} + \mathbf{I})\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]}, \tag{5.28}$$

$$\left(\mathbf{L} + \frac{1}{\Delta t}\mathbf{I}\right)\mathbf{x}^{[k+1]} = \frac{1}{\Delta t}\mathbf{x}^{[k]}. \tag{5.29}$$

This backward Euler method is equivalent to a solution of the Extended Basic Energy Model when $p = 2$, since the minimum of $\mathcal{E}_{\text{EBEM}}[\mathbf{x}]$ is given by

$$(\mathbf{L} + \lambda\mathbf{I})\mathbf{x} = \lambda\mathbf{s}. \tag{5.30}$$

Consequently, the solution for the backward Euler solution for the diffusion problem in (5.29) is equivalent to (5.30) when $\mathbf{s} = \mathbf{x}^{[0]}$ and $\lambda = 1/\Delta t$. Using this connection allows us to intuitively view the λ parameter in the Extended Basic Energy Model as the reciprocal number of iterations employed for solving the Basic Energy Model.

The form of the Extended Basic Energy Model, which consists of a smoothness term and a data term, is also adopted by several other energy minimization filtering models. In the next section we consider another of these models which makes a modification to the smoothness term.

5.2.3 The Total Variation Model

In the domain of image processing, the underlying graph is a lattice embedded into the Euclidean plane. When $p = 2$, the solution to the Basic Energy Model behaves as if the lattice were a representation of the continuous plane. However, this behavior is not observed when $p \neq 2$. Therefore, when $p \neq 2$, the filtering of images produces undesirable gridding artifacts (sometimes called "metrication artifacts").

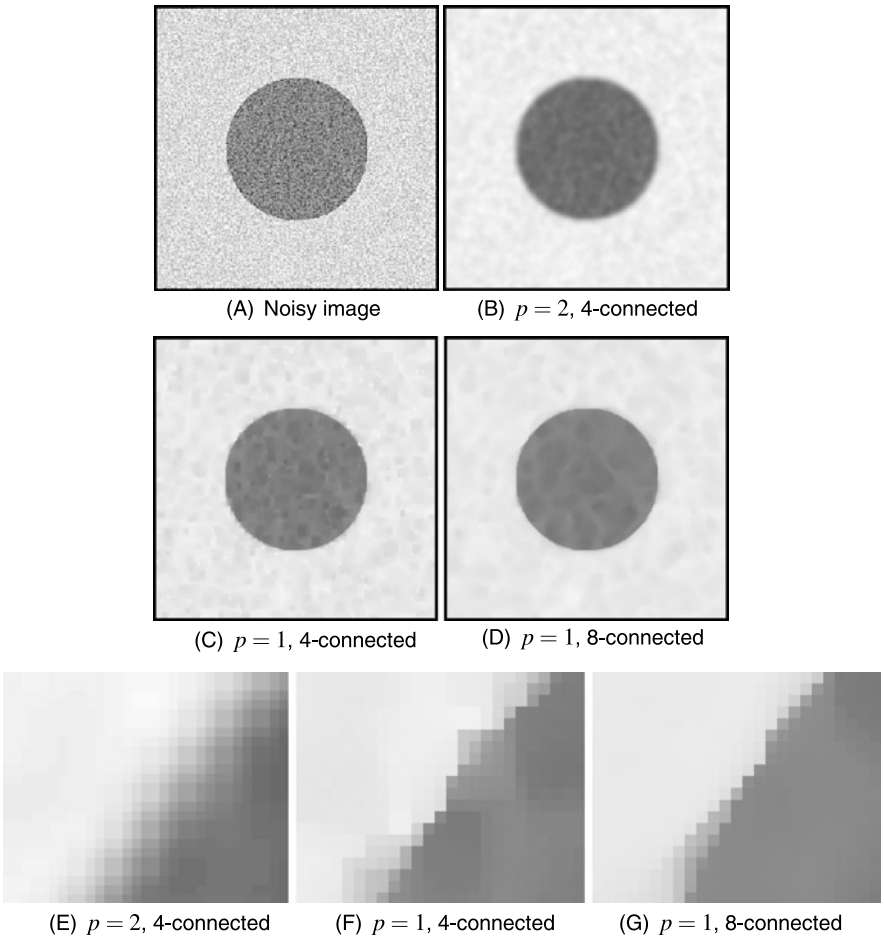

Fig. 5.2 Gridding artifacts observed in the solution to the Basic Energy Model when $p = 1$ but not $p = 2$. Note that the introduction of more edges (an 8-connected lattice) substantially reduces these gridding artifacts. The *bottom row* shows closeup views of the circle boundary to illustrate the variation in gridding artifacts

These gridding artifacts may be reduced by extending the number of edges in the lattice [54]. However, by modifying the smoothness term of the Basic Energy Model such that the smoothness is measured by quantities at *nodes* rather than quantities on *edges*, it was shown that these gridding artifacts may be reduced [386]. This is the approach of the **Total Variation Model**. Figure 5.2 shows the gridding artifacts produced by filtering with the Basic Energy Model when $p = 1$ (a median filter) for a 4-connected and 8-connected lattice. The 8-connected lattice provides substantially reduced gridding artifacts at the cost of additional edges. Although the solution to the $p = 2$ model does not exhibit gridding artifacts, the boundary localization is blurred more for the $p = 2$ solution than the $p = 1$ solution. Preserving

both boundary localization and a reduction of gridding artifacts is obtained by the Total Variation Model.

The Total Variation Model based on nodal smoothness is described by the energy functional

$$\mathcal{E}_{\mathrm{TV}}[x] = \int \|\nabla x\|_2^p \,\mathrm{d}t + \lambda \int |x - s|^p \,\mathrm{d}t, \tag{5.31a}$$

$$\begin{aligned}\mathcal{E}_{\mathrm{TV}}[\mathbf{x}] &= \mathbf{1}^{\mathsf{T}}(|\mathbf{A}^{\mathsf{T}}||\mathbf{A}\mathbf{x}|^2)^{\frac{p}{2}} + \lambda \mathbf{1}^{\mathsf{T}}|\mathbf{x} - \mathbf{s}|^p \\ &= \sum_{v_i}\left(\sum_{j \forall e_{ij}} |x_i - x_j|^2\right)^{\frac{p}{2}} + \lambda \sum_{v_i} |x_i - s_i|^p. \end{aligned} \tag{5.31b}$$

Within the image processing literature, this model is usually credited to [326] and is sometimes referred to as the "Rudin, Osher, Fatemi" model. In contrast to the formulation of the Extended Basic Energy Model given in (5.24b), the minimization of the Total Variation Model searches for the minimum vector p-norm measured at each node. Generally, the Total Variation Model given in (5.31b) is more difficult to solve than the Extended Basic Energy Model since the 1-cochains produced by the gradient operator are then transfered back to 0-cochains. Despite this enhanced difficulty, the total variation model (5.31b) is convex when $p \geq 1$ and is thus solvable by any descent algorithm [123]. When $p = 2$, the Extended Basic Energy Model and the Total Variation Model are equivalent, and when $p = 1$ this model is a *total variation* minimization from which this model takes its name. Filtering via total variation minimization has been well studied (see [69, 326]) and fast algorithms are known for this case [68, 98]. The total variation model was formulated initially in the continuum and on a graph much later [300] and then further extended to weighted graphs and arbitrary p-norms [49, 123].

A problem with all of these energy minimization algorithms is the assumption that *all* high frequencies represent noise. Solutions of these models with lower values of p typically preserve high-frequency content. We may therefore alter our filtering approach to preserve high-frequency signal content by modeling signals as consisting of smooth regions which are separated by a small number of sharp discontinuities. These discontinuities are often thought of as region boundaries. The discontinuities may be incorporated into the energy minimization models by *weighting* changes in the filtered signal differently. Spatial gradients in the filtered signal which occur over discontinuities are penalized less than spatial gradients inside regions. Since these weights are used only as approximations to the discontinuity locations, we call this approach *filtering with implicit discontinuities*.

5.3 Filtering with Implicit Discontinuities

The assumption that low frequency content represents signal while high frequency content represents noise is often not valid in real data, particularly in image data.

Specifically, it is often more accurate to assume that we want to smooth more over similar data at neighboring nodes while smoothing less over dissimilar data at neighboring nodes. The model considered in this section stops short of looking for explicit object boundaries to avoid smoothing over (covered next in Sect. 5.4), but rather performs smoothing on a *weighted* graph (using affinity weights). Generally, the edge weights are derived from some signal feature that can be used to roughly detect object boundaries (e.g., image intensity, image color, node coordinates, texture coefficients). See Chap. 4 for different options to set weights. For the rest of this section, assume that we have a set of nonnegative, normalized, real-valued weights, w_{ij}, that give some indication of discontinuity locations (i.e., $w_{ij} \to 0$ for edges bridging discontinuities and $w_{ij} \to 1$ for edges bridging nodes likely to take the same filtered value).

There are three general approaches to the weighted filtering case: (i) *Spectral filtering* methods compute the eigenvectors of the weighted Laplacian, project the signal, dampen the high frequencies, and reconstruct. Practically, it is preferable to use Taubin's λ–μ algorithm on the weighted Laplacian (as described in Sect. 5.1). (ii) *Edge-based filtering* methods find the solution that tries to fit the data while penalizing smoothness measured across edges. (iii) *Node-based filtering* finds the solution that tries to fit the data while penalizing smoothness measured at nodes.

The Basic Energy Model and the Extended Basic Energy Model are modified easily to incorporate weights, i.e.,

$$\mathcal{E}_{\text{BEM}}[\mathbf{x}] = \sum_{e_{ij}} w_{ij}^p |x_i - x_j|^p, \tag{5.32}$$

$$\mathcal{E}_{\text{EBEM}}[\mathbf{x}] = \sum_{e_{ij}} w_{ij}^p |x_i - x_j|^p + \lambda \sum_{v_i} |x_i - s_i|^p. \tag{5.33}$$

The Basic Energy Model (5.33) now corresponds to minimization via a *weighted* mode, median, mean or minimax filter. The Extended Basic Energy Model (5.33) may be minimized for $p \geq 1$ using any of the convex optimization techniques discussed in the previous section.

Two options are available for introducing weights into the total variation model described in (5.31b). As before, we can use edge weights to modify the scalar-valued *components* that comprise the gradient vector field across edges, or we may use node weights to modify the *norms* of the gradient vector field defined at the nodes. Component weighting was described in [49, 123, 423] and may be formulated as

$$\mathcal{E}_{\text{TV}}[\mathbf{x}] = \sum_{v_i} \left(\sum_{j \,\forall e_{ij}} w_{ij} |x_i - x_j|^2 \right)^{\frac{p}{2}} + \lambda \sum_{v_i} |x_i - s_i|^p. \tag{5.34a}$$

As discussed in [123], the introduction of nonnegative weights still preserves the convexity of the Total Variation Model for $p \geq 1$, and therefore any descent algorithm may be used to perform the optimization.

