J Fourier Anal Appl (2008) 14: 744–763
DOI 10.1007/s00041-008-9038-9

Three Novel Edge Detection Methods for Incomplete and Noisy Spectral Data

Eitan Tadmor · Jing Zou

Received: 21 June 2007 / Revised: 1 July 2008 / Published online: 16 September 2008

Abstract We propose three novel methods for recovering edges in piecewise smooth functions from their possibly incomplete and noisy spectral information. The proposed methods utilize three different approaches: #1. The randomly-based sparse Inverse Fast Fourier Transform (sIFT); #2. The Total Variation-based (TV) compressed sensing; and #3. The modified zero crossing. The different approaches share a common feature: edges are identified through separation of scales. To this end, we advocate here the use of *concentration kernels* (Tadmor, Acta Numer. 16:305–378, 2007), to convert the global spectral data into an approximate jump function which is localized in the immediate neighborhoods of the edges. Building on these concentration kernels, we show that the sIFT method, the TV-based compressed sensing and the zero crossing yield effective edge detectors, where finitely many jump discontinuities are accurately recovered. One- and two-dimensional numerical results are presented.

Keywords Edge detection · Concentration factors · The RAℓSFA algorithm · Compressed sensing · Zero crossing · Incomplete data

Communicated by Ronald A. DeVore.

Research was supported in part by NSF grants DMS04-07704, DMS07-07949 and ONR grant N00014-91-J-1076.

E. Tadmor (✉)
Department of Mathematics, Institute for Physical Science and Technology and Center of Scientific Computation and Mathematical Modeling (CSCAMM), University of Maryland, College Park, MD 20742, USA
e-mail: tadmor@cscamm.umd.edu
url: http://www.cscamm.umd.edu/~tadmor

J. Zou
Center of Scientific Computation and Mathematical Modeling (CSCAMM), University of Maryland, College Park, MD 20742, USA
e-mail: jzou@cscamm.umd.edu
url: http://www.cscamm.umd.edu/~jzou

Mathematics Subject Classification (2000) 60G35 · 65T50

1 Introduction

We are concerned here with edge detection in piecewise smooth functions. Let f be such a piecewise smooth function with discontinuous jumps, $[f](\xi_j)\neq 0$, at finitely many locations $\xi_1,\dots,\xi_B$; here, $[f](x):=f(x+)-f(x-)$ denotes the jump function with the right and left side limits $f(x\pm)$. The data is given to us in terms of its spectral content, $\{\widehat{f_k}\}_{\{|k|\le N\}}$, and the task is to recover the edges which are sought in the physical space. Thus, given the $\widehat{f_k}$'s, one is interested in computing the locations ξ_j's and amplitudes of the jumps, $[f](\xi_j)$, $j=1,\dots,B$.

Our starting point is the *conjugate coefficients*, $\widetilde{f}=(\widetilde{f}_{-N},\dots,\widetilde{f_N})$, given by

$$\widetilde{f_k}:=\pi i\,\mathrm{sgn}(k)\sigma\left(\frac{|k|}{N}\right)\widehat{f_k}. \tag{1.1a}$$

Here, $\sigma(\cdot)$ is a properly normalized but otherwise *arbitrary* function at our disposal,

$$\int_0^1\frac{\sigma(\theta)}{\theta}d\theta=1. \tag{1.1b}$$

Given the $\widetilde{f_k}$'s, one computes the corresponding *concentration kernel*

$$K_N^\sigma[f](x):=\sum_{|k|\le N}\widetilde{f_k}e^{ikx}. \tag{1.1c}$$

In [17, Theorem 4.1] and its refinement in [9, Theorem 2.3], it was shown that

$$K_N^\sigma[f](x)=[f](x)+\begin{cases}\mathcal{O}(\frac{\log(N)}{N}), & d(x)\lesssim\frac{\log(N)}{N},\\ \mathcal{O}(\frac{\log N}{(Nd(x))^s}), & d(x)\gg\frac{1}{N}.\end{cases} \tag{1.2}$$

Here, $d(x)$ denotes the distance between x and the nearest jump discontinuity, $d(x)=\min_i|x-\xi_i|$, and $s=s_\sigma>0$ depends on our choice of σ in (1.1a). Equation (1.2) tells us that $K_N^\sigma[f](x)$ tends to *concentrate* near jumps where $K_N^\sigma[f](x)=\mathcal{O}(1)$ for $x\approx\xi_j$, whereas $K_N^\sigma[f](x)=\mathcal{O}((Nd(x))^{-s})\approx 0$ away from the jumps where $d(x)\gg 1/N$. This implies that the concentration kernel $K_N^\sigma[f]$ can be used to detect edges by separation of scales.

Equipped with the conjugate coefficients $\widetilde{f_k}$, we develop in this paper three novel methods for edge detection. These methods are based on separation of scales but otherwise they provide more robust alternatives to the use of concentration kernels like (1.1c). An overview of the three methods is provided in Sect. 2 below; they include the following.

(1) The sparse (or super) Fast Fourier Transform (sFFT), see e.g., [19, 21]. The sFFT employs random sampling to extract large scales by separating the Fourier modes with large amplitudes from those with small amplitudes.

(2) The compressed sensing technique, e.g., [2, 3, 7, 8]. The compressed sensing identifies the ("relatively few") non-zero large amplitudes of the jump function, $[f](x)$, using ℓ_1 optimization to separate them from small amplitudes in the smooth regions where $[f](x) \approx 0$.
(3) The improved zero crossing algorithm, e.g., [15]. In this improved approach, we combine compressed sensing with zero crossing technique, inspired by the investigation of concentration kernels (1.1c) in [11, 12].

These methods offer several advantages over the straightforward use of concentration kernels (1.1): they can detect edges from *incomplete data*, they are very *robust to noise*, and the sFFT-based method takes only *sublinear time*. We now turn to elaborate on each of these three methods.

We begin with the sFFT method (also called RAℓSFA—Randomized Algorithm for Sparse Fourier Approximation, e.g., [19, 21]). The sFFT computes a (near-) optimal sparse Fourier representation of N-dimensional data in *sublinear* time; indeed, it uses *poly*$(\log N)$ spatial data to produce the Fourier representation with high success probability. This sublinear efficiency is achieved by a *randomized algorithm*, which takes only random samples to estimate necessary information. This approach was extended to process incomplete data in [20]. In our first novel method we import the sFFT approach to detect edges. To this end we need to overcome one discrepancy, namely, the sFFT produces a sparse discrete Fourier transform by processing information in physical space, whereas our edge detection is required to work in the other direction, processing the prescribed Fourier information $\{\widehat{f_k}\}$ in order to produce its corresponding sparse jump function in the physical space. In this context, we observe that the conjugate coefficients, $\{\widetilde{f_k}\}$, yield a well-localized approximation of the jump function which we are trying to recover,

$$\{\widehat{f}\}_{\{|k|\le N\}} \mapsto \{\widetilde{f}\}_{\{|k|\le N\}} \mapsto K_N^\sigma[f](x) = \sum_{|k|\le N} \widetilde{f_k} e^{ikx} \approx [f](x).$$

Our first edge detector is therefore based on an *inverse* sparse Fast Fourier Transform (abbreviated sIFT), which processes the information in spectral space, $\{\widetilde{f_k}\}_{\{|k|\le N\}}$, and detects the desired information of the sparsely located edges, $\{\xi_j : [f](\xi_j)\}_{j=1}^B$ in the physical space. The basic ingredients of our sIFT-based edge detectors are presented in Sect. 2.1, followed by a detailed description and accompanied by numerical results provided in Sect. 3. Our results illustrate that the proposed sIFT-based has a strong denoising feature.

