Abstract. Detecting edges in piecewise smooth images is important in a variety of applications. In some cases, such as magnetic resonance imaging, data are collected as Fourier samples, and are often over-sampled at low frequencies but more sparsely acquired in the high frequency range. This under-sampling causes Fourier based edge detection algorithms to yield false positives in smooth regions. The problem is compounded when the data are noisy. In this paper we demonstrate that even when the given data are noisy and under-sampled, the edge response behavior obtained using certain Fourier based edge detection methods can be accurately characterized. In particular, the responses of different high order methods vary little, both in regions away from the discontinuities and at true edges. By contrast, the variance is large in cells neighboring each true edge. By locating these “double peaks” in variance, it is possible to determine edges from noisy and under-sampled Fourier data. Alternative methods of generating and refining variance data are also explored. Specifically, a method based on regularized image reconstruction that leverages the sparsity of edges and image smoothness between these edges is effective at eliminating false positives.

1. Introduction. Several important applications including magnetic resonance imaging (MRI) and synthetic aperture radar (SAR) acquire data by way of Fourier sampling, [1, 5, 14, 19]. To reduce data acquisition times, some MRI techniques forgo collection of frequency data on a uniform Cartesian grid, opting instead to collect information on trajectories that densely sample low frequencies and sub-sample high frequencies, [9,10,13,21]. Recovering edge information from this data is useful in a variety of situations, such as producing edge maps that can be interpreted directly by practitioners, as a source of data for feature detection and categorization, or as information that can be used in combination with other processing techniques to enhance image recovery.

The concentration factor edge detection method recovers edge information directly from acquired Fourier data, [6]. It operates by applying a set of concentration factors to the supplied Fourier samples. Starting with a complete, band limited set of Fourier samples the recovered “edge map”, or synonymously “jump map”, exhibits a corresponding response at the location of true edges, but also exhibits oscillations that decay away from edges. Isolating the true edges requires some form of thresholding. The particular structure of these oscillations is dependent on the chosen set of concentration factors, and in [7] results from different concentration factors were combined with the minmod algorithm to remove some of these oscillations. Unfortunately, additional oscillations occur when the sampled Fourier data are noisy or in cases where the band of Fourier coefficients is not adequately resolved. Unless removed by additional thresholding, these oscillations in smooth regions translate into false positive edge detections. But the additional thresholding may cause failure in identifying true edges.

In this paper we demonstrate that in spite of these false positives, there is valuable information contained in the structure of the oscillations in the neighborhood of jumps that can be extracted to recover the edges of the underlying image. In particular we note that at jump locations and in smooth regions the different jump approximations are similar, while in the neighborhood of jumps there is an exploitable variance in behavior. On measuring this variance, we note a “two peak” signature surrounding jumps that is not generally present around spurious oscillations not in the vicinity of edges. Thus we develop an algorithm that uses these variance signatures as a way to reject the false positives in the edge map approximation.

Measuring the variance in behavior between concentration factors to eliminate false positives succeeds when the false positives are the result of oscillations inherent in the concentration method edge approximations. However, false edges in the Fourier reconstruction from under-sampled or noisy Fourier data may become indistinguishable from true edges without additional prior information. To eliminate these false edges, we call upon the assumption that the underlying function is piecewise smooth with a sparse edge map. In [2], piecewise smooth functions were reconstructed from under-sampled and noisy Fourier data using $l^1$ regularization to promote the sparsity of edges. In spite of sub-sampling and noise, smooth regions are
recovered accurately with this method. To achieve this accuracy, the sparsifying transform must distinguish smooth regions with high regularity from the edges. In particular, Total Variation (TV) cannot be used for the sparsifying transform because of the resulting “staircasing effect”. Instead the polynomial annihilation (PA) transform, [3], is used, resulting in faster convergence to the underlying function in the smooth regions. In this investigation, we make the observation that the $\ell^1$ regularization reconstruction method (equipped with the polynomial annihilation transform), consistently reconstructs smooth regions from a variety of significantly under-sampled noisy Fourier data. Indeed, even when the reconstructed image is under-resolved, the consistency of the reconstruction in the smooth regions yields enough information via its variance signature to detect edges. Thus we introduce treatments involving subsets of the already sub-sampled Fourier data set. The variance in the reconstruction from these subsets exhibits a similar “two peak” signature, which can again be used to eliminate false positive edge detections. The important advantage of this method is that false jumps resulting from the incomplete and noisy Fourier samples are distinguishable from true jumps based on the regularized reconstruction.