The scalar-valued vector norms may be weighted similarly to the edge weights based on the estimation of whether or not a node is a boundary node. Chapter 4

provides examples of node weighting and we will proceed by assuming that we have a set of nonnegative, normalized, real-valued weights, w_i, that give some indication of discontinuity locations (i.e., $w_i \to 0$ for nodes on the border of discontinuities and $w_i \to 1$ for "internal" nodes). Given these node weights, we may consider a node-weighted formulation of (5.31b) as

$$\mathcal{E}_{\mathrm{TV}}[\mathbf{x}] = \sum_{v_i} w_i \left(\sum_{j \,\forall e_{ij}} |x_i - x_j|^2 \right)^{\frac{p}{2}} + \lambda \sum_{v_i} |x_i - s_i|^p. \tag{5.34b}$$

Once again, the introduction of node weights preserves the convexity of (5.34b) when $p \geq 1$ and therefore the optimization may be performed using any descent algorithm. Due to the historical derivation of this filtering model from continuum mechanics, the previous node-weighted formulation of the Total Variation Model described in (5.34b) has been more common in the literature (e.g., [11, 386]). However, since discontinuities in 0-cochains are defined *between* nodes, we generally recommend the edge-weighting formulation described by (5.34a). The node-based weighting loses both *directional* discontinuity information and *precision* of the boundary location, since boundaries lie *between* groups of nodes.

We have considered the application of the various filtering approaches on a weighted graph in which the weights were used to encode knowledge of discontinuities. However, these discontinuities were viewed as real-valued weights and were not constrained to form any sort of closed *boundary* surrounding different data clusters. In the next section, we will remove the weights and introduce an explicit boundary variable that explicitly encodes the discontinuities over which smoothing is not permitted.

5.4 Filtering with Explicit, but Unknown, Discontinuities

Instead of treating the discontinuities implicitly with weights, we can formulate the discontinuities explicitly as a boundary that separates some nodes from others over which we permit no smoothing. In a sense, these explicit discontinuities could be viewed as a special case of the models in the previous section in which the weights are restricted to be binary-valued. However, most treatments of explicit discontinuity models impose the additional constraint that the discontinuities form a *closed boundary*. It is rare in practice to know where this boundary is, so the boundary must also be estimated in the filtering process. Unfortunately, this additional unknown variable usually destroys the convexity of the implicit discontinuity models that were previously considered, forcing the estimation of local minima.

When the discontinuities form a closed boundary, the standard approach is to view the filtered values inside the region as a window into a foreground function, x, and the values outside the region as a window into a background function, y. Therefore, in an explicit discontinuity model with a closed boundary, one seeks to find the optimum value of both the reconstructed foreground variable and the reconstructed background variable.

The prototypical energy for filtering with explicit discontinuities is the "piecewise smooth" Mumford–Shah model [289] (strongly related to the Geman and Geman model of [145] and the "weak membrane model" of Blake and Zisserman [39]). In this work, we follow the level set literature to consider the piecewise smooth model [289, 384], formulated as

$$\mathcal{E}_{\mathrm{MS}}[x_{\mathcal{R}}, x_{\bar{\mathcal{R}}}; \mathcal{R}] = \int_{\mathcal{R}} \|\nabla x_{\mathcal{R}}\|_2^2 \, \mathrm{d}t + \int_{\bar{\mathcal{R}}} \|\nabla x_{\bar{\mathcal{R}}}\|_2^2 \, \mathrm{d}t + \lambda \int_{\mathcal{R}} |x_{\mathcal{R}} - s|^2 \, \mathrm{d}t + \lambda \int_{\bar{\mathcal{R}}} |x_{\bar{\mathcal{R}}} - s|^2 \, \mathrm{d}t + \nu \Gamma(\mathcal{R}), \quad (5.35)$$

where we introduce the new variable $\mathcal{R}$ which is a subset of the domain and the function $\Gamma(\mathcal{R})$ measures the boundary length of the set. In a more generalized context, we may consider $\mathcal{R} \subseteq \mathcal{V}$, represented by a binary-valued 0-chain, $\mathbf{r}$, indicating membership in $\mathcal{R}$, i.e., $r_i = 1$ if $v_i \in \mathcal{R}$ and $r_i = 0$ otherwise. In this more generalized setting, we may define $\Gamma(\mathcal{R})$ as

$$\Gamma(\mathcal{R}) = \sum_{e_{ij}} w_{ij} |r_i - r_j|. \quad (5.36)$$

When $w_{ij} = 1$ everywhere, this definition equates $\Gamma(\mathcal{R})$ with the number of edges spanning $\mathcal{R}$ and $\bar{\mathcal{R}}$ (although any affinity weighting function could also be applied). This model implicitly assumes that the data may be divided into two groups (e.g., "foreground" and "background"), although more complex models have been considered as well [389] which could be easily adapted to a discrete framework. In the context of a 4-connected image lattice embedded into the two-dimensional Euclidean plane in the usual fashion, the definition of $\Gamma(\mathcal{R})$ given in (5.36) measures the boundary of the region defined by $\mathcal{R}$ with an ℓ_1 metric. If a Euclidean (or other) measure were desirable, the weights could be modified to reflect the desired metric, as described in Chap. 4.

Following the treatment in [163], the discrete formulation of (5.35) may be given as

$$\mathcal{E}_{\mathrm{MS}}[\mathbf{x}_{\mathcal{R}}, \mathbf{x}_{\bar{\mathcal{R}}}; \mathbf{r}] = \mathbf{r}^{\mathsf{T}} |\mathbf{A}^{\mathsf{T}}| |\mathbf{A}\mathbf{x}_{\mathcal{R}}|^2 + (\mathbf{1} - \mathbf{r})^{\mathsf{T}} |\mathbf{A}^{\mathsf{T}}| |\mathbf{A}\mathbf{x}_{\bar{\mathcal{R}}}|^2 + \lambda \mathbf{r}^{\mathsf{T}} |\mathbf{x}_{\mathcal{R}} - \mathbf{s}|^2 + \lambda (\mathbf{1} - \mathbf{r})^{\mathsf{T}} |\mathbf{x}_{\bar{\mathcal{R}}} - \mathbf{s}|^2 + \Gamma(\mathcal{R}). \quad (5.37)$$

Although this formulation is no longer convex, additional information is gained by finding a solution for the explicit boundary, $\mathcal{R}$, which allows interpretation of a minimum for (5.37) as a segmentation or clustering algorithm. Consequently, this model will be discussed in greater detail in Chap. 6. Note that the Mumford–Shah model defined on a graph was previously used to establish a set of filter coefficients that could be applied iteratively to perform filtering, see [342].

5.5 Filtering by Gradient Manipulation

A different approach to filtering may be achieved by manipulating the *gradients* of the data and reconstructing a least-squares fit of the node data, $\mathbf{x}$. Specifically, if we consider the gradient vectors $v = \nabla u$, then we may apply any manipulating function to these vectors, $\eta(v)$ and then look for the new scalar-valued function for which these vectors are the gradients, $\nabla\tilde{u} = \eta(v)$. Of course, not every set of vectors is the gradient of some scalar function (i.e., the vector field may contain a curl component). Therefore, the standard approach would be to find a scalar function with a gradient field that is as close as possible to $\eta(v)$ in the least-squares sense (see, e.g, [131, 393, 410]). Specifically, the goal is to find the scalar field $\tilde{u}$ that minimizes

$$\mathcal{E}[\tilde{u}] = \int \|\nabla\tilde{u} - \eta(v)\|_2^2 \, \mathrm{d}t, \tag{5.38}$$

which takes a minimum at the solution to the Poisson equation

$$\nabla^2\tilde{u} = \nabla \cdot \eta(v). \tag{5.39}$$

Beyond least-squares minimization, other options are available for reconstructing a scalar field u from a vector field with a nonzero curl component (see [2]).

In a discrete setting, the manipulation of the gradient field occurs over the 1-cochain $\mathbf{y}$. Specifically, $\mathbf{y} = \mathbf{A}\mathbf{x}$, which is manipulated with $\eta(\mathbf{y})$ and a least-squares solution is produced via the energy functional

$$\mathcal{E}[\tilde{\mathbf{x}}] = (\mathbf{A}\tilde{\mathbf{x}} - \eta(\mathbf{y}))^{\mathsf{T}}(\mathbf{A}\tilde{\mathbf{x}} - \eta(\mathbf{y})), \tag{5.40}$$

which takes a minimum when

$$\mathbf{L}\tilde{\mathbf{x}} = \mathbf{A}^{\mathsf{T}}\mathbf{y}. \tag{5.41}$$

Although traditional applications of this model have not considered weighted edges, these edge weights could easily be introduced into this context. For example, the least-squares solution of the problem in (5.41) would simply be modified to solve the Poisson equation

$$\mathbf{L}\tilde{\mathbf{x}} = \mathbf{A}^{\mathsf{T}}\mathbf{G}^{-1}\mathbf{A}\tilde{\mathbf{x}} = \mathbf{A}^{\mathsf{T}}\mathbf{G}^{-1}\mathbf{y}. \tag{5.42}$$

5.6 Nonlocal Filtering

Recent studies have suggested using nonlocal neighborhood relationships to perform filtering [15, 63]. Instead of an edge set based upon a local neighborhood and gradient-based weighting, these methods have advocated for employing a fully-connected graph in which each edge weight is dependent upon the statistical relationship or similarity of the data within a local neighborhood of each node. In other

words, each node is considered to have a *local* neighborhood and a *distant* neighborhood (which is a superset of the local neighborhood), where the edges of the distant neighborhood derive their edge weights by comparing local neighborhoods. In this way, weights form connections at two different scales. The principle with these nonlocal techniques is that many patterns are repeated throughout a dataset and therefore the restoration of the pattern at one location can benefit from looking at patterns from other locations. Of course, the introduction of a fully-connected graph makes the computation intense and slow. Despite this computational hurdle, the quality of the results is sufficiently impressive that the technique remains an active area of research.

Since this idea was introduced in the field of image processing, a regular data grid is assumed in which it is possible to measure neighbors in a small window around two pixels. Specifically, the weights can be derived as

$$w_{ij} = \sigma(\|\mathbf{x}_{\text{nbhd}(i)} - \mathbf{x}_{\text{nbhd}(j)}\|), \tag{5.43}$$

where $\sigma(\cdot)$ is any affinity weighting function discussed in Chap. 4 and $\mathbf{x}_{\text{nbhd}(i)}$ represents the collection of data values, arranged into a vector, from the nodes in the local neighborhood of v_i. The weights generated in this fashion may be used directly in any of the methods described in previous sections. This distinction between local and distant neighborhoods also carries over easily to arbitrary regular graphs, but not to irregular graphs. One possible avenue for generalization of these methods to irregular graphs is to modify $\sigma(\cdot)$ to input two distributions as arguments and output a distance between those distributions. For example, $\sigma(\cdot)$ could measure the difference in entropy of the data values in the two local neighborhoods, even if the neighborhoods were of different size. Another approach for generalization of this method to irregular graphs is to consider two nodes to be distant neighbors only if they have the same degree.