Next, we turn our attention to the compressed sensing technique, e.g., [2, 3, 7, 8, 16]. The focus is on the recovery of sparse data in physical space from an *incomplete* spectral information, $\{\widehat{f_k}\}_{\{k\in\Omega\}}$, prescribed at the subset $\Omega \subset \{k : |k| \le N\}$. The recovery is achieved by a Total Variation-based (TV) optimization. As before, the task is to recover $K_N^\sigma(x)$ as a well localized approximation to the jump function, $[f](x)$. The novelty here is the use of the corresponding incomplete set of conjugate coefficients, $\{\widetilde{f_k}\}_{\{k\in\Omega\}}$, as an input for the TV-based compressed sensing model, which in turn yields a sparse realization of the concentration kernel,

$$\{\widehat{f}\}_{\{k\in\Omega\}} \mapsto \{\widetilde{f}\}_{\{k\in\Omega\}} \mapsto \overbrace{\sum_{k\in\Omega}\widetilde{f}_k e^{ikx}}^{\text{prescribed data}} + \overbrace{\sum_{k\notin\Omega}\widetilde{f}_k e^{ikx}}^{\text{recovered data}} \approx K_N^\sigma[f](x).$$

The TV-based approach for spectral edge detection is outlined in Sect. 2.2, followed by a detailed description and accompanied by numerical results in Sect. 4.

So far, our aim in the construction of the sIFT- and TV-based edge detectors was to separate scales, so that edges are identified as isolated extrema of the concentration kernel $K_N^\sigma(x)$. It remains to actually trace these extrema. This brings us to the third and final method of edge detection based on the zero-crossing approach, which is one of the more popular edge detectors by practitioners, e.g., [15]. Here, we combine the construction of a concentration kernel, together with tracing its extrema which are sought as the zero level-set of, say, the corresponding discrete Laplacian. Indeed, zero crossing is intimately connected with the concentration kernels: if we set $\sigma(\theta)=\theta$, then the corresponding concentration kernel in (1.1a) amounts to the derivative of the usual partial Fourier sum,

$$K_N^\sigma[f]=\frac{\pi}{N}\big(S_N[f](x)\big)_x, \quad S_N[f](x):=\sum_{|k|\le N}\widehat{f}_k e^{ikx}.$$

Thus, edges—which are sought as extrema points of $K_N^\sigma[f]$, can be identified as the zeros of $(K_N^\sigma[f])_x$, or what amounts to the same thing—the zeros of $(S_N[f])_{xx}$. Similarly, in two-space dimensions one is led to zero crossing of the Laplacian, $\Delta_x S_N[f]\approx 0$. In fact, by a proper choice of σ, one obtains a *larger* class of "regularized" zero crossing. Of course, seeking the zero crossing of the Laplacian may introduce spurious, redundant edges, beyond those "true" edges which are extrema values of $\nabla_x S_N[f](x)$. We therefore propose to post-process the zero-crossing in order to remove this redundant, non-extrema zero-crossing. The resulting method is further extended to deal with incomplete data by incorporating the compressed sensing approach. Our improved zero-crossing edge detection method is outlined in Sect. 2.3, followed by a detailed description and numerical results provided in Sect. 5.

2 An Overview of the Proposed Edge Detection Methods

The purpose of this section is to provide an overview of the three edge detectors proposed in this paper: the super Fast Fourier Transform (sFFT), the TV-based compressed sensing technique and the improved zero crossing algorithm. A detailed description of each method is outlined in the respective Sects. 3, 4 and 5.

2.1 Edge Detection by Sparse Inverse Fourier Transform (sIFT)

Consider the jump function associated with $f(\cdot)$

$$[f](x):=\sum_{j=1}^{B}[f](\xi_j)1_{\xi_j}(x), \tag{2.1a}$$

where 1_{ξ_j} is the indicator function supported at the discrete interval containing ξ_j,

$$1_{\xi_j}(x) = \begin{cases} 1, & \text{if } x \in I_{\xi_j}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.1b}$$

Remark 2.1 In practice, the function f is realized by its *discrete gridvalues*, $\{f(x_\nu)\}_\nu$, although our terminology throughout the paper does not distinguish between the continuous and discrete versions of f. In this context, I_{ξ_j} should be interpreted as the unique interval enclosing the jump discontinuity

$$I_{\xi_j} := [x_{\nu_j}, x_{\nu_j+1}) \quad \text{such that } \xi_j \in [x_{\nu_j}, x_{\nu_j+1}). \tag{2.1c}$$

The exact jump function, $[f](x)$, is not available to us. Instead, we use the available spectral data of f to form the concentration kernel, $K_N^\sigma[f](x)$, as an *approximate jump function*, which returns the same sparsity of $[f](x)$. Viewed as an approximate jump function, $K_N^\sigma[f](x)$ takes the form

$$K_N^\sigma[f](x) = \sum_{j=1}^{B} [f](\xi_j)\Lambda_{\xi_j}(x), \tag{2.2}$$

where $\Lambda_{\xi_j}(x) = \mathcal{O}((Nd(x))^{-s})$ are the *approximate* indicators localized around ξ_j, consult (1.2),

$$\Lambda_{\xi_j}(x) \approx \begin{cases} 1, & \text{if } |x - \xi_j| \lesssim \frac{\log N}{N}, \\ 0, & \text{if } |x - \xi_j| \gg \frac{1}{N}. \end{cases}$$

Here is our main point: although we do not have the representation of $K_N^\sigma[f]$ in physical space, we have its Fourier representation in terms of the conjugate coefficients in (1.1a),

$$K_N^\sigma[f](x) = \sum_{|k|\le N} \widetilde{f}_k e^{ikx}, \qquad \widetilde{f}_k = \pi i \,\mathrm{sgn}(k)\sigma\left(\frac{|k|}{N}\right)\widehat{f}_k. \tag{2.3}$$

Starting from the $\widetilde{f}_k$'s, the evaluation of (2.3) can be carried either by a direct summation with $\mathcal{O}(N^2)$ operations, or using the $\mathcal{O}(N \log N)$ Fast Fourier Transform (FFT). In contrast, we advocate here the use of the sFFT approach for edge detection in $\mathcal{O}(\log N)$ operations. To this end we utilize the fact that the “signal” to be recovered—the approximate jump function $K_N^\sigma(x)$, is sparse

The major discrepancy for using the sFFT for edge detection is that they work on different domains: edge detection works on spectral data $\widetilde{f}_k$, while the sFFT processes physical data. To overcome this discrepancy, we employ the usual duality between physical and spectral space: one can easily obtain physical information from spectral data by the inverse Fourier Transform. Consequently, the key is to develop an *sparse Inverse Fast Fourier Transform* (sIFT) with the conjugate coefficients $\widetilde{f}_k = \widehat{K_N^\sigma[f]}_k$ as its input.

To this end, we introduce the main idea of sIFT. Let

$$R[\widetilde{f}\,](x) = \sum_{j=1}^{b} z_j 1_{\xi_j}(x), \tag{2.4}$$

where $z_j = z_j(\widetilde{f}\,)$ and $\xi_j = \xi_j(\widetilde{f}\,)$ are computed iteratively as the approximate jumps and locations sought in (2.1a). The algorithm is essentially a greedy pursuit approach: it iteratively updates the function $R[\widetilde{f}\,], b = 0, 1, \dots, B$, by identifying additional approximate amplitudes z_j located at ξ_j, thus improving $R[\widetilde{f}\,]$ as an approximation to $K_N^\sigma[f]$ in (2.2). In each iteration, one uses the current $R[\widetilde{f}\,]$ to update the so-called residual vector, $\widetilde{f}_k - \widehat{R[\widetilde{f}\,]}_k$. The residual vector contains information about edges that have not yet been captured by $R[\widetilde{f}\,]$. At the heart of our procedure are the randomly based iterations to find edges of the residual vector (outlined in Sect. 3 below). This iterative procedure "enforces" the jumps z_j to be assigned to their precise location ξ_j, and it is repeated until the residual vector converges below a preassigned tolerance.