The rest of the paper is organized as follows. In Section 2 we review the necessary background and establish the test case framework. In Section 3 we introduce our new variance signature approach based on both the concentration factor edge detection method, developed in [6], and the $\ell^1$ regularization technique using the polynomial annihilation transform, [2]. In Section 4 we extend the results to two dimensions. Concluding remarks are provided in Section 5.

2. Preliminaries. Let $f$ be a periodic piecewise smooth function defined on $[-1, 1)$. We define the corresponding jump function, $[f]$, as the difference between the right-hand and left-hand limits of the function i.e.

$$[f](x) = f(x^+) - f(x^-).$$

Thus in smooth regions $[f] = 0$ and at discontinuities $[f]$ is the value of the jump. Since our ultimate goal is to construct an edge map of $f$, we discretize $x$ uniformly as $D = \{x_j = -1 + \frac{2(j-1)}{2N}\}_{j=1}^{2N+1}$. We note that the jump function approximation method described here does not restrict the solution to uniform points, but using this distribution may improve its computational efficiency. We also make the assumption that there is at most one jump discontinuity within a cell $I_j = [x_j, x_{j+1})$, which is reasonable when data are acquired as the first $2N + 1$ Fourier coefficients, or some sparse subset of those values. Thus, if $[f](x_j)$ is the value of the jump occurring in $I_j$, we can write

$$[f](x) = \sum_{j=1}^{2N+1} [f](x_j) \chi_{I_j}(x),$$

where $\chi_{I_j}(x)$ is defined as

$$\chi_{I_j}(x) = \begin{cases} 1 & \text{if } x \in I_j \\ 0 & \text{for all other } x. \end{cases}$$

For simplicity, the numerical algorithms used in this investigation all place the jump discontinuity at the left boundary of its corresponding cell.

To demonstrate basic features of the algorithms discussed in this paper, we will consider the following function, displayed in Figure 1.

$$f_1(x) = \begin{cases} 2 - \frac{\pi x}{3} + \sin (\pi x + \frac{1}{16}) & -\frac{3}{4} \leq x < \frac{3}{16} \\ \frac{4\pi x - 9}{4\pi x} & \frac{1}{2} \leq x < \frac{15}{16} \\ 0 & \text{otherwise}, \end{cases}$$

where

$$[f_1](x) = \begin{cases} 2.04 & x = -\frac{3}{4} \\ -2.41 & x = \frac{3}{16} \\ -2.72 & x = \frac{1}{2} \\ 2.71 & x = \frac{15}{16}. \end{cases}$$
2.1. The concentration factor edge detection method. For algorithmic development, let us assume that \( f \) has a single discontinuity at \( x = \xi, \xi \in (-1,1) \). Suppose we are given a (sparse) subset of Fourier coefficients,

\[
\hat{f}(k) = \frac{1}{2} \int_{-1}^{1} f(x) e^{-ik\pi x} dx, \quad -N \leq k \leq N.
\]

Integration by parts provides the relationship

\[
\hat{f}(k) = [f]\xi e^{-ik\pi \xi} 2\pi i k + O \left( \frac{1}{k^2} \right),
\]

where \([f]\xi\) is the value of the jump at \( x = \xi \). In [6], (6) was used to generate the concentration factor edge detection method, given by

\[
S_N^{\sigma} [f](x) = i \sum_{k=-N, k \neq 0}^{N} \hat{f}(k) \text{sgn}(k) \sigma \left( \frac{|k|}{N} \right) e^{ik\pi x},
\]

where the “concentration factor” \( \sigma(\eta), \eta \in [0,1] \), discretized at \( \left( \frac{|k|}{N} \right) \), satisfies the following admissibility requirements:

1. \( \sum_{k=1}^{N} \sigma \left( \frac{k}{N} \right) \sin (kx) \) is odd
2. \( \sigma(\eta) \in C^2 (0,1) \)
3. \( \int_{\epsilon}^{\pi} \sigma(\eta) \frac{d\eta}{\eta} \to -\pi \) where \( \epsilon \) is small.