5.7 Filtering Vectors and Flows

The focus of the previous sections has been the filtering of 0-cochains or functions defined on nodes. In this section we discuss how to apply these same techniques to filtering 1-cochains or functions defined on edges, such as vectors or flows. For the remainder of this section, we will refer to all 1-cochains as flows. We begin our application into flow filtering by making the same assumption as we did with the filtering of scalar fields—that the target of our filtering operation is to suppress high frequencies. This assumption can be motivated by the same arguments as in Sect. 5.1.2 if the 1-cochains are acquired directly (i.e., if our acquisition device measures flows, differences, gradients or other vectors). The filtering of high frequency components in a flow field is identical to the case of filtering a scalar field: *low-pass filtering entails suppressing the components corresponding to high frequency eigenvectors of the Laplacian*. Instead of suppressing the high frequencies

of the *scalar* Laplacian, flow filtering suppresses the high frequencies of the *vector* Laplacian discussed in Chap. 2.

Recall that the continuous formulation of the vector Laplacian is

$$\boldsymbol{\nabla}^2 = (\nabla\nabla\cdot) - (\nabla \times \nabla\times), \tag{5.44}$$

and the (unweighted) edge Laplacian

$$\mathbf{L}_1 = \mathbf{B}^\mathsf{T}\mathbf{B} + \mathbf{A}\mathbf{A}^\mathsf{T}, \tag{5.45}$$

where $\mathbf{B}$ is the face–edge incidence matrix. Although many aspects of our smoothing models for scalar fields translate directly, there are also important differences. We can note several aspects of the edge Laplacian in the context of filtering:

1. The node Laplacian for a connected graph has a rank one nullspace corresponding to the constant vector, which means that a diffusion process governed by the scalar Laplacian will always approach a constant. The edge Laplacian has a rank zero nullspace when $|\mathcal{F}| = |\mathcal{E}| - |\mathcal{V}| + 1$ (see Chap. 4), meaning that "diffusion" with the edge Laplacian will always drive the solution to zero.
2. The edge structure may also be shift-invariant for several common types of complex. For example, a wrapping (or infinite) lattice will have a circulant edge Laplacian (assuming that the cycle set consists of all the local cycles). If the edge Laplacian is circulant, then the DFT may be used to efficiently filter flows, as described above for the case of nodal filtering.
3. From the definition of the edge Laplacian, it is clear that a "smooth" flow field is one for which there is a small curl *and* a small divergence. In other words, the ideal smooth flow is one in which all of the flow vectors are pointing in the same direction. Figure 5.3 illustrates the principle of vector smoothing and Fig. 5.4 gives an example on an arbitrary graph. These examples illustrate vector "diffusion" in which the initial high-frequency flow field is smoothed.

Thus filtering higher-order data is a straightforward generalization of the intuitive filtering of nodal data presented above.

> The filtering of a flow field or any edge data is possible using the same formalism for filtering scalar fields or node data by using the eigenvectors of the edge Laplacian.

This direct extension of filtering to higher-order data provides a clear example of the generality of the discrete framework.

5.7.1 Translating Scalar Filtering to Flow Filtering

If the edge Laplacian is circulant, then standard DFT filtering techniques may be applied since the vectors in the DFT matrix will also be the eigenvectors of the edge

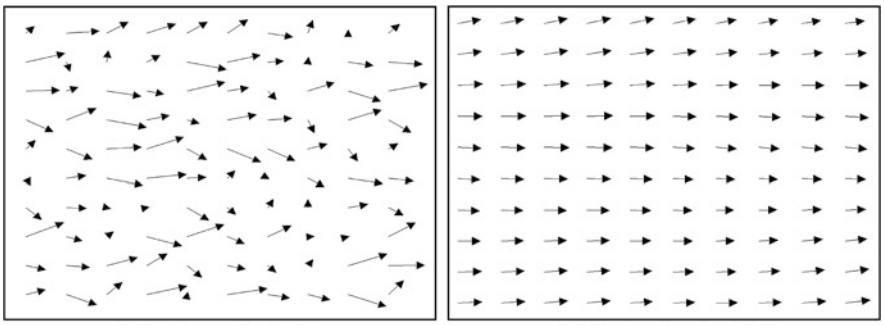

Fig. 5.3 An example of vector smoothing in the plane. (**A**) The initial, noisy vector field with nonzero curl and divergence throughout the field. (**B**) The vector field smoothed via diffusion (the Basic Energy Model applied to flows (5.49b) with $p = 2$). In the Euclidean plane, this smoothing is equivalent to smoothing the two coordinate components of each vector independently (i.e., treating both as scalar fields). In general, vector smoothing reduces both the curl *and* divergence of the vector field, and adds spatial coherence or correlation to the vector-valued data

Laplacian. However, if the edge Laplacian is not circulant, then we may still apply Taubin's smoothing method to the flow field. Specifically, if we have flow field $\mathbf{y}$, then we may alternate "diffusion" and "unshrinking" steps via

$$\mathbf{y}^{[2k+1]} = \mathbf{y}^{[2k]} - \lambda \mathbf{L}_1 \mathbf{y}^{[2k]}, \tag{5.46}$$

$$\mathbf{y}^{[2k+2]} = \mathbf{y}^{[2k+1]} + \mu \mathbf{L}_1 \mathbf{y}^{[2k+1]}. \tag{5.47}$$

In order to utilize the gradient-based filtering techniques that we defined for scalar functions, we need to ask how to define a "gradient" on a vector/flow field. A natural choice of the "gradient" of a flow field might be to view the gradient as the 0-coboundary operator (see Chap. 2) and simply replace it with the 1-coboundary operator, i.e., the curl operator $\mathbf{B}$. However, such an approach would view a minimization of the gradient operator as the goal rather than viewing the gradient operator as a proxy for dampening the high-frequency eigenvectors of the Laplacian, as originally derived in (5.10b). For scalar functions, the Laplacian consists of $\mathbf{L} = \mathbf{A}^\mathsf{T}\mathbf{A}$ and therefore an iterative reduction of $\mathbf{1}^\mathsf{T}|\mathbf{Ax}|^p$ will have the effect of filtering high frequencies in the initial data. However, when we consider the edge Laplacian, then $\mathbf{L}_1 = \mathbf{B}^\mathsf{T}\mathbf{B} + \mathbf{A}\mathbf{A}^\mathsf{T}$ and we can see that simply replacing gradient with curl, or replacing a minimization of $\mathbf{1}^\mathsf{T}|\mathbf{Ax}|^p$ with $\mathbf{1}^\mathsf{T}|\mathbf{By}|^p$, addresses only the first term of $\mathbf{L}_1$. Therefore, if we intend to extend the gradient-based techniques to flow filtering, we must minimize both $\mathbf{1}^\mathsf{T}|\mathbf{By}|^p$ *and* $\mathbf{1}^\mathsf{T}|\mathbf{A}^\mathsf{T}\mathbf{y}|^p$, i.e., in order to dampen high frequencies we must simultaneously minimize both the curl and the divergence of the flow field.

We may now reformulate the gradient-based techniques employed in Sect. 5.2 for scalar functions in the context of flow field filtering by again considering the energy minimization models in this context. The formulation of the Basic Energy

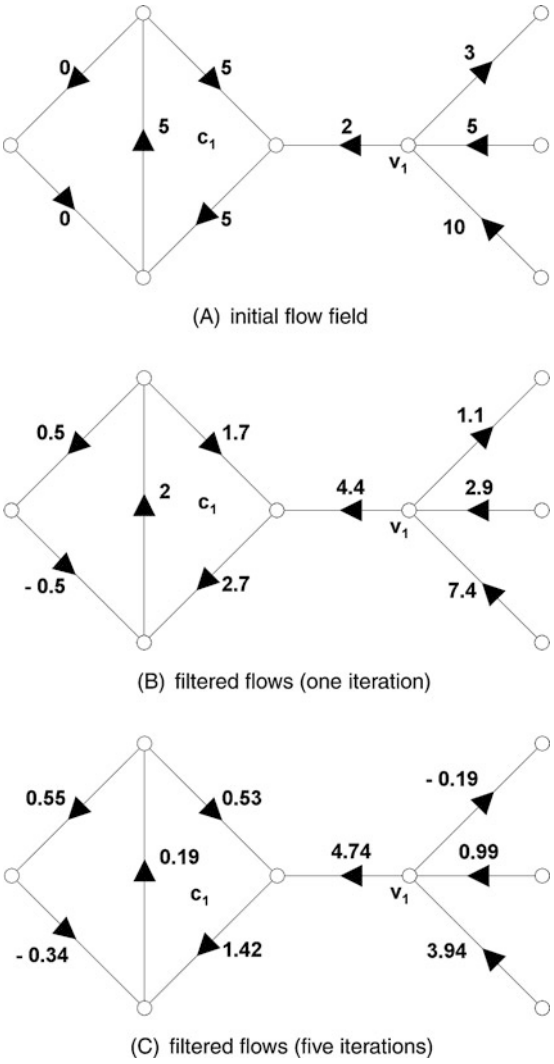

Fig. 5.4 An example of flow filtering in an arbitrary graph. A lowpass flow filtering process attempts to reduce both flow divergence and flow curl. Node v_1 and cycle c_1 have been designated as locations where the initial flow field has high divergence and curl, respectively. (**A**) The initial flow field. Note the strongly nonzero divergence at v_1, equaling $10+5-2-3=10$ and the nonzero curl around cycle c_1 equaling $5+5+5=15$. (**B**) The vector field smoothed via diffusion after one iteration (forward Euler on the Basic Energy Model (5.49b) with $p=2$ and timestep $\Delta t = 0.1 = 1/\lambda$). Even after one iteration, the divergence at v_1 has been reduced to $7.4+2.9-4.4-1.1=4.8$ and the curl around cycle c_1 has been reduced to $2+1.7+2.7=6.4$. (**C**) The vector field smoothed via diffusion after five iterations (forward Euler on the Basic Energy Model for flows (5.49b) with $p=2$ and timestep $\Delta t = 0.5 = 1/\lambda$). The divergence at v_1 has been reduced to $3.94+0.99+0.19-4.74=0.38$ and the curl around cycle c_1 has been reduced to $1.42+0.53+0.19=2.14$

Model for flow filtering therefore gives

$$\mathcal{E}_{\mathrm{BEM}}[y] = \int \|\nabla \cdot \vec{y}\|_p^p \, \mathrm{d}t - \int \|\nabla \times \vec{y}\|_p^p \, \mathrm{d}t, \tag{5.48a}$$

$$\mathcal{E}_{\mathrm{BEM}}[\mathbf{y}] = \mathbf{1}^{\mathsf{T}} |\mathbf{B}\mathbf{y}|^p + \mathbf{1}^{\mathsf{T}} |\mathbf{A}^{\mathsf{T}}\mathbf{y}|^p. \tag{5.48b}$$

Recall that the sign discrepancy between the continuous and discrete formulations was addressed in Chap. 2. In the scalar case, the minimum of $\mathcal{E}_{\mathrm{BEM}}[\cdot]$ was trivial (constant) and this case is no different (zero). However, just as in the scalar case,

a few steps of an iterative minimization algorithm for (5.48b) will serve to quickly dampen the highest frequencies.