We should emphasize that the algorithm, as a randomized algorithm, employs only a small fraction of the data $\widehat{R[\widetilde{f}\,]}$. Hence, there is no need to compute $\widehat{R[\widetilde{f}\,]}_k$ and $\widetilde{f}_k$ for all k's. Instead, we compute $\widehat{R[\widetilde{f}\,]}_k$ and $\widetilde{f}_k$ for specific k's only when necessary. Hence, the computation of the $\widetilde{f}_k$'s is carried out throughout the sIFT procedures "on the fly". Moreover, due to sparsity, $b \le B \ll N$, one computes $\widehat{R[\widetilde{f}\,]}_k = \sum_{j=1}^{b} z_j e^{i\xi_j k}$, in only $\mathcal{O}(b)$ evaluations.

Algorithm 2.2 (Sparse Inverse FFT (sIFT) edge detection) The sIFT algorithm uses three sub-algorithms (outlined in Sects. 3 and 6 below): computing the approximate amplitude of the jump in Algorithm 3.1, the group testing Algorithm 7.1, and the energy estimation Algorithm 7.3.

(1) Input: signal $\widehat{f}_k$, an upper bound of the signal energy M, a ratio α for relative precision, success probability $1-\delta$, and accuracy factor ϵ.
(2) Initialize: set $R[\widetilde{f}\,] \equiv 0$ and $\widehat{R[\widetilde{f}\,]}_k = 0$ for all $-N \le k \le N$. Set the maximum number of iterations $T = \mathcal{O}(B\log(N)\log(1/\delta)/\epsilon^2)$.
(3) Test: use the energy estimation Sub-algorithm 7.3 to test whether $\|\widetilde{f} - \widehat{R[\widetilde{f}\,]}\|_{\ell_2}^2 \le \alpha \|R[\widetilde{f}\,]\|_{\ell_2}^2$. If yes, end Algorithm 2.2 with output $R[\widetilde{f}\,]$; else continue.
(4) Detect: use the group testing procedure in Sub-algorithm 7.1 to detect $x = \xi$ as an approximate location for an edge of the spectral data, $\widetilde{f} - \widehat{R[\widetilde{f}\,]}$.
(5) Estimate: use Sub-algorithm 3.1 to estimate the approximate amplitude of the edge at $x = \xi$, $z := \mathcal{F}^{-1}(\widetilde{f} - \widehat{R[\widetilde{f}\,]})(\xi) \approx [f](\xi)$.
(6) Update: add $z1_\xi(x)$ to $R[\widetilde{f}\,](x)$.
(7) Iterate: if the total number of iterations is less than T, go to #3; else end the algorithm with output $R[\widetilde{f}\,]$.

The resulting sIFT-based edge detection yields very accurate results as confirmed in Sect. 3.

2.2 Compressed Sensing-based Edge Detection for Incomplete Data

Assume that the spectral data is *incomplete*, that is, we have access only to $\widehat{f_k}$, $k \in \Omega$, where Ω is a strict subset of $\{-N, \dots, N\}$. The framework for edge detection in such cases where only partial information is available, is to combine the use of concentration kernels with the compressed sensing approach [2]. Equipped with the partial information of $\widehat{f_k}$ and hence, by (1.1a), of $\widetilde{f_k}$ in $k \in \Omega$, we aim at recovering an *approximate concentration kernel* $g(x) \approx K_N^\sigma[f](x)$, of the form

$$g(x) = \sum_{k \in \Omega} \widetilde{f_k} e^{ikx} + \sum_{k \notin \Omega} \widehat{g}_k e^{ikx}.$$

The $\widetilde{f_k}$'s are prescribed, while the free $\{\widehat{g}_k | k \notin \Omega\}$ at our disposal and are chosen by the total variation (TV) compressed sensing model, so that $\|g\|_{TV}$ is minimized,

$$\min_{\{\widehat{g}_k | k \notin \Omega\}} \left\{ \|g\|_{TV} := \sum_\nu |g(x_{\nu+1}) - g(x_\nu)| \mid g(x) := \sum_{k \in \Omega} \widetilde{f_k} e^{ikx} + \sum_{k \notin \Omega} \widehat{g}_k e^{ikx} \right\}. \tag{2.5a}$$

Here $x_\nu = \nu \Delta x$ are the equidistant sampling points of $g(x)$. Similar methodology will apply in the multidimensional case. Consider for example, the two-dimensional setup where we have access to partial set of Fourier modes $\widehat{f_k}$ with multi-index $k \in \Omega \subsetneq [-N, N]^2$. We set a rectangular grid $(x_\nu, y_\mu) = (\nu \Delta x, \mu \Delta y)$, and the missing $\widehat{g}_k$'s for $k = (k_1, k_2) \notin \Omega$ are sought to minimize the two-dimensional variation $\|g\|_{TV}$. Here, the two-dimensional TV can be expressed as,[1]

$$\|g\|_{TV} := \sum_{\nu,\mu} |g(x_{\nu+1}, y_\mu) - g(x_\nu, y_\mu)| \Delta x + \sum_{\nu,\mu} |g(x_\nu, y_{\mu+1}) - g(x_\nu, y_\mu)| \Delta y. \tag{2.5b}$$

Why compressed sensing? The TV minimization or the ℓ_1 minimization of the differences, imposes *sparsity* in the sense of maximizing the number of zero differences (the "ℓ_0" norm), e.g., [2–4, 7, 8]. It is in this sense that the framework of compressed sensing combined with the conjugate coefficients amounts to an effective edge detector, by imposing an approximate jump function $g(x)$ with a minimal number of piecewise constant components. The variational model (2.5) can be formulated as an optimization problem, which can be solved by the second order cone programs (SOCP's), with $\mathcal{O}(N^3 \log N)$ operations; in practical implementation, SOCP operates much faster, [4]. A detailed discussion on the proposed TV-based compressed sensing edge detection approach is found at Sect. 4.

We close this section by noting that in many situations, the (possibly incomplete) spectral data may be also contaminated by noise. We aim at both—to recover the missing data and to denoise the prescribed data. Inspired by [2], we then propose the modified TV-based edge detector

$$\min_{\{\widehat{g}_k\}} \{ \|g\|_{TV} \mid \|\widehat{g}_k - \widetilde{f_k}\|_{\ell^2(\Omega)} \le \beta \}. \tag{2.6}$$

[1] Other definitions of the two-dimensional total-variation can be used here, e.g., an ℓ_2 version which is equivalent with the ℓ_1 definition in (2.5b).

Here β is a tolerance measure of inaccuracies due to noise; choices for β are discussed in [3].

2.3 Zero Crossing Edge Detection for Incomplete Data

Zero crossing can be viewed as a particular case of edge detection based on properly tailored concentration kernel. To clarify this point, we begin with the one-dimensional example of concentration factor $\sigma(\theta) := \theta\widehat{\eta}(\theta)$, with a unit mass $\widehat{\eta}(\theta)$, where (1.1a)–(1.1c) amount to

$$K_N^\sigma[f](x) = \frac{\pi}{N}\sum_{|k|\le N} i\,\mathrm{sgn}(k)|k|\widehat{\eta}\left(\frac{|k|}{N}\right)\widehat{f}_k e^{ikx}, \qquad \int_{\theta=0}^{1}\widehat{\eta}(\theta)d\theta = 1.$$

Thus, if we let $\eta(x)$ denote the indicator function,

$$\eta(x) := \frac{1}{2N}\sum_{-N}^{N}\widehat{\eta}\left(\frac{|k|}{N}\right)e^{ikx}, \tag{2.7a}$$

then $K_N^\sigma[f](x)$ is nothing but a mollification of $S_N[f](x)$ with η_x, which can be viewed as an approximate *derivative* of the delta function,

$$K_N^\sigma[f](x) = \frac{\pi}{N}\left(\sum\widehat{\eta}\left(\frac{|k|}{N}\right)\widehat{f}_k e^{ikx}\right)_x = \eta_x * S_N[f](x). \tag{2.7b}$$

We note in passing that zero crossing may introduce *redundancy*: edges which were originally sought as extrema values of $K_N^\sigma[f](x)$, are now identified as zeros of $K_N^\sigma[f](x)_x$, i.e.,

$$\left\{\xi_j \;\middle|\; \frac{d}{dx}K_N^\sigma[f](x)|_{x=\xi_j} = \eta_{xx} * S_N[f](x)|_{x=\xi_j} = 0\right\}. \tag{2.8}$$

Consequently, zero crossing may identify inflection points as spurious edges which otherwise are ruled out as extrema values of the concentration kernel (1.1c).