Some examples of admissible concentration factors are listed in Table 1, while the results of \( S_N^{\sigma} [f_1](x) \) for \( N = 32, 64, \) and 128 are shown in Figure 3(a).

Due to the relationship obtained in (6), the concentration factor edge detection method in (7) can be characterized using the ramp function \( r_\xi(x) = r(x - \xi) \), where

\[
r(x) = \begin{cases} 
-\frac{x+1}{2} & -1 \leq x < 0 \\
-\frac{x-1}{2} & 0 \leq x < 1,
\end{cases}
\]

and

\[
[r_\xi](x) = \begin{cases} 
1 & x = \xi \\
0 & x \neq \xi.
\end{cases}
\]

The corresponding Fourier coefficients of \( r_\xi(x) \) are

\[
\hat{r}_\xi(k) = \begin{cases} 
e^{-ik\pi \xi} & k \neq 0 \\
0 & k = 0.
\end{cases}
\]
Concentration factor type | expression | details
--- | --- | ---
Trigonometric | $\sigma_G(\eta) = \frac{\sin(\pi \eta)}{\sin(\pi)}$ | $\text{Si}(\pi) = \int_0^\pi \sin x \, dx$
Polynomial | $\sigma_p^p(\eta) = p \pi \eta^p$ | $p$ is the order of polynomial concentration factor.
Exponential | $\sigma_E^\alpha(\eta) = C \eta e^{(\pi - 1)} \eta^{(\pi - 1)}$ | $\alpha$ is the order of the exponential concentration factor.
$C$ is a scale factor given by
$C = \frac{1}{f_1^{-1/N} e^{\pi/2}} \int_{-1/2}^{1/2} \, dr$

Table 1: Sample concentration factors

Using the above notation, the linear approximation of piecewise smooth $f$ with a single discontinuity at $x = \xi$ can now be written as

\begin{equation}
(11) \quad f(x) \approx [f](\xi) r_\xi(x).
\end{equation}

Analogously, if $f$ has jump discontinuities at $\xi_m$, $m = 1, \ldots, M$, the piecewise linear approximation of $f$ is

\begin{equation}
(12) \quad f(x) \approx R(x) = \sum_{m=1}^M [f](\xi_m) r_{\xi_m}(x).
\end{equation}

Substituting the corresponding (multiple jump) values of $r_{\xi}(k)$ in (10) into $\hat{f}_k$ in (6), it is evident that (7) approximates $[R](x)$. Moreover, by translating each discontinuity location a distance $\xi_m$, we can further characterize (7) by defining

**Definition 1 (The jump response).** Given concentration factor $\sigma$, we define the jump response as

\begin{equation}
(13) \quad W_N^\sigma(x) := S_N^\sigma[r](x) = \frac{1}{2\pi} \sum_{0 < k \leq N} \frac{\sigma(\frac{|k|}{N})}{|k|} e^{ik\pi x}.
\end{equation}

Figure 2 shows the jump response using various specific concentration factors. It demonstrates that in the neighborhood of a jump discontinuity, the behavior of (13) is highly dependent on the particular concentration factor chosen, while away from discontinuities the convergence rate is $O(\frac{1}{N})$. These different behavior patterns have been exploited in [7], where post processing algorithms were developed to pinpoint the edges. While such techniques were shown to be effective given all $2N+1$ Fourier samples in (5), we will demonstrate that they are not as robust when the data are noisy or further sub-sampled. Instead we will employ an $l^1$ regularization technique that exploits the sparsity of $[f](x)$, as observed by the sparse number of non-zero coefficients in (2). We first review $l^1$ regularization below.