In the scalar case we employed a data term to avoid a trivial minimum of our gradient term and the same technique may be used again for flow fields. If we have a noisy observation of our flow field (represented by function $\mathbf{s}$), then we may trade off between smoothness and the noisy data via a minimization of

$$\mathcal{E}_{\text{EBEM}}[y] = \int |\nabla \cdot \vec{y}|^p \, \mathrm{d}t - \int \|\nabla \times \vec{y}\|_p^p \, \mathrm{d}t + \lambda \int |\vec{y} - \vec{s}|^p \, \mathrm{d}t, \tag{5.49a}$$

$$\mathcal{E}_{\text{EBEM}}[\mathbf{y}] = \mathbf{1}^\mathsf{T} |\mathbf{A}^\mathsf{T}\mathbf{y}|^p + \mathbf{1}^\mathsf{T} |\mathbf{B}\mathbf{y}|^p + \lambda \mathbf{1}^\mathsf{T} |\mathbf{y} - \mathbf{s}|^p. \tag{5.49b}$$

All of the above filtering techniques may be modified to include implicit discontinuities (weights) by using the appropriately weighted operators. Recall from Chap. 2 that the weighted edge Laplacian is given by $\mathbf{L}_1 = \mathbf{A}\mathbf{G}_0\mathbf{A}^\mathsf{T}\mathbf{G}_1^{-1} + \mathbf{G}_1\mathbf{B}^\mathsf{T}\mathbf{G}_2^{-1}\mathbf{B}$ for node weighting $\mathbf{G}_0$, edge weighting $\mathbf{G}_1$ and face weighting $\mathbf{G}_2$. Letting $\mathbf{G}_1 = \mathbf{I}$ gives

$$\mathcal{E}_{\text{BEM}}[\mathbf{y}] = \mathbf{1}^\mathsf{T}\mathbf{G}_0 |\mathbf{A}^\mathsf{T}\mathbf{y}|^p + \mathbf{1}^\mathsf{T}\mathbf{G}_2^{-1} |\mathbf{B}\mathbf{y}|^p, \tag{5.50}$$

$$\mathcal{E}_{\text{EBEM}}[\mathbf{y}] = \mathbf{1}^\mathsf{T}\mathbf{G}_0 |\mathbf{A}^\mathsf{T}\mathbf{y}|^p + \mathbf{1}^\mathsf{T}\mathbf{G}_2^{-1} |\mathbf{B}\mathbf{y}|^p + \lambda \mathbf{1}^\mathsf{T} |\mathbf{y} - \mathbf{s}|^p. \tag{5.51}$$

In the next section we use the generalizations of the filtering models to flow filtering to further extend these filtering techniques to functions defined on cells of any dimension (i.e., to filtering a general p-cochain).

5.8 Filtering Higher-Order Cochains

Now that we have examined the Fourier and variational approaches for filtering scalar (node) and vector (edge) functions, we complete the exposition by briefly considering the filtering of functions defined on higher-dimensional cells (e.g., faces). As before, the filtering of high frequencies depends on a definition of a higher-order Laplacian. As we saw in Chap. 2, the general definition of the higher-order p-Laplacian matrix for arbitrary p-cells is given as

$$\mathbf{L}_p = \mathbf{N}_p\mathbf{N}_p^* + \mathbf{N}_{p+1}^*\mathbf{N}_{p+1}. \tag{5.52}$$

If we consider a specific value of p, then we may still employ DFT-based techniques if the $\mathbf{L}_p$ matrix is circulant. Even if $\mathbf{L}_p$ is not circulant, then we may still apply Taubin's algorithm with the steps

$$\mathbf{z}^{[2k+1]} = \mathbf{z}^{[2k]} - \lambda \mathbf{L}_p \mathbf{z}^{[2k]}, \tag{5.53a}$$

$$\mathbf{z}^{[2k+2]} = \mathbf{z}^{[2k+1]} + \mu \mathbf{L}_p \mathbf{z}^{[2k+1]}, \tag{5.53b}$$

where $\mathbf{z}$ is the p-cochain variable.

Similarly, when $\mathbf{G}_p = \mathbf{I}$, the variational approaches may be defined by producing a minimum of

$$\mathcal{E}_{\text{BEM}}[\mathbf{z}] = \mathbf{1}^{\mathsf{T}}\mathbf{G}_{(p-1)}|\mathbf{N}_p^{\mathsf{T}}\mathbf{z}|^q + \mathbf{1}^{\mathsf{T}}\mathbf{G}_{(p+1)}^{-1}|\mathbf{N}_{(p+1)}\mathbf{z}|^q, \tag{5.54}$$

$$\mathcal{E}_{\text{EBEM}}[\mathbf{z}] = \mathbf{1}^{\mathsf{T}}\mathbf{G}_{(p-1)}|\mathbf{N}_p^{\mathsf{T}}\mathbf{z}|^q + \mathbf{1}^{\mathsf{T}}\mathbf{G}_{(p+1)}^{-1}|\mathbf{N}_{(p+1)}\mathbf{z}|^q + \lambda\mathbf{1}^{\mathsf{T}}|\mathbf{z} - \mathbf{s}|^q. \tag{5.55}$$

Note that we used q as the exponent parameter to avoid confusion with p used to designate the p-cell. The above equations detail the more general case of variational filtering methods with implicit boundaries (weights). If we desired to use the variational filtering methods without implicit boundaries (unweighted) we may simply set $\mathbf{G}_{(r+1)} = \mathbf{G}_r = \mathbf{G}_{(r-1)} = \mathbf{I}$ in the equations above.

5.9 Applications

The filtering procedures described in this chapter may be applied to any situation in which data measured at discrete locations contains noise to be removed. In this section, we consider several applications of the filtering procedures applied to very different types of data. At the end of these experiments, we summarize our observations to give the reader a guide for when to employ the various filtering techniques. In all of the experiments presented here using the total variation model, we set $p = 1$, since setting $p = 2$ in the total variation model is equivalent to setting $p = 2$ in the Extended Basic Energy Model. Unless otherwise noted, the total variation filtering used an unweighted edge model. The λ parameter was set individually for each algorithm.

5.9.1 Image Processing

5.9.1.1 Regular Graphs and Space-Invariant Processing

A classical application of filtering is image processing. In this problem domain, the pixels are identified with nodes, edges are derived from a local neighborhood (e.g., a 4-connected or 8-connected lattice), and the pixel intensities are the data associated with each node. Therefore, the $\mathbf{x}$ variable in the filtering routines above is the unknown filtered image intensities that are solved for, while the initial data $\mathbf{s}$ is identified with the noisy image intensities.

In the first experiment, a synthetic image of a black circle in a white background was corrupted with noise by adding an independent random variable to each pixel with a uniform distribution. Figure 5.5 displays the results of the various filtering procedures discussed above. We may make several observations from these results.

First, the localization of circle boundaries (i.e., the image discontinuities) improves in the Basic Energy Model as the parameter p decreases. Therefore, the

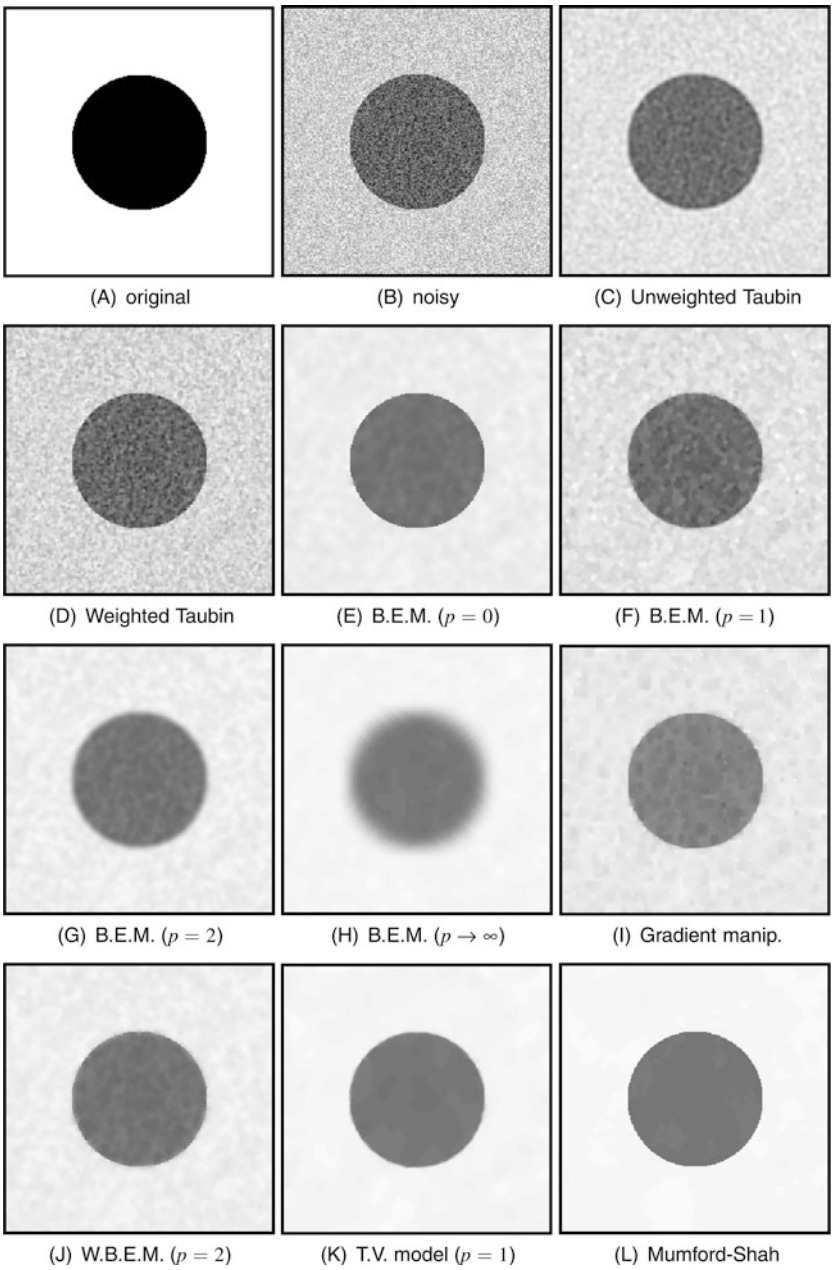

Fig. 5.5 Image filtering of a synthetic image of a black circle on a white background which has been corrupted by additive i.i.d. random noise with a uniform distribution. Note that unweighted filtering with the Basic Energy Model consistently produces worse boundary localization as p increases. However, for any filtering model, weighted filtering is generally better at preserving boundary location. Since the source image exactly matches the Mumford–Shah model, this filtering result is nearly perfect

mean filter localizes boundaries better than the minimax filter, the median filter localizes boundaries better than the mean filter and the (approximated) mode filter localizes boundaries better than the median filter. A different method of providing better localization boundaries is to use a *weighted* filter, as seen by the improvement in the Taubin technique and the improvement in the mean filter attained by using edge weights obtained from the Welsch function in Chap. 4. In the gradient manipulation example, the gradients were manipulated by setting all gradients to zero for which the magnitude was smaller than a fixed threshold. Since this image content allowed such a simple approach, the reconstruction was nearly perfect from this algorithm. Finally, we note that this example perfectly fits the model of the Mumford–Shah functional in the sense that the underlying image consists of two objects (foreground and background) with different intensities. If the boundary between these two objects may be localized well, then this filtering procedure smooths only within these boundaries to achieve a near-perfect filtered result.