We now turn to the two-dimensional case, where we consider the generic case of extreme curves. Extending the framework of one-dimensional concentration kernels, (2.7b)–(2.7a), we now seek extrema values of the *gradient* of $K_N^\sigma(x, y)$. Set $\sigma(\theta) = \theta\widehat{\eta}(\theta)$ where $\widehat{\eta}$ is a unit mass concentration factor at our disposal

$$\eta(x, y) := \frac{1}{(2N)^2}\sum_{|k_1|,|k_2|\le N}\widehat{\eta}\left(\frac{|k|}{N}\right)e^{ik_1x+k_2y}, \qquad \int_0^1\widehat{\eta}(\theta)d\theta = 1.$$

The edge detection associated with such concentration kernels, [13], is based on a straightforward Cartesian adaptation of the one-dimensional setup (2.7. In this case, we detect separated curves of discontinuities, and these curves are sought as *joint* extrema along the x- and y-axis of

$$\partial_x\left(\sum_{|k|\le N}\widehat{\eta}\left(\frac{|k|}{N}\right)\widehat{f}_k e^{ik_1x+ik_2y}\right) = \eta_x * S_N[f](x, y), \tag{2.9a}$$

$$\partial_y\left(\sum_{|k|\le N}\widehat{\eta}\left(\frac{|k|}{N}\right)\widehat{f_k}e^{ik_1x+ik_2y}\right)=\eta_y * S_N[f](x,y). \tag{2.9b}$$

One way to detect these extrema is to identify them as the zero crossing of the Laplacian of $\eta * S_N$, which yields a "traditional" zero crossing method,

$$\left\{(\xi_j,\zeta_k)\ \big|\ \Delta\eta * S_N[f](x,y)|_{(\xi_j,\zeta_k)}\equiv \Delta S_N^{\eta}|_{(\xi_j,\zeta_k)}=0\right\},\qquad S_N^{\eta}[f]:=\eta * S_N[f]. \tag{2.10}$$

The resulting method is even more redundant than in the one-dimensional case: the method adds a considerable amount of spurious edges, as observed in [6, 10]. Indeed, not all zero crossing of the regularized Laplacian, $\Delta S_N^{\eta} f$ are necessarily extrema of $\nabla S_N^{\eta}[f]$, which end up as "false" edges despite being zero crossings of (2.10).

Our Algorithm 2.3 outlined below will improve the "vanilla" zero-crossing corresponding to (2.10) with $\widehat{\eta}(\theta)\equiv 1$, in two ways: (i) eliminating spurious edges when viewed as extrema of the corresponding concentration kernel; and (ii) enabling a larger class of mollifiers η. Moreover, we can now extend the zero-crossing method to deal with incomplete data. Here we combine improved zero crossing with compressed sensing, seeking an approximate concentration kernel $g=\Delta\eta * S_N[f]$, such that

$$\min_{\{\widehat{g}_k\,|\,k\notin\Omega\}}\left\{\|g\|_{TV}\ \Big|\ g(x):=\sum_{|k|\le N}\widehat{g}_k e^{ikx}\ \text{s.t.}\ \widehat{g}_k=|k|^2\widehat{\eta}\left(\frac{|k|}{N}\right)\widehat{f_k}\ \text{for}\ k\in\Omega\right\}. \tag{2.11}$$

A typical choice of zero crossing mollifier η that we will be using below, is the normalized Gaussian function

$$\widehat{\eta}(\theta)=\frac{1}{\sqrt{2\pi}}e^{-\theta^2/2}. \tag{2.12}$$

Our improved zero-crossing edge detection method is summarized in the following algorithm. A detailed discussion of this algorithm is postponed to Sect. 5 below.

Algorithm 2.3 (Zero crossing edge detection for incomplete data)

(1) Input: the Fourier coefficients $\widehat{f_k}$ for $k\in\Omega$, and a threshold parameter, $\gamma>0$.
(2) Calculate: $\widehat{g}_k=|k|^2\widehat{\eta}(|k|/N)\cdot\widehat{f_k}$ for $k\in\Omega$.
(3) Solve the TV-based compressed sensing model (2.11) for $g(x)=\sum_{|k|\le N}\widehat{g}_k e^{ikx}$.
(4) Identify possible edges as zero crossing points of g, such that $g(x_\nu,y_\mu)=0$.
(5) Post-process redundancy: scan the zero-crossing points. If either $|g_x(x_\nu,y_\mu)|\neq 0$ or $|g_y(x_\nu,y_\mu)|\neq 0$, and at least one of the immediate neighbors of (x_ν,y_μ) satisfies $|g(x,y)|>\gamma$, accept (x_ν,y_μ) as an edge point.

3 Details: The Sparse Inverse Fourier Transform (sIFT) Edge Detector

This section presents the details involved in the sIFT-edge detection method. Specifically, we discuss how to find edge locations in Sect. 3.1 and approximate jump amplitudes in Sect. 3.2. Numerical results are demonstrated in Sect. 3.3.

3.1 How to Locate Edges

Step 4 in Algorithm 2.2 is the key procedure of the sIFT-based edge detection: it locates edges. The procedure is a dual version of the sFFT procedure in [19], where the input of conjugate coefficients and inverse Fourier basis functions replace Fourier basis functions. Accordingly, we list the various algorithms involved in the appendix and we limit ourselves here to a few clarifications of the overall sIFT-based edge detector.

Assume, for simplicity, that f has only one dominant edge. A recursive procedure, called group testing and outlined in Algorithm 7.1 below, is repeatedly used to reduce the interval sought to contain the dominant jump discontinuity: it splits into two halves the current interval, compares signal energies in the two half-intervals and keeps the sub-interval with the larger energy. The same divide-and-conquer procedure is repeated for the remaining tree of smaller half intervals, so that we end up with a final candidate interval that contains the dominant discontinuity. In doing so, the group testing algorithm uses two other sub-algorithms: the choice of an interval sought to contain a significant jump is made by Algorithm 7.2, locating the jump by its Most Significant Bit (MSB), one bit at a time. The MSB algorithm finds the significant bits by estimating the energy in each suspected interval, using Algorithm 7.3.

The extension to general piecewise smooth f's which contain several jumps proceeds as follows. We first construct a new signal F with only one dominant jump, which is then processed by the group testing in Algorithm 7.1. To generate such a signal, we convolve the original f with the randomly permuted box-car pass filter,

$$\widehat{H}(k) = \chi_{[-q_1,q_1]}(\tau k)e^{2i\pi k\theta/N} * \widehat{S}_k; \tag{3.1}$$

here $2q_1 + 1$ is the filter width and τ and θ are random dilation and modulation factors. The decay behavior of the Box-car filter preserves the energies at the center pass region and reduces the influence of other regions. This plays a key role in generating a new function F with one dominant jump. Moreover, to avoid jump discontinuities which might be clustered in the same pass region, we permute the signal randomly to separate neighboring jumps. Although the random permutations may fail to yield a new signal with just one dominant jump, we nevertheless continue working with the procedures of group testing and energy estimation, and we accept the candidate jump if its amplitude is estimated to be large enough.