The second experiment uses a photograph which is much more complicated than the circle in the previous experiment (see Fig. 5.6). Unlike the previous image, the noiseless image contains significant high spatial frequency content (in the child's hair). In this experiment, zero-mean Gaussian noise was added independently to each pixel to corrupt the original image. As before, we see that as p increases in the Basic Energy Model, the boundaries are progressively blurred in the filtered image. Additionally, the use of a weighted graph increases the boundary localization for the Taubin and mean filtering methods. Although the total variation model continues to produce a good filtering, the Mumford–Shah model does not perform well on this image. The Mumford–Shah model explicitly assumes that there are two regions (possibly consisting of multiple connected components) which are smoothly varying in intensity except at the boundary transitions. Since the example image contains many regions of sharp intensity changes (textures), the Mumford–Shah model is forced to choose (through parameter settings) between many small regions or large overly smoothed regions. The parameters were set in this experiment to produce many small regions. Results of this experiment are displayed in Fig. 5.6.

5.9.1.2 Space-Variant Imaging

Although standard image processing applies to images which are uniformly sampled, there are several situations in which the image data is acquired with nonuniform samples. Some image acquisition devices explicitly acquire data which does not have a Cartesian sampling (e.g., ultrasound medical images). Additionally, almost all known biological vision systems acquire light data nonuniformly in space [209]. Although most biological vision systems employ sampling schemes which are difficult to describe mathematically as a function of space, there has been more success in mathematically describing the sampling of visual space employed by humans and by non-human primates such as the macaque monkey. The macaque is of particular interest because it is considered to have a similar retinal organization to humans and similar visual capabilities [336], and there is vast amounts of data on the

Fig. 5.6 Image filtering of a photograph containing high-frequency texture. This image has been corrupted by i.i.d. random noise with a zero-mean Gaussian distribution. As before, weighted filtering is generally better at preserving boundary location. However, since this image does not match well with the Mumford–Shah model, the filtered image is not nearly as close to the noiseless image as it was in the previous example

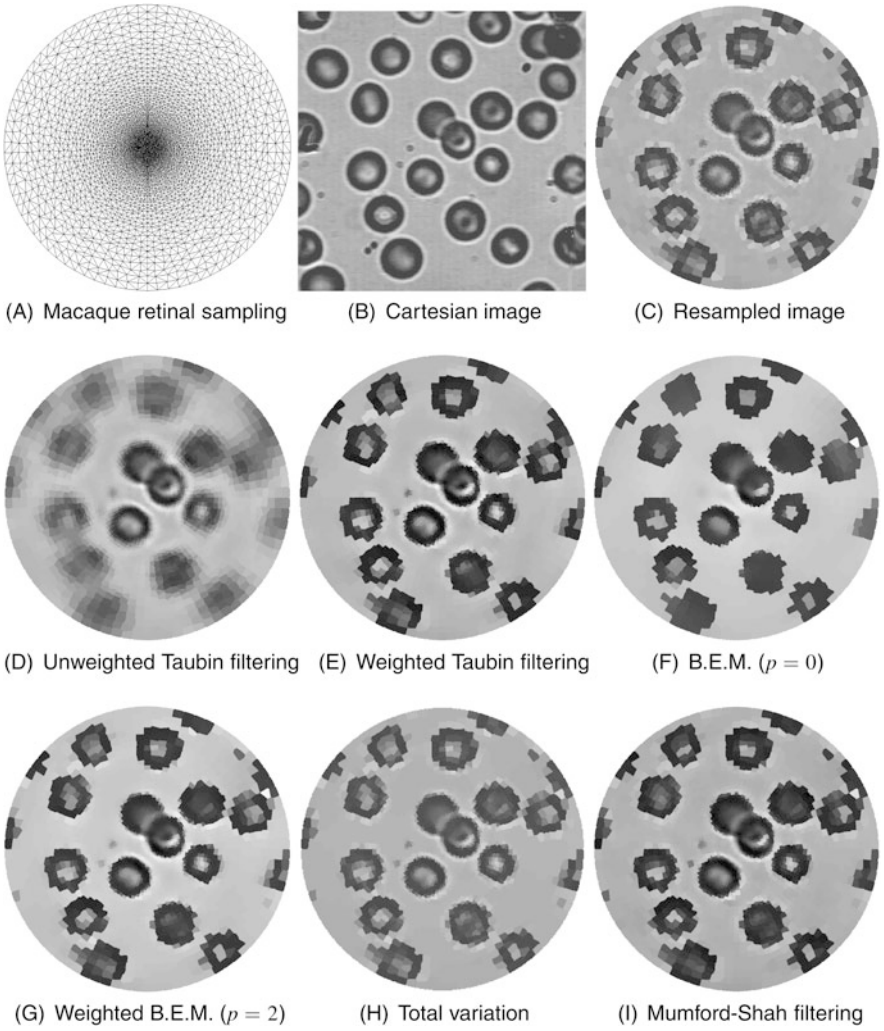

Fig. 5.7 Filtering image data on a biologically sampled image. (**A**) A sampling mesh modeled after the macaque retina. (**B**) Cartesian image. (**C**) Image resampled with the macaque mesh, (**D**)–(**I**) filtering of the data. Note the visual disturbance caused by blurring edges in the poorly sampled peripheral regions in (D)

macaque visual system. Several researchers have taken inspiration from this nonuniform biological sampling of visual space to pursue computer vision approaches or hardware with a similar sampling [158, 277, 321, 328].

We may filter these nonuniform biological samplings of image data in the same framework as before. As with Cartesian sampling, the image sample locations are viewed as nodes, the edge structure is defined by a Delaunay triangulation in the Euclidean plane and the filtered image data $\mathbf{x}$ is associated with each node (see

[158, 391] for more information). Aside from this new graph, there is *no difference in filtering operation with the standard Cartesian data*. In fact, exactly the same software implementation may be used to perform filtering by simply applying it to the new (non-lattice) graph. For this experiment, the nonuniform sampling structure for the macaque was loaded from the Graph Analysis Toolbox software package [166] that contains an implementation of the filtering techniques discussed in the chapter. Using this same toolbox, a standard Cartesian image of blood cells was imported to the space-variant structure and filtered. Results of the experiment are displayed in Fig. 5.7. Since the same properties of the filtering procedures observed for the Cartesian images apply to the space-variant images as well, only a subset of the filtering methods were employed for this experiment. However, we may make a few observations which are specific to this experiment. The first observation is that the blurring over object boundaries is more visually disturbing in regions of the image which are represented by only a few samples (the periphery in these images). Our second observation is that although the edge weights continue to help avoid blurring over object boundaries, the edge weights may be set in this scenario based on both image data and graph geometry. Therefore, edges connecting nodes which are further apart spatially may be given a lower weight, in addition to a weighting based on intensity difference (see Chap. 4 for more details). In this experiment, edge weights were generated purely from intensity changes in the image.

5.9.2 Three-Dimensional Mesh Filtering

Filtering of geometric data is an important process in computer graphics and the processing of data obtained from various three-dimensional scanners. Is this context, the node data (0-cochain) is a tuple of coordinates assigned to each node. Therefore, the nodal variable $\mathbf{x}$ in the above algorithms corresponds to an $n \times K$ set of K-dimensional filtered coordinates assigned to each node, with $\mathbf{s}$ corresponding to the $n \times K$ noisy coordinate values acquired for each node. The output of a filtering procedure is therefore a new set of coordinates for each node. The edge structure of the graph is generally given via a surface extraction preprocessing step. Since the most common method for rendering three-dimensional data requires a list of faces for the surface, the faces are usually extracted via a triangulation process.

We begin this section with a synthetic example of filtering coordinates obtained by generating a circle and adding noise to the coordinates of each point on the circle. Figure 5.8 gives an example of a lowpass filter obtained via the Basic Energy Model with $p = 2$ (i.e., a mean filter) and a lowpass filter given by Taubin's method. By subtracting the lowpass coordinates from the original coordinates, a highpass filter of this ring is obtained.

5.9.2.1 Mesh Fairing

The problem of **mesh fairing** is to produce a smooth three-dimensional mesh from a mesh with noisy coordinate values. In the current framework, the mesh points are

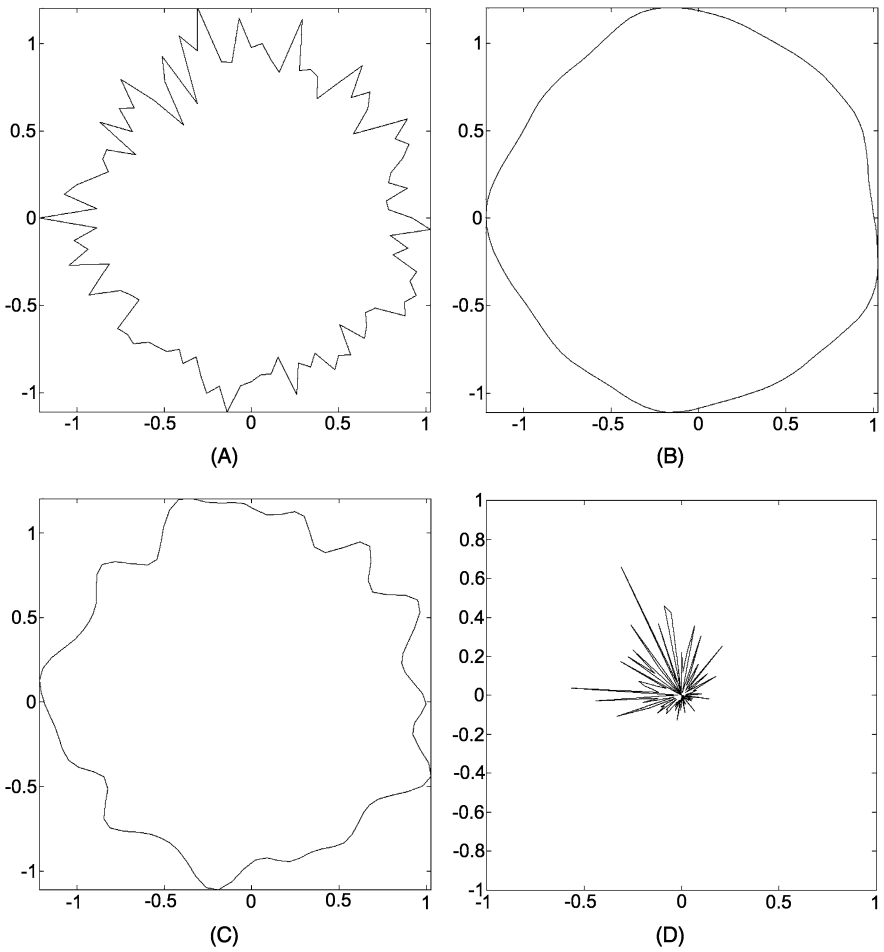

Fig. 5.8 Filtering coordinate data on a ring graph. (**A**) A noisy ring graph produced by adding Gaussian random noise to the radius of nodes arranged in a perfect circle. (**B**) The effect of applying the Basic Energy Model with $p = 2$ (i.e., a mean filter) to the coordinates of the graph in (A). (**C**) The low-pass filter of Taubin [371] applied to the coordinates of the graph in (A). (**D**) A high-pass filter of the coordinates in (C), produced by differencing the low-pass signal of (C) with the original

associated with nodes, the edges are given explicitly by the mesh, and the data tuple $\tilde{s}_i$ associated with each node v_i represents the three-dimensional coordinates of that node. The goal in mesh fairing is to produce filtered data (coordinates) $\mathbf{x}$.

Figure 5.9 shows a three-dimensional mesh of a horse. The noise observed in meshes is typically in the direction of the surface normal. For this example, Gaussian noise was added to each of the three coordinates and to each node independently to generate the noisy mesh. Several of the filtering procedures described in this chapter were applied to produce a fairer mesh. In this application, it is possible

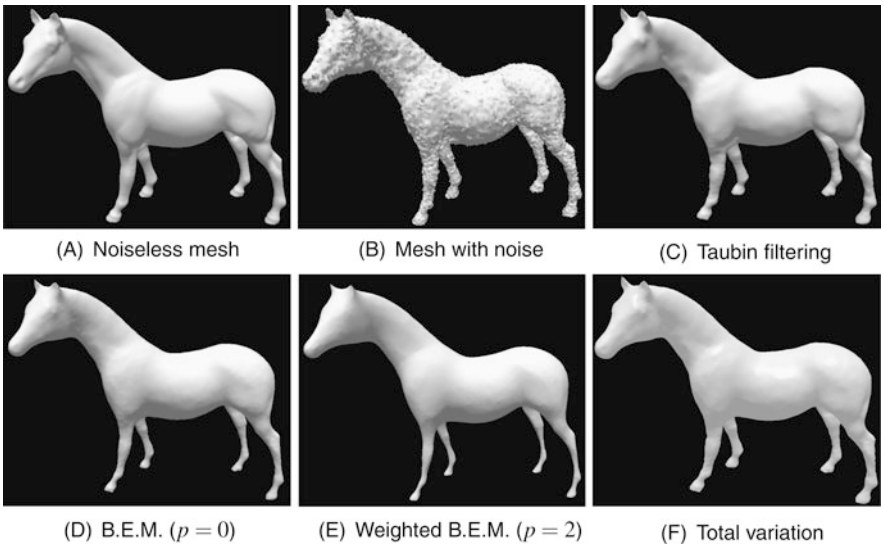

Fig. 5.9 An example of mesh fairing. Gaussian noise was added to each of the coordinates of a three-dimensional mesh and these coordinates were filtered to produce a faired (smoothed) mesh. The nodes and edges are given explicitly by the mesh and the data are the three-dimensional node coordinates. Note how the Basic Energy Models shrink portions of the figure while Taubin's spectral filtering smooths without shrinking by preserving the low frequencies

to see that one of the major benefits of Taubin's spectral approach to filtering is to avoid shrinking of the mesh. Since the Basic Energy Model drives the filtered solution toward a constant value (regardless of the choice of p), the effect observed in mesh fairing is to drive all of the nodes closer together (to the same, constant, location in space). However, by preserving the low frequencies, Taubin's filtering approach avoids the shrinking observed from the other algorithms. The shrinking is particularly noticeable in these figures around the horse's legs, ears and snout. The total variation filtering results in less shrinking than the Basic Energy Model, but smooths the fine details somewhat more than Taubin's filtering algorithm.

5.9.3 Filtering Data on a Surface

In many applications, data is measured at spatial locations along a surface, and to interpret this data properly the analysis must be carried out in a way that respects how the data is distributed along the surface. Examples of such measurements would be data collected from a network of touch sensors on an article of clothing, or samples of the distribution of current along a conductive sheet. Here we consider the application of filtering functional Magnetic Resonance Imaging (fMRI) data measurements of neural activity from positions along the surface of the cerebral cortex of the brain.

We will consider an example taken from an fMRI study that sought to locate brain areas implicated in the processing of vision and, in particular, those areas that

are known to be responsible for the processing of *motion* in the visual field. In this example, the neural responses were measured during a visual stimulation consisting of presenting subjects with patterns of moving shapes, or *motion stimuli*, to activate those areas of the visual cortex responsible for processing motion so that these areas could be identified and located within the cortex. Neural "activity" is quantified through statistical analysis of the measured responses, and those measured locations whose statistical significance exceeds a fixed threshold are considered to be the sites of true activations. These activations can be visualized with *activation maps* that depict where on the cortical surface significant activation has been identified. However, random noise in the measurement leads to spurious significant activations by chance, which generates false positives in the activation maps that must be detected and removed for proper interpretation of the data. Because for most experiments it is expected that groups of active locations are nearby in space, spatial filtering of the data helps coalesce locations of true activity while suppressing the significance levels of spurious activity at isolated nodes, forcing them below the significance threshold and thus eliminating them from the final activity map. This spatial prior is often used in fMRI analysis (e.g., [406]).

MRI data is acquired in the form of a stack of images of the brain, thus the measurement consists of a volume of image data. Most techniques for spatial smoothing of fMRI data smooth the data in the original space of the acquired images, i.e., the volumes of image data represented as voxels, and therefore the conventional smoothing can be conveniently enacted by three-dimensional smoothing kernels applied to the volume of image data. Unfortunately much of the spatial structure of the relevant neuronal activity patterns is contained within the surfaces of the brain, such as the cortical gray matter of the cerebral hemispheres where most of the sensory, motor, and higher cognitive functions take place. Smoothing the voxel data in three dimensions is harmful since voxels that are nearby in three dimensions are often sampled from positions on the cortical surface that are far apart when distance is measured *along* the two-dimensional cortical surface—as in the case of two adjacent voxels that sample from opposite, abutting banks of a sulcus. Thus, volumetrically smoothing the voxel data and ignoring the boundary of the cortical surface can mix activity patterns across distant locations of the cortical surface, corrupting the spatial structure of local activity patterns existing along the surface. For this reason, it is advantageous to smooth the data in a way that respects the natural geometry of the cortical surface.

An example of surface smoothing applied to brain activation maps measured with fMRI is presented in Fig. 5.10 (contained in the color plate section at the end of the book). In surface-based fMRI analysis, a mesh representation of the cortical gray matter of the cerebral hemispheres is generated from anatomical MRI data (e.g., [96, 136]), including the two-dimensional exterior and interior boundaries of the gray matter ribbon. For this example data set, the exterior surface is shown in Fig. 5.10(A) and the corresponding interior surface is shown in Fig. 5.10(B). All analysis is restricted to the interior surface of the cortical gray matter, and for ease of visualization the activation maps are typically presented on an "inflated" surface representation (as shown in Fig. 5.10(C)) to reveal the activity buried within the deep sulci of the cortical folds.

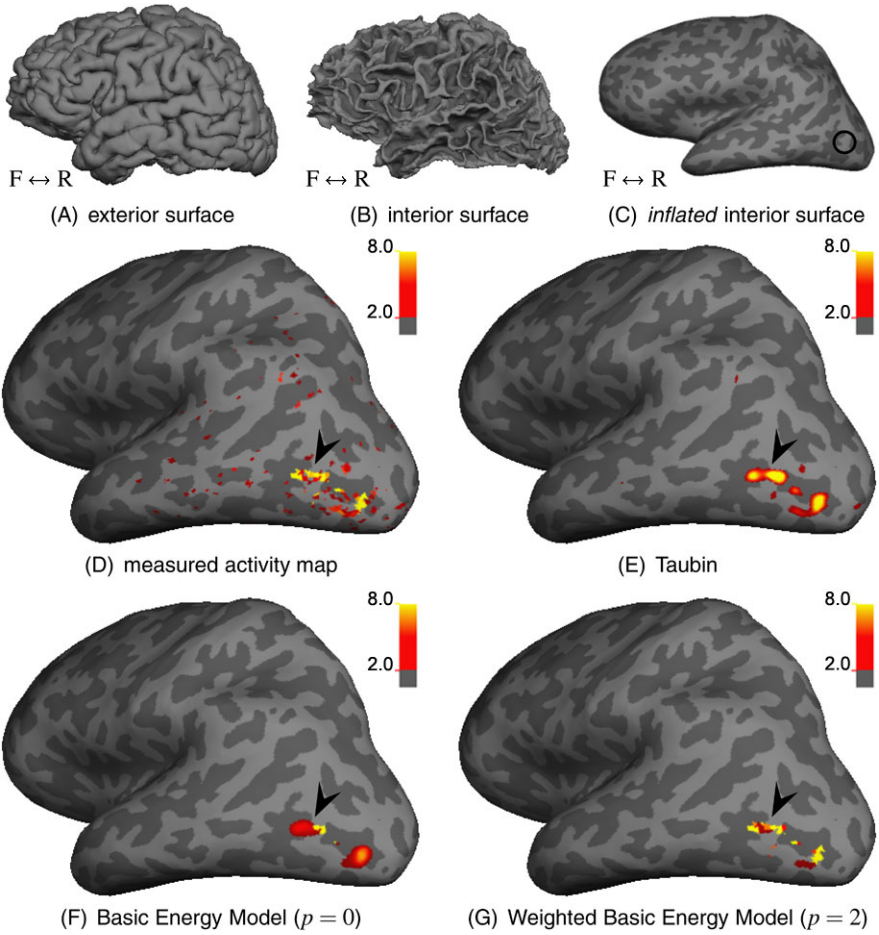

Fig. 5.10 Filtering fMRI data along a cortical surface model. Surface models of the (**A**) exterior surface, (**B**) interior surface, and (**C**) the "inflated" interior surface of the cortical gray matter of the left cerebral hemisphere, with approximate location of area MT indicated by a *circle*. (Surfaces generated with FREESURFER [96, 136].) The *legend* indicates Front–Rear axis of brain. Locations of negative mean curvature (within sulci) are rendered in the *dark gray* and locations of positive mean curvature (within gyri) are rendered in *light gray*. Measured activity map plotted as z-statistics, with color scale provided at *upper right*. The threshold is set to exclude nodes where the activity is not statistically significant, which leads to many isolated points or small clusters of activation appearing in the map—likely false positives due to noise. The results of filtering the data using (**E**) spectral filtering, (**F**) the Basic Energy Model with $p = 0$ and (**G**) $p = 2$, (**H**) Total Variation, and (**I**) the Mumford–Shah algorithm are provided with the same color scale representing the statistical significance. Note that many of the false positives are removed with the filtering. *Arrows* indicate the site of MT activation

The functional activation data from this example is represented on the vertices of the triangular mesh surface representation of the interior surface shown in Fig. 5.10(B), then smoothed with the filtering methods discussed in this chapter, and

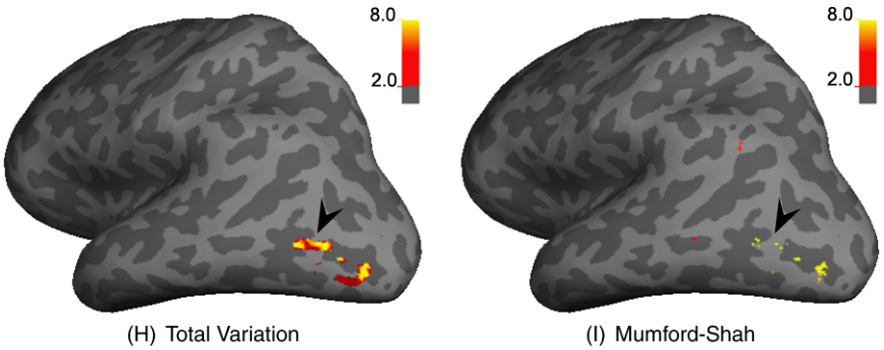

Fig. 5.10 (Continued)

the results are visualized on the inflated surface. In the original, unsmoothed data seen in Fig. 5.10(D), we see a cluster of activated nodes in the rear of the brain—which is within the part of the brain that is responsible for vision—accompanied by noisy activations extending up and further into the front of the brain. In order to remove these noisy activations while (ideally) retaining the true activations, surface-based smoothing can effectively highlight the true activity while removing noise.