3.2 How to Estimate the Amplitude of Edges

Given the location of an edge $x = \xi$, this subsection describes how we estimate the amplitude of the jump there. This corresponds to step 5 in Algorithm 2.2. There are two different approaches.

In the first approach, one computes the amplitude of the approximate jump function (1.1c). It has the advantage of computing the amplitude accurately, yet it introduces a linear, $\mathcal{O}(N)$ computational cost. Therefore, this approach is desirable in the situations where the accuracy is more important than the speed.

The second approach is essentially a Monte Carlo integration. It follows the ideas of coefficient estimation in the original sFFT. The method produces a good approximate value of $K_N^\sigma[f](x)$ by taking means and medians of $\widetilde{f}_k e^{ikx}$ for randomly generated indices k, i.e.,

$$K_N^\sigma[f](x) = \sum_{-N}^{N} \widetilde{f}_k e^{ikx} \approx \operatorname*{median}_{j=1,\dots,2\log(1/\delta)} \left(\operatorname*{mean}_{l=1,\dots,8/\epsilon^2} (\widetilde{f}_{k_{j,l}} e^{ik_{j,l}x}) \right).$$

Here, we use only a small fraction of the data and hence speed up the estimation process, at the expense of losing accuracy.

Following our discussion in Sect. 2.1, we shall apply appropriate "duality" changes to the original coefficient estimation in [19]. Thus, we replace the inverse Fourier basis by Fourier basis, and sampling of the spectral data generated by (1.1). The resulting algorithm reads as follows.

Algorithm 3.1 (Estimate the amplitude of jump discontinuity)

Input: the spectral data $\widehat{f}_k$, the candidate ξ for the jump location, accuracy factor ϵ, success probability $1-\delta$.

(1) Generate: uniformly random indices $k_{j,l} \in \{-N, \dots, N\}$ and derive $\widetilde{f}_{k_{j,l}}$: $\widetilde{f}_{k_{j,l}} = \pi i \operatorname{sgn}(k_{j,l}) \sigma(\frac{|k_{j,l}|}{N}) \widehat{f}_{k_{j,l}}$, where $j = 1, \dots, 2\log(1/\delta)$, $l = 1, \dots, 8/\epsilon^2$.
(2) Compute: for a fixed $j = 1, \dots, 2\log(1/\delta)$, take the empirical mean:

$$\operatorname{mean}(j) = \frac{\epsilon^2}{8} \sum_{l=1}^{8/\epsilon^2} \widetilde{f}_{k_{j,l}} e^{ik_{j,l}\xi}.$$

(3) Compute: Take the median $z = \operatorname{median}_{j=1,\dots,2\log(1/\delta)}(\operatorname{mean}(j))$.
(4) Output: z as the approximation of the amplitude of the edge $[f](\xi)$.

The following lemma guarantees that with high probability, the above algorithm for estimating the jump amplitude, is within $\mathcal{O}(\epsilon)$ of the energy of $K_N^\sigma[f]$.

Lemma 3.2 *Every application of Algorithm 3.1 generates a realization $R[f]$ of a random variable z, which estimates the jump amplitude, $K_N^\sigma[f]$, up to a tolerance of order $\epsilon^2 \|K_N^\sigma[f]\|_2^2$ with high probability $\geq 1-\delta$, i.e.,*

$$\operatorname{Prob}\left(|z - K_N^\sigma[f](\xi)|^2 \geq \epsilon^2 \|K_N^\sigma[f]\|_2^2 \right) \leq \delta. \tag{3.2}$$

The proof is similar to [19, Lemma 3.4], where S and e^{-ikx} are replaced by $\widetilde{f}$ and e^{ikx} respectively.

As observed in [19], fewer samples than the theoretical requirement already yield an accurate estimate for the amplitude of the discontinuity with high probability. For instances, 150 samples are enough to determine one coefficient with accuracy 10^{-4}. This means that when the fast speed is desirable and the number of data is huge, the Monte Carlo integration for amplitude estimation computes only a small fraction of the data and can thus be very efficient.

3.3 Numerical Results for sIFT Edge Detection

To demonstrate the performance of the sFFT-based method, we compare it with the edge detector based on the minmod limiter employed in [14, Sect. 4],

$$\operatorname{minmod}\big(K_N^{\exp}[f](x), K_N^{\sigma_3}[f](x)\big); \tag{3.3a}$$

here exp is the highly localized exponential advocated in [12, (2.23)], and [13, (3.21)], $\sigma_3(\theta) = 3\theta^3$, and minmod stands for the usual limiter,

$$\operatorname{minmod}(a, b) := \begin{cases} s \cdot \min(|a|, |b|), & \text{if } s = \operatorname{sgn}(a) = \operatorname{sgn}(b), \\ 0, & \text{if } \operatorname{sgn}(a) \neq \operatorname{sgn}(b). \end{cases} \tag{3.3b}$$

The first set of numerical experiments involves the function $f_a(x)$,

$$f_a(x) = \begin{cases} (\frac{x+\pi}{\pi})^5 & \text{if } x < 0, \\ (\frac{x-\pi}{\pi})^5 & \text{if } x > 0, \end{cases} \tag{3.4}$$

with the corresponding jump function

$$[f_a](x) := \begin{cases} -2 & \text{if } x = 0, \\ 0 & \text{if } x \neq 0. \end{cases} \tag{3.5}$$

Figure 1 shows numerical results of the sIFT edge detection technique. The sIFT method achieves more accurate results than the edge detector based on the minmod limiter (3.3). Specifically, it pinpoints the location of the jump and reduces oscillatory artifacts around it, an improvement over the minmod results.

Next, we carry out experiments for noisy data. We consider the function f_a contaminated by a white Gaussian noise with variance β, where the SNR—signal-to-noise ratio measured as $10\log_{10}(\|f_a\|_2^2/N\beta^2)$, equals to 10 dB. As a second example we

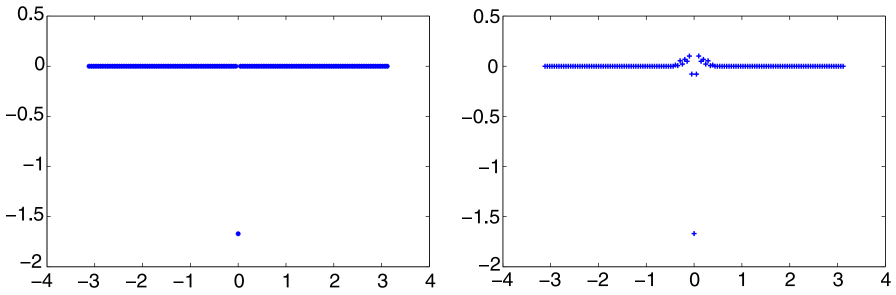

Fig. 1 The recovery of $[f_a](x)$ from its first $N = 128$ Fourier modes. The results of the sIFT edge detection method (*on the left*), and the results of minmod (3.3) *on the right*. Both experiments use $\sigma(\theta) = 3\theta^3$ corresponding to the unit mass concentration factor $\widehat{\eta}_3(\theta) = 3\theta^2$

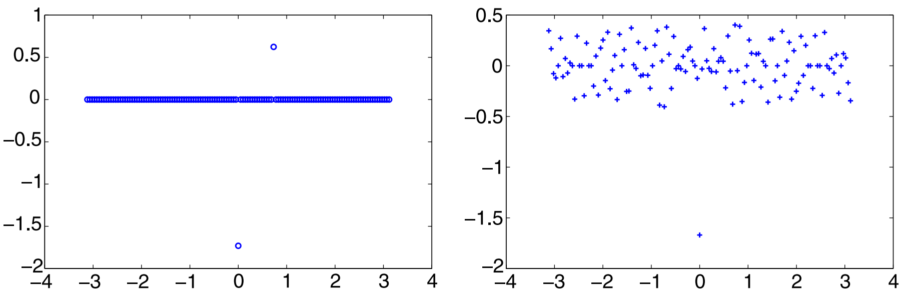

Fig. 2 Demonstration of the sIFT edge detection method for recovering from noisy $f_a(x)$. *Left*: the results of the sIFT method. *Right*: the results of minmod detector (3.3)