Each of the filtering methods succeeds in suppressing false activations attributable to noise. The results of spectral filtering of Taubin, the two energy models, and Total Variation shown in Figs. 5.10(E)–(H) highlight two loci of activity in locations near areas where visual motion processing is known to occur known as the "middle temporal" area, or cortical area MT. Beyond removing spurious or noisy activations outside of the visual motion area, the spectral filtering shown in Fig. 5.10(E) also smooths the "true" activity pattern within MT, suggesting that some of the relevant features of the data may be lost along with the false positives. However, this filtering may also aid in eliminating aliasing artifacts in the measurement due to the coarse spatial sampling, and thereby the smoothing process may potentially recover a more faithful representation of the true activation pattern. The results of the basic energy model shown in Fig. 5.10(F) contain a distinct discontinuity that is not salient in the original measurement, which is a sharp feature that violates the expected spatial resolution of the fMRI technique and therefore is likely to be an artifact of the smoothing.

The results of the weighted Basic Energy Model and Total Variation shown in Figs. 5.10(G) and (H) demonstrate both suppression of false positives outside of the presumed true site of activation and retention of most of the structure of the original measured activity map. Therefore, if another form of filtering were desired for the remaining activity cluster, such as additional anti-aliasing filtering, it could be subsequently applied.

The results of the Mumford–Shah algorithm shown in Fig. 5.10(I) drive most of the measured activity below the statistical threshold, and thereby suppresses all but a small island of activity. To gain insight into the relative performance of these filtering methods, the results of Fig. 5.10 are re-plotted in Fig. 5.11 (contained in

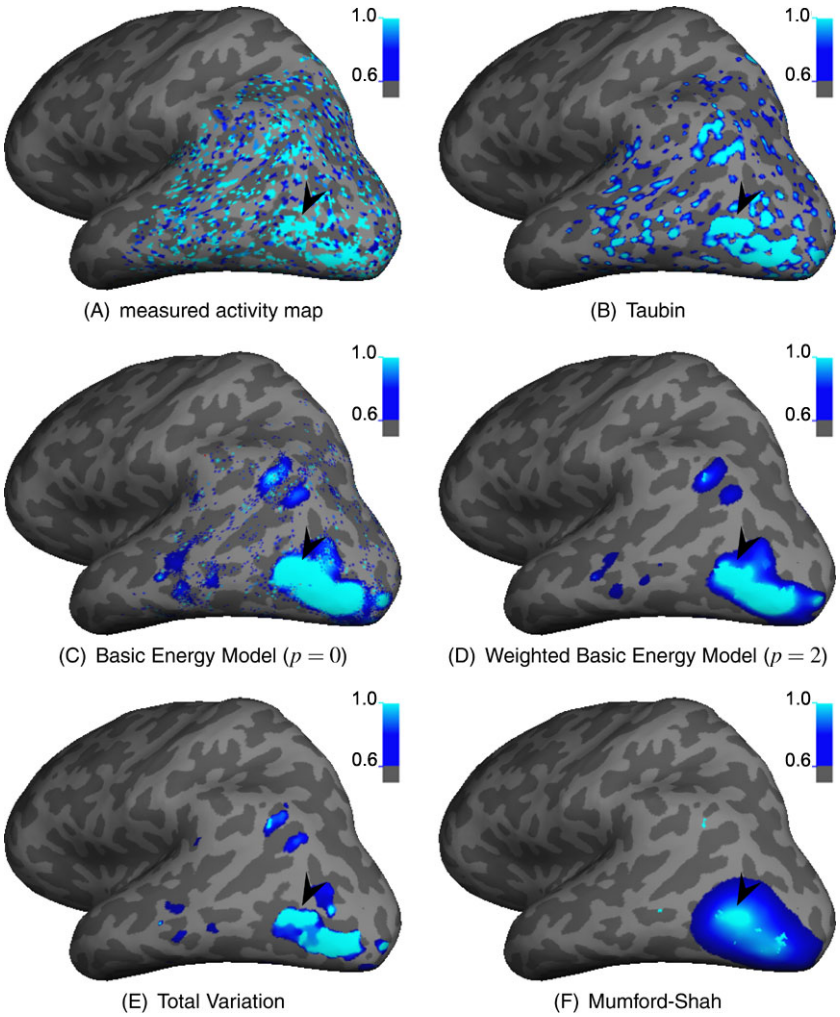

Fig. 5.11 Effect of smoothing methods on sub-threshold fMRI activity. The data of Fig. 5.10 is re-plotted with a color scale that highlights the relative performance and behavior of the filtering methods on activity below the significance threshold. *Reference arrows* are positioned as in Fig. 5.10

the color plate section at the end of the book) but with a lower statistical threshold to examine the sub-threshold patterns of the filtered data. With this color scale, the degree of noise suppression outside of the area of activity is more clear, with the results of the weighted Basic Energy Model shown in Fig. 5.11(D), Total Variation Fig. 5.11(E), and the Mumford–Shah algorithm shown in Fig. 5.11(F) performing best. Additionally, a salient and undesirable feature of the Mumford–Shah algorithm is that it spreads the activity pattern diffusely, losing most of the structure of the original data in this case.

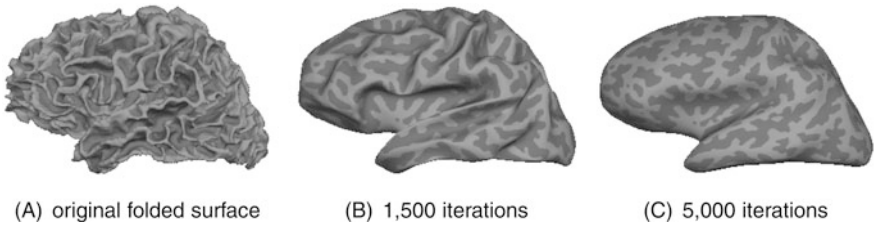

Fig. 5.12 Surface smoothing for cortical inflation. (**A**) Original folded surface. (**B**) Surface after 1,500 iterations of spatial filtering using the Basic Energy Model with $p = 2$. (**C**) Surface after 5,000 iterations of filtering. The high-frequency folds are removed with smoothing, leaving an "inflated" surface in which the regions within the cortical sulci are clearly visible, similar to the explicitly inflated surface presented in Fig. 5.10(C)

In this example, surface data are visualized on the inflated surface representation, which is very common in fMRI studies. Although several tools exist for rapidly computing inflated brain surface, it is instructive to note that the same smoothing operations used to filter the data along the surface can be applied to *filtering the surface mesh vertex coordinates themselves*—as in the previous example on mesh fairing—to smooth out the folding pattern and produce an "inflated" version of the cortical surface representation. Figure 5.12 demonstrates an example of how iterative smoothing using the Basic Energy Model with $p = 2$ can produce an inflated surface representation.

5.9.4 Geospatial Data

A different type of application involving data analysis at discrete locations comes from a parcellation of continuous space into subregions in which measurements are made. Many types of geospatial data fit this description in which a geographical area is parcellated into regions that fit a political, topographic or property description. Examples of this type of data would be soil samples, transportation data, pollution measurements, incidence of infectious disease, or population of a species. Geospatial data is typically managed, analyzed and visualized by software known as a *geographical information system* and this data is typically analyzed with a set of tools known as **spatial statistics** [82, 317].

In this section, we adopt an example of state polling data from the 2008 US Presidential election for the 48 continental states. Each US state is assigned a number equal to the percentage of poll respondents who favored (then candidate) Barack Obama just prior to the 2008 election. Each state is colored brighter if more respondents favored Mr. Obama and darker if fewer respondents favored Mr. Obama. Due to small samples and different polling methodologies, we can assume that there is noise present in this data. A simple model for filtering this polling data would be to assume that a state is more likely to favor a candidate if its neighboring states favor a candidate and less likely to favor a candidate if its neighboring states do not

favor the candidate. This model is justified by the concept of **Tobler's Law** or **the First Law of Geography** which asserts that data situated at nearby geographical locations are likely to be correlated [377]. Given this correlation, we may apply a spatial filtering algorithm to the polling data to remove noise. In this example, each state is represented by a node, two states are connected by an (unweighted) edge if they share a border, the measured data $\mathbf{s}$ is the polling data and the goal is to produce filtered polling measurements $\mathbf{x}$. We stress that although the underlying domain (the continental United States) is continuous, the parcellation of this space into political states for which polling measurements are made transforms this problem into the current framework of analyzing data associated with a graph.

Figure 5.13 displays the original polling data and the filtered data. Taubin's filtering method and the weighted Basic Energy Model with $p = 2$ yield similar results. The output of both of these filtering methods is largely unchanged from the original polling data, except that spatial outliers are softened. For example, the weak poll numbers for Mr. Obama in South Carolina and Indiana were improved after filtering because the poll numbers for Mr. Obama in neighboring states were generally stronger. Similarly, the polling numbers in New Hampshire and Maine were balanced after filtering, with the filtered polling numbers showing stronger support for Mr. Obama in New Hampshire and weaker support for him in Maine. The Mumford–Shah filtering produces substantially different results in this application. Specifically, this approach attempts to identify regions within which the polling numbers are expected to be relatively homogeneous. Therefore, the mid-Atlantic states are grouped with most of New England by this algorithm to produce one large voting bloc, the southeastern states are grouped with areas of the midwest to create a second voting bloc, a third voting bloc is produced by the great lakes states and upper midwest and a final voting bloc is produced by the west coast and the southwest. Florida and Maine-New Hampshire are also considered by the Mumford–Shah filtering approach as independent voting blocs. Within these voting blocs, the polling numbers are made more homogeneous (similar to the noisy circle image processing example). Although these voting blocs roughly correspond with meaningful political battle lines in the US in 2008, other parameter settings for this algorithm might produce larger or smaller voting blocs within which the data would be smoothed.

5.9.5 Filtering Flow Data—Brain Connectivity

In this section, we give an example of filtering real flow data along edges. Examples of flow data encountered in practical applications would be traffic networks, communication networks, or migration networks. In fact, if connection strengths between nodes are provided for any directed graph, these strengths could be considered as flow data. When considering flow data, a directed graph may be viewed simply as an undirected graph for which the edge directions represent the directions in which flow is considered to be positive.

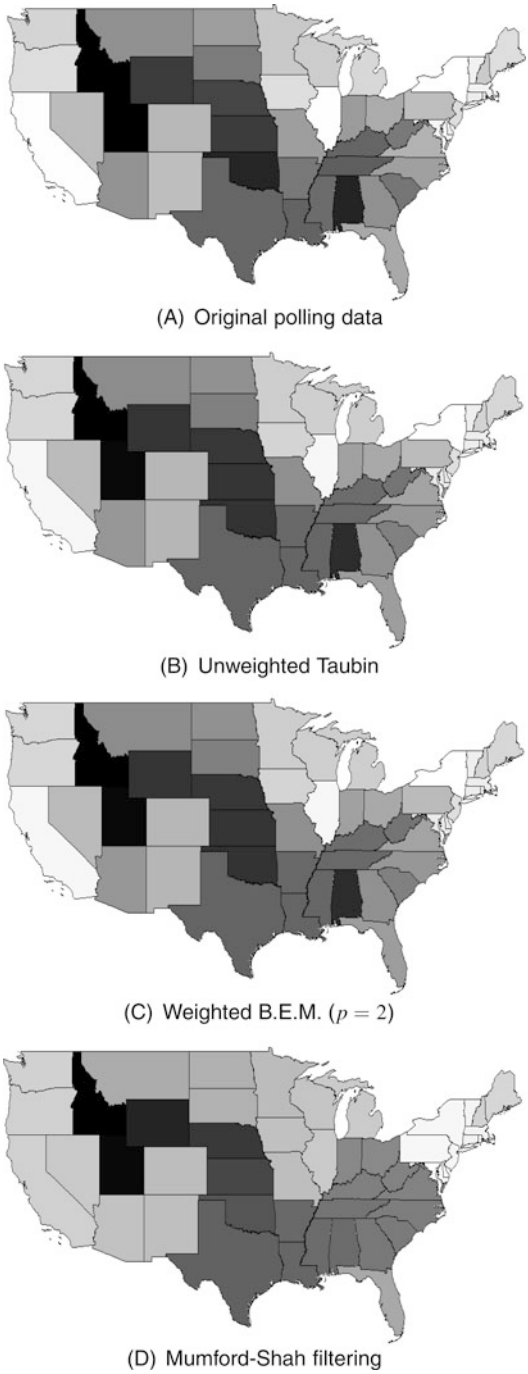

Fig. 5.13 Filtering of polling data for the 2008 US Presidential election. Brighter coloring indicates stronger support for (then candidate) Mr. Obama and darker coloring indicates weaker support for Mr. Obama. Each state is represented by a node in the graph where two nodes share an (unweighted) edge if they share a border

We address the filtering of the measured strength of brain connectivities between parcellated regions of the cat brain measured by Scannell et al. [329] (and subsequently studied by others [354]). This connectivity network consists of 52 brain regions (nodes) and 818 directed edges. Scannell et al. assigned each edge a connection strength of '1' (weak), '2' (medium) or '3' (strong), which was determined from a compilation of measurements from the cortico-thalamic system of the cat. We may assume that noise was present in these measurements as a result of imprecise measuring devices and as a result of the severe quantization of the data into three categories. The justification for removing noise with the filtering techniques described above also applies to the filtering of flow data. Specifically, in the scalar case, data points within a low-frequency scalar (node) distribution are similar between neighboring nodes. Similarly, data within a low-frequency flow (edge) distribution are similar between neighboring edges in which neighboring edges are either incident on the same node (where similarity means small divergence) or neighboring edges are incident on the same cycle (where similarity means small circulation). Above in Fig. 5.3 we saw that a flow field through a continuous domain straightens out after filtering. Additionally, the noise model justifications for these filtering procedures also applies to the flow case—a zero-mean noise flow distribution will be expected to have zero divergence at all nodes and zero circulation around all cycles.

Several liberties were taken with this data in order to make the filtering operations clearer. First, a random subset of the graph nodes were sampled for presentation purposes to better visualize the results of the filtering. Second, we arbitrarily removed one edge from every pair of nodes connected by two directed edges in the opposite directions. This removal was also made to improve visualization of the results. Figure 5.14 displays the results of our filtering operation on the flow data. The first two figures show the full network and the connections in the subnetwork. The next figures illustrate the measured flow strength (represented by line thickness) and the lowest-frequency eigenvector of the edge Laplacian. The lowest frequency component of the network distributes the flows equally across edges in order to minimize flow divergence at nodes and to minimize flow circulation (curl) around cycles. Note that the eigenvector is *signed*, which is indicated by a change in direction (arrow) for negative flows. Taubin's spectral filtering and the Basic Energy Model ($p = 2$, corresponding to diffusion or generalized mean filtering) were applied to filter the initial flow data. The results of these filtering operations reduce the divergence and circulation of the original measured flow while driving the filtered flow toward the low-frequency eigenvector. The filtering especially dampened the edges contributing to the high-divergence of the leftmost node in the diagram, as well as dampening the edges causing divergence on the central node. In addition to reducing noise, the filtering operation also produced real-valued flows from the initial quantized flows which may allow for a better comparison between edge connectivities.

Since the underlying network is directed, care must be taken when applying these filtering operations that none of the flow signs change, causing direction changes. In this example, this constraint was enforced simply by using a small number of filtering iterations so as to not oversmooth the data.

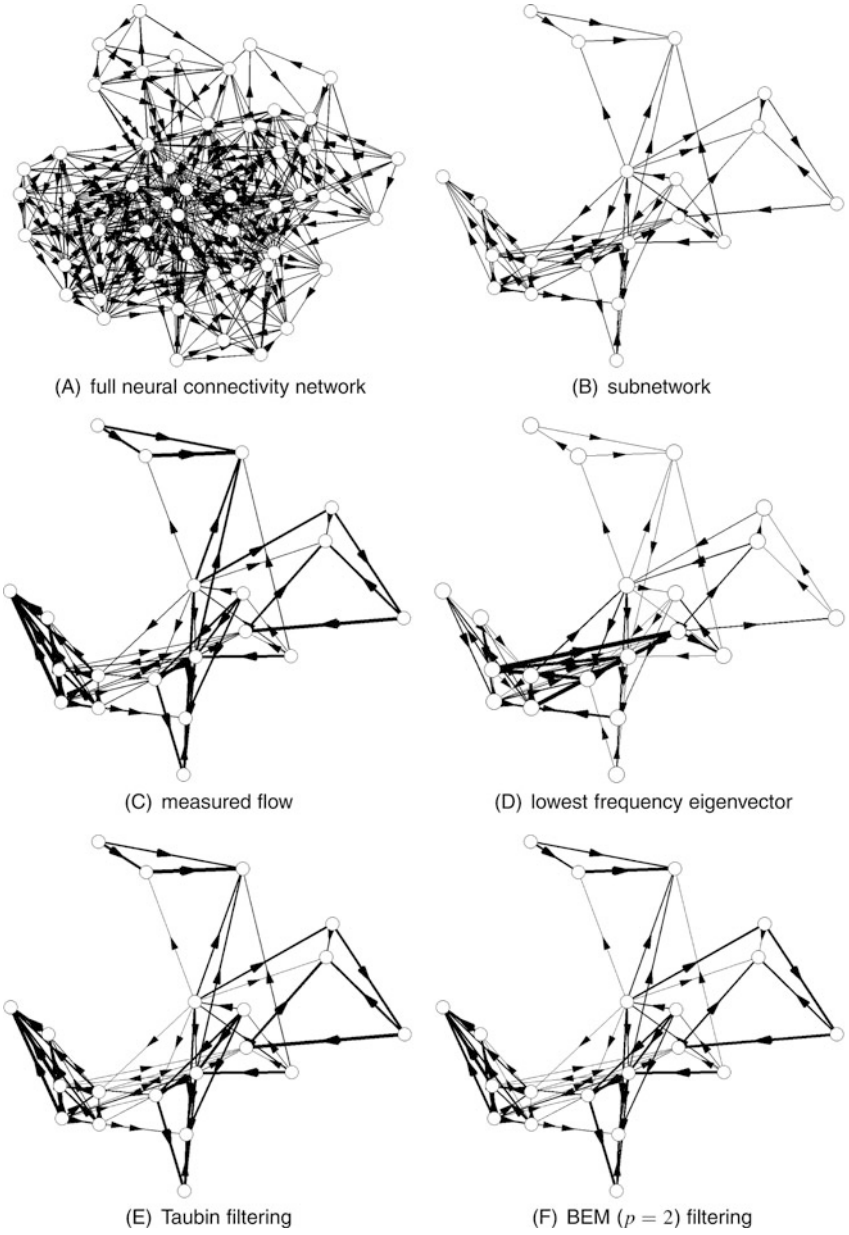

Fig. 5.14 Example of flow filtering on a network of brain connectivity data collected in the cortico-thalamic system of the cat [329]. A subnetwork of the original data was instead processed for visualization purposes. Line thickness represents the measured strength of the connections (the original "flow" data and filtered data). This example illustrates that the lowest frequency component of the network distributes the flows equally across edges in order to minimize flow divergence at nodes and to minimize flow circulation (curl) around cycles. Lowpass Taubin or diffusion ("mean") filtering drive the flows closer toward the flow given by low frequency eigenvector

5.10 Conclusion

In this chapter we reviewed several broad approaches for filtering data defined on arbitrary graphs—even irregular graphs. Although several examples were given, a reader with a particular filtering problem may still be wondering which method to use. The most classic filtering approach is Fourier-based filtering, but such an approach does not permit the preservation of discontinuities, nor does it apply to data defined on irregular (or weighted) graphs. In a more general setting, by far the most common methods are mean filtering or nonlinear anisotropic diffusion since these methods are straightforward to implement, predictable, and run in low-constant linear time. However, mean filtering has a tendency to oversmooth (even with discontinuities permitted), and to drive the data to a single value (the "shrinking" problem in mesh filtering). The filter described by Taubin is, in our opinion, an underutilized method (outside of computer graphics) which solves the second problem with little additional overhead or coding complexity (although it does require the specification of an additional parameter). If more computation is tolerable to provide a better result, then the variational approaches described in this chapter (e.g., median filtering with discontinuities, total variation filtering) are not difficult to implement and produce results that are not oversmoothed, but they do require more computation. Data which fits the assumptions of the Mumford–Shah model—that the data belongs to multiple regions which have smooth internal data—may be filtered well by minimizing the Mumford–Shah energy. If even better results are required, the underlying graph is shift-invariant and computation time is less important, then the variational filters with nonlocal neighborhoods are likely to produce the best results known so far. Finally, the gradient manipulation methods are moderately computationally intense (e.g., requiring a sparse linear system solve), but they provide substantial flexibility for combating data corruption if a model of the expected gradients is known, such as the image processing example of the circle in which it was asserted that the gradients were either large or zero. In the next chapter, we show how these same filtering models may be applied to derive a variety of clustering algorithms.