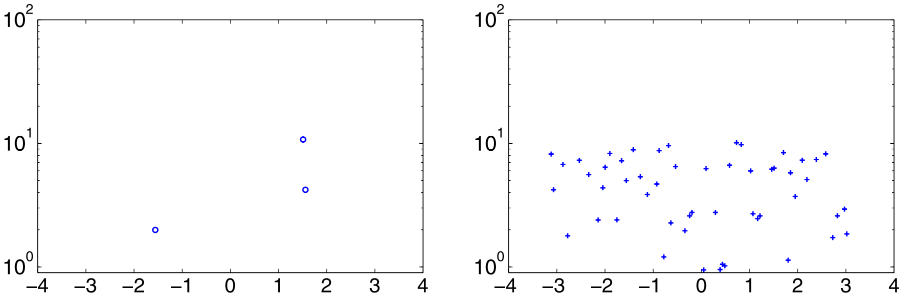

Fig. 3 Demonstration of the sIFT edge detection method for recovering from noisy $f_b(x)$. *Left*: the results of the sIFT method. *Right*: the results of minmod method (3.3), where the true edges are concealed in the noisy data

consider the function $f_b(x)$,

$$f_b(x) := \begin{cases} \sin(x+1)^7 & \text{if } x < -\frac{\pi}{2}, \\ (\frac{x}{\pi})^3 - \sin(\frac{9x}{2}) + 1 & \text{if } -\frac{\pi}{2} < x < \frac{\pi}{2}, \\ \sin(x-1)^7 & \text{if } x > \frac{\pi}{2}, \end{cases} \tag{3.6}$$

which is contaminated by noise of SNR 1.0 dB; the aim is to recover its jump function

$$[f_b](x) = \begin{cases} 0.582 & \text{if } x = -\frac{\pi}{2}, \\ -1.418 & \text{if } x = \frac{\pi}{2}, \\ 0 & \text{otherwise}. \end{cases} \tag{3.7}$$

Our results in Figs. 2 and 3 illustrate that the sIFT edge detection method yields a good approximate jump function despite the presence of noise, thanks to the strong denoising feature of the sIFT method. The results are compared against the minmod detector (3.3). It should be point out that the minmod-based edge detector was not designed to treat noisy data, which explains why the noisy results in Fig. 3 are far off: minmod-based detection was used here for comparison purposes, demonstrating the effect noise. Concentration factors which are adapted to noisy data can be found in the recent work [9].

4 Details: The TV-based Compressed Sensing Edge Detection

This section elaborates on the TV-based compressed sensing approach for edge detection which was outlined in Sect. 2.2. Recall that the goal here is to recover the approximate jump function $K_N^\sigma[f]$, based on the *incomplete* spectral data $\widehat{f_k}$, $k \in \Omega$. In particular, we will discuss the sparsity requirement in Sect. 4.1, limitations in Sect. 4.2 and numerical results are reported in Sect. 4.3.

Remark 4.1 When dealing with discrete data, there are two different approaches to generate the corresponding discrete conjugate coefficients $\widetilde{f_k}$. The straightforward approach is to compute $\widetilde{f_k}$ from the *discrete* Fourier coefficients $\widehat{f_k}$, which in turn are defined in terms of the discrete gridvalues, $f(x_\nu, y_\mu)$. In a second approach, one evaluates $\widetilde{f_k}$ *directly* in terms of *differences* of these discrete values. Indeed, first order differences correspond to the concentration kernel[2] $(K_N^\sigma[f])_x$ with $\sigma(\theta) = \theta$. In the TV-based compressed sensing detection reported below, we chose to use the latter approach: we computed the first order differences of the discrete data.

4.1 Sparsity

How many samples are needed to recover the approximate jump function? The ability to recovery a function from a few spectral coefficients depends on the sparsity of its jump function. The following result is straightforward adaptation of [2].

Lemma 4.2 *Suppose $K_N^\sigma[f]$ is a superposition of B spikes*

$$K_N^\sigma(x) = \sum_{j=1}^{B} z_j 1_{\xi_j}(x). \tag{4.1}$$

Choose a set Ω uniformly at random. There exists a constant $C_m \sim 1/m$ such that if

$$B \leq C_m \cdot (\log N)^{-1} \cdot |\Omega|, \tag{4.2}$$

then with probability of at least $1 - \mathcal{O}(N^{-m})$, the minimizer to the following problem

$$\min_{\{\widehat{g}_k | k \notin \Omega\}} \left\{ \|g\|_{TV} = \sum_\nu |g(x_{\nu+1}) - g(x_\nu)| \;\middle|\; g(x) = \sum_{k\in\Omega} \widetilde{f}_k e^{ikx} + \sum_{k\notin\Omega} \widehat{g}_k e^{ikx} \right\}, \tag{4.3}$$

recovers the exact jump function $K_N^\sigma(x)$.

We note that the concentration kernel K_N^σ is only an approximate superposition of "pure" B spikes alluded in (4.1). Accordingly, one expects an approximate recovery of K_N^σ using the stability of the TV-based compressed sensing in (4.3).

[2] In general, concentration factors $\sigma_k(\theta) \sim \theta^k$ with odd k's correspond to local differences in physical space, whereas even k's yields global discrete operators; we refer the reader to the detailed discussion in [13].

4.2 Limitations

Unlike the one-dimensional case, concentration kernels in $d \geq 2$ dimensions remain open to generalizations. In Sect. 2.2 we were tracing the extrema values of the *gradients* in (2.9), inspired by [13]. This is a straightforward adaptation of the one-dimensional framework based on a tensor product of concentration kernels in x- and y-dimensions. But it becomes a barrier for accurate recovery of edges from incomplete spectral data; for example, the use of tensor products yields staircase effects. One way to improve it is to take the largest component of the gradient in absolute value, so that when an edge point is missed in the horizontal direction, it may still be captured by the concentration kernel in the vertical direction. Still, the use of an "ultimate" two-dimensional edge detector lies on progress of developing high dimensional concentration kernels, beyond simple tensor products.

4.3 Numerical Results

We begin with the numerical investigation of TV-based compressed sensing edge detector in the context of Fig. 4—a prototype example of the Shepp-Logan phantom image, and Fig. 5. The main feature here is data which is collected by medical instruments, given in Fourier space along radial lines (Radon transform). There is a discrepancy, however, between the radial lines and the rectangular grid and the first task is therefore to transfer the radial spectral data into the Cartesian grid.

Given a phantom graph at $[0, \ldots, 256] \times [0, \ldots, 256]$ polar grid points. Since the samples along radial lines are not necessarily located exactly on the rectangular, one cannot use the spectral data directly without proper adjustment. To this end, we follow [2], and assign the gridvalue sampled on the polar grid into an appropriately chosen neighboring Cartesian grid point: it is chosen as the closest Cartesian gridpoint—either horizontally or vertically, according to the radial direction. The Cartesian neighboring gridpoint is determined by taking the same label in one direction and shifting a pixel in another direction. Specifically, assume there are N grid points in each direction and let ϕ denote the angle between the radial line and the horizontal axis. If either $\phi \leq \pi/4$, or $\phi \geq \frac{3\pi}{4}$, we take the Cartesian point with the same x-coordinate of the point on the radial line, $x = x_\nu$ and we set

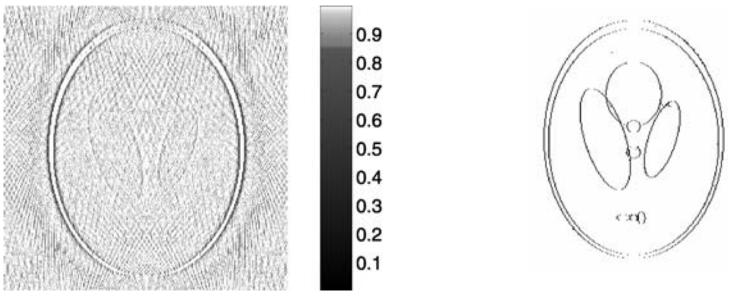

Fig. 4 The recovered phantom image from incomplete spectral data. *Left*: the result by the standard back projection. *Right*: the recovered edges of Shepp-Logan phantom graph by our compressed sensing edge detection method

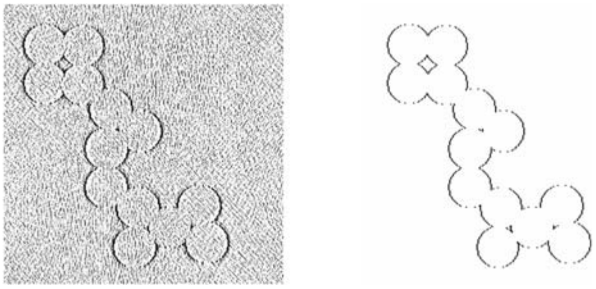

Fig. 5 The recovery results from incomplete data by compressed sensing edge detection method. *Left*: the result by the back projection method. *Right*: the recovered edges by the compressed sensing method. Here the original image is based on N^2 grid points, where $N = 256$. We take the spectral data of the first difference and gather all the samples along each of the 120 lines in the spectral domain

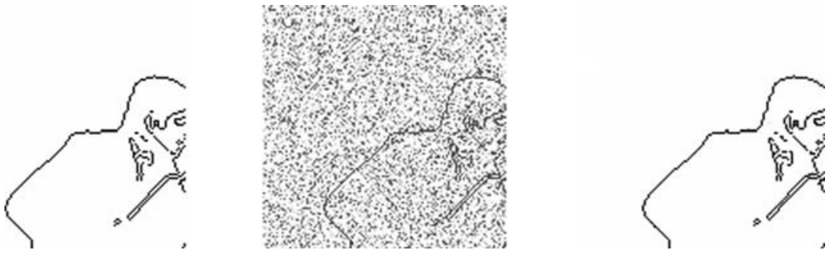

Fig. 6 The recovered images by the TV-based compressed sensing edge detector. The available data consists of 3000 Fourier coefficients which are uniformly and randomly distributed over the 128^2 grid points in the spectral domain. *Left*: the original cameraman image. *Center*: the image recovered from available data, with unobserved data set as zero. *Right*: the recovered edges by the compressed sensing edge detection method (2.5)

$y = \lfloor \tan\phi(\nu - N/2)\rfloor + N/2 + 1$; if, on the other hand, $\pi/4 < \phi < 3\pi/4$, we take the same y-coordinate, $y = y_\mu$, and compute $x = \lfloor \cot\phi(\mu - N/2)\rfloor + N/2 + 1$.

Figures 4 and 5 demonstrate the recovery of edges from incomplete data, using the TV-based compressed sensing edge detector. Here, we randomly sample discrete Fourier coefficients and gather all the samples at grids along each of 100 radial lines in the spectral domain. The results are compared with the standard back projection, when unobserved frequencies are taken as zero (minimal L^2-energy), which should be contrasted with the minimal TV (or L^1-energy of differences) employed by the compressed sensing approach. Next we turn to a different setup of an image given in terms of its sampled gridvalues in the *physical space*. Our compressed sensing-based edge detector is tested with incomplete data which was randomly distributed on rectangular grids, shown in Fig. 6.

We note that in these examples, one recovers the edge detectors, $g(x)$, which should be viewed as two-dimensional concentration kernels, $g(x) \approx K_N^\sigma(x)$. Edges are sought as the extrema of these approximate concentration kernels, and are captured by the Sobel detection method, [5], where extrema of $g(x)$ are identified with zero crossings of its proper (discrete) gradient.

5 Details: The Zero Crossing Compressed Sensing Edge Detection

5.1 Sparsity and Post Processing

Following the discussion in Sect. 2.3, the zero crossing edge detection method recovers edges from incomplete data set Ω. It traces local extrema of gradients of the concentration kernels, by capturing the zero crossings of (appropriate combination of) second derivatives.

As pointed out in Sect. 2.3, with the incomplete spectral data $\widehat{\Delta\eta} \cdot \widehat{f}$ for $k \in \Omega$, our purpose in using the TV-based compressed sensing is to recover the complete "signal", $\Delta\eta * S_N[f]$. According to Lemma 4.2, if

$$B \leq C_m \cdot (\log N)^{-1} \cdot |\Omega|,$$

then with probability at least $1 - \mathcal{O}(N^{-m})$, the minimizer of (2.10) uniquely recovers $\widehat{\Delta\eta} \cdot \widehat{f}$. As usual, we assume finitely many edges whose number is denoted by B and $|\Omega|$ is the number of available spectral modes. Also, the recovery is guaranteed if Ω is large enough so that

$$B + |\Omega| \geq \text{Const.} (\log N)^{-1/2} N.$$

The output of this procedure is the convolved Laplacian (in our computations, convolved with the Gaussian function). Seeking edges as the zero crossing of this convolved Laplacian may yield spurious edges, due to inflection points and points with small amplitude second derivatives. Hence, we need to post process the zero crossing points in order to rule out these "false" edge points. Here we employ one simple rule to post process the zero crossing points where $g(x_\nu, y_\mu) = 0$: We check whether $|g_x(x_\nu, y_\mu)| \neq 0$, so that $g(x_{\nu-1}, y_\mu) \cdot g(x_{\nu+1}, y_\mu) < 0$; similarly, the ruling in the y direction requires $g(x_\nu, y_{\mu-1}) \cdot g(x_\nu, y_{\mu+1}) < 0$. More complicated post processing to remove redundancy can be found in [6, 18].

5.2 Numerical Results

We present the result of compressed sensing-based zero crossing in Fig. 7. Here, we are given incomplete Fourier coefficients $\widehat{f_k}$ of the image. By using the compressed

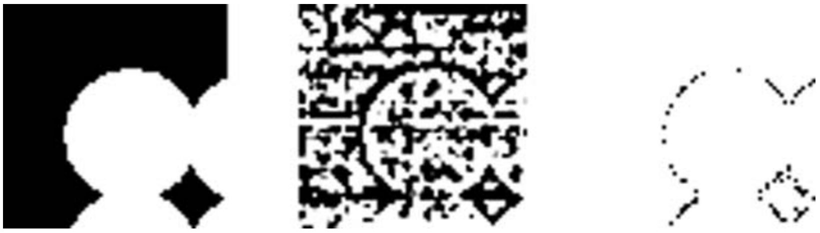

Fig. 7 Demonstrating the recovery results by zero crossing compressed sensing model. The available 2000 data points are uniformly and randomly distributed over 64^2 grids. *Left*: original image. *Center*: the image recovered from incomplete data by back projection method. *Right*: the result obtained by zero crossing compressed sensing method. Here we choose the parameters $\gamma = 3$

sensing-base zero crossing approach outlined in Algorithm 2.3, most of edge points are captured and there are very little artifacts.

6 Conclusion

In this paper, we utilize conjugate concentration kernels and sparse representation to recover edges from spectral information of piecewise smooth data. We propose the sIFT-based edge detection method, the TV-based compressed sensing method and an improved zero-crossing method as novel methods to detect edges from both complete and incomplete spectral data, which is possibly contaminated with noise. Experimental evidence supports the relative advantage of these novel methods as effective edge detectors in practical applications.

Acknowledgements We thank Anne Gelb and Rick Archibald for providing their minmod codes. J.Z. thanks Emmanuel Candès and Tony Chan for hosting her in CalTech and in UCLA, and Ingrid Daubechies, Justin Romberg and Jared Tanner for valuable discussions.

Appendix: The sIFT Sub-algorithms

We provide here the details for serious sub-algorithms involved in the process of sIFT-based edge detection discussed in Algorithm 2.2.

Algorithm 7.1 (Group testing)

Input: signal F, the length N of the signal F.

Initialize: set the signal F to F_0, iterative step $i = 0$, the length N of the signal, the accumulation factor $q = 1$, the number of nonzero taps of the filter χ.

In the ith iteration,

(1) If $q \geq N$, then return 0.
(2) Find the most significant bit v and the number of significant intervals c by the MSB Algorithm 7.2.
(3) Update $i = i + 1$, modulate the signal F_i by $\frac{(v+0.5)N}{4(2k_1+1)}$ and dilate it by a factor of $4(2k_1 + 1)/c$. Store it in F_{i+1}.
(4) Call the group testing algorithm again with the new signal F_i; store its result in g.
(5) Update the accumulation factor $q = q \cdot 4(2k_1 + 1)/c$.
(6) If $g > N/2$, then $g = g - N$.
(7) Return $\mod(\lfloor \frac{cg}{4(2k+1)} + \frac{(v+1/2)N}{4(2k_1+1)} + 0.5 \rfloor, N)$.

The group testing makes use of following procedure, the so called Most Significant bit (MSB) algorithm. It identifies the location of the jump by computing its significant bits one by one.

Algorithm 7.2 (Most Significant Bit (MSB))

Input: signal F with length N, a threshold $0 < \eta < 1$.

(1) Get a series of new signals $G_j(k) = D\widetilde{F} \star (e^{ikx/4(2k_1+1)} H_{k_1})$, $j = 0, \dots, 8k_1 + 4$. That is, each signal G_j concentrates on the pass region $[\frac{(j-1/2)N}{4(2k_1+1)}, \frac{(j+1/2)N}{4(2k_1+1)}] := pass_j$.
(2) Estimate the energies e_j of G_j, $j = 0, \dots, 8k_1 + 4$.
(3) Let l be the index for the signal with the maximum energy.
(4) Compare the energies of all other signals with the lth signal. If $e_i < \eta e_l$, label it as an interval with small energy.
(5) Take the center v_s of the longest chain of consecutive small energy intervals, suppose there are c_s intervals altogether in this chain.
(6) The center of the large energy intervals is $v = 4(2k_1 + 1) - v_s$, the number of intervals with large energy is $c = 4(2k_1 + 1) - c_s$.
(7) If $c > 4(2k_1 + 1)/2$, then do the original MSB [19] to get v and set $c = 2$, and $v =$ center of the interval with maximal energy.
(8) Output the dilation factor c and the most significant bit v.

A key procedure in group testing requires us to estimate the energy of a signal, as a criteria for excluding the less energetic half intervals. Ideally, the signals to be processed have most of the energy concentrate in one location, e.g., with $N = 16$, the signals to be processed are of the form

$$\widehat{F}(k) = \chi_1(k)e^{2ij\pi k/16} \cdot \chi_{[-q_1,q_1]}(\tau k)e^{2i\pi k\theta/N} \cdot \widehat{S}_k, \quad j = 0, 1, \dots, 15. \tag{7.1}$$

In the group testing procedure one needs to compare the energies of different signals. Instead of an $\mathcal{O}(N)$ exact computation of the energy, we use the following faster energy estimation.

Algorithm 7.3 (Energy estimation)
Input: signal $\widehat{F}$ to be processed by the randomly permuted group testing with success probability $1 - \delta$.

(1) Initialize: the number of samples: $r = \lfloor 12.5 \ln(1/\delta) \rfloor$.
(2) Take r independent random samples from the signal $\widehat{F}$: $\widehat{F}(k_1), \dots, \widehat{F}(k_r)$, where r is a multiple of 5.
(3) Return $N\times$ "60-th percentile of" $|\widehat{F}(k_1)|^2, \dots, |\widehat{F}(k_r)|^2$.

Following [19, Lemma 3.7], we can show that if a discrete signal $\widehat{F}$ is 93% "pure" and the number of samples $r > 12.5 \ln(1/\delta)$, then the output of Algorithm 7.3, X, satisfies

$$\|X\|_{\ell_2}^2 \geq 0.3 \|\widehat{F}\|_{\ell_2}^2$$

with probability exceeding $1 - \delta$.

References

1. Alvarez, L., Morel, J.-M.: Formulation and computational aspects of image analysis. Acta. Numer. **3**, 1–59 (1994)

2. Candes, E., Romberg, J., Tao, T.: Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Trans. Inf. Theory **52**, 489–509 (2004)
3. Candes, E., Romberg, J., Tao, T.: Stable signal recovery from incomplete and inaccurate measurements. Commun. Pure Appl. Math. **59**, 1207–1223 (2005)
4. Romberg, J., Candes, E.: l1 magic software, http://www.acm.caltech.edu/l1magic/
5. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**, 679–714 (1986)
6. Clark, J.J.: Authenticating edges produced by zero-crossing algorithms. IEEE Trans. Pattern Anal. Mach. Intell. **2**(1), 43–57 (1989)
7. Donoho, D., Elad, M., Temlyakov, V.: Stable recovery of sparse overcomplete representations in the presence of noise. IEEE Trans. Inf. Theory **52**, 6–18 (2006)
8. Donoho, D., Tanner, J.: Sparse nonnegative solutions of underdetermined linear equations by linear programming. Proc. Natl. Acad. Sci. **102**(27), 9446–9451 (2005)
9. Engelberg, S., Tadmor, E.: Recovery of edges from spectral data with noise—a new perspective. SIAM J. Numer. Anal. **46**(5), 2620–2635 (2008)
10. Fleck, M.M.: Some defects in finite-difference edge finders. IEEE Trans. Pattern Anal. Mach. Intell. **14**(3), 337–345 (1992)
11. Gelb, A., Tadmor, E.: Detection of edges in spectral data. Appl. Comput. Harmon. Anal. **7**, 101–135 (1999)
12. Gelb, A., Tadmor, E.: Detection of edges in spectral data II. Nonlinear enhancement. SIAM J. Numer. Anal. **38**, 1389–1408 (2001)
13. Gelb, A., Tadmor, E.: Spectral reconstruction of piecewise smooth functions from their discrete data. Math. Model. Numer. Anal. **36**(2), 155–175 (2002)
14. Gelb, A., Tadmor, E.: Adaptive edge detectors for piecewise smooth data based on the minmod limiter. J. Sci. Comput. **23**(2–3), 279–306 (2006)
15. Marr, D., Hildreth, E.: Theory of edge detection. Proc. R. Soc. Lond. B **207**, 187–217 (1980)
16. Rudelson, M., Vershynin, R.: Geometric approach to error correcting codes and reconstruction of signals. Int. Math. Res. Not. **64**, 4019–4041 (2005)
17. Tadmor, E.: Filters, mollifiers and the computation of the Gibbs phenomenon. Acta Numer. **16**, 305–378 (2007)
18. Ulupinar, F., Medioni, G.: Refining edges detected by a LoG operator. Comput. Vis. Graph. Image Process. **51**(3), 275–298 (1990)
19. Zou, J., Gilbert, A., Strauss, M., Daubechies, I.: Theoretical and experimental analysis of a randomized algorithm for sparse Fourier transform analysis. J. Comput. Phys. **211**, 572–595 (2006)
20. Zou, J.: A sublinear algorithm for the recovery of signals with sparse Fourier Transform when many samples are missing. Appl. Comput. Harmon. Anal. **22**, 61–77 (2007)
21. Zou, J.: Sublinear Algorithms for the Fourier transform of signals with very few Fourier modes: theory, implementations and applications. Ph.D. Dissertation, Princeton University, 2005