**2.2. Sparsity promoting regularization.** Regularization is a well-known technique for solving ill-conditioned inverse problems, and regularizations using $l^1$ penalty terms have enjoyed great success in the fields of image processing and compressed sensing, [15, 20]. A driving principle in many of these techniques is that some attribute of a good solution will be sparse. An ideal $l^0$ penalty term would seek to minimize the count of non-zero entries in its argument, but $l^0$ terms are not convex and a general numerical solution is impossible. Hence $l^1$ terms are used as convex relaxations for $l^0$ terms in compressed sensing, [4]. For the algorithms described below we will use regularizations of the general form

\begin{equation}
(14) \quad \tilde{u}^* = \arg\min_{\tilde{u}} \left( \|G\tilde{u} - \mathcal{H}\tilde{b}\|_2^2 + \sum_{\mu} \lambda_{\mu} \|\phi_{\mu}(\tilde{u})\|_1 \right).
\end{equation}

Here $\tilde{u}^*$ is the desired solution, $\tilde{b}$ is the provided data set, $G$ and $\mathcal{H}$ are optional operators to make trial solutions $\tilde{u}$ compatible with the provided data set $\tilde{b}$, $\{\phi_{\mu}\}$ is a set of sparsifying operators, and $\{\lambda_{\mu}\}$ is a
set of regularization parameters weighting individual penalty terms. For the algorithms described below the operators $G$, $H$ and $\{\lambda_\mu\}$ will be linear and the minimization problem in two dimensions can be efficiently solved using the Split-Bregman algorithm, [8].

2.3. Polynomial annihilation edge detection as a sparsifying operator. Recall that $f$ is a piecewise smooth periodic function on $[-1, 1)$. We now review the polynomial annihilation method that was introduced in [3] and used as a sparsifying operator in a penalty term in [2, 16, 18] as a means for reconstructing functions from sparse or noisy Fourier data. Although the method was developed for multiple dimensions and non-uniform data, we have found that for a uniform grid it is most efficient to employ the polynomial annihilation operator dimension by dimension, [2,18]. Thus we describe the technique for given data $f(x_j)$, $x_j = -1 + \frac{2(j-1)}{2N}$, $j = 1, \ldots, 2N + 1$. In practice $f(x_j)$ will be the elements of the solution vector $u$ in (14), and $L^p f(y)$, $y \in [-1, 1)$, given in (15) will be recovered on uniform points $y_n = x_n$. The polynomial annihilation edge detection method is defined as [3]:

$$L^p f(y) = \frac{1}{q^p(y)} \sum_{x_j \in S} c_j(y) f(x_j),$$

where $c_j(y)$ are polynomial annihilation coefficients, $q^p(y)$ is a normalization factor, and $S_x$ is a set of $p + 1$ grid points surrounding $y$, which can be extended periodically as necessary.\(^1\) The polynomial annihilation coefficients, $c_j(y)$, are designed to annihilate polynomials up to degree $p$, and are obtained by solving the system

$$\sum_{x_j \in S} c_j(\zeta) \varphi_\ell(x_j) = \varphi_\ell^{(p)}(\zeta), \quad j = 1, \ldots, p + 1,$$

where $\varphi_\ell$, $\ell = 0, \ldots, p$, is a basis of for the space of polynomials of degree $\leq p$ and $\zeta \in [-1, 1)$. With the restrictions just described, the solution to (16) is given by [3]

$$c_j = \frac{p!}{\prod_{k=1 \ldots p+1, k \neq j} (j-k) h},$$

\(^1\)The polynomial annihilation method does not restrict the class of underlying functions to be periodic. Indeed, the stencils $S$ can be made one sided as the boundaries of the domain are approached.
where \( h \) is the distance between grid points \( \frac{1}{N} \).

The normalization factor, \( q^p(y) \), assures the proper convergence of \( L^p f(y) \) to the jump value at each discontinuity and is given by

\[
q^p(y) = \sum_{x_j \in S, x_j \geq y} c_j(y).
\]

Since \( f \) is periodic and the reconstruction grid points in (15) are \( y_n = x_n \), we write \( q^p_n := q^p(y_n) \). We can therefore define the polynomial edge detection operator \( \mathbf{P}^p \) as a circulant matrix such that

\[
\mathbf{P}^p_{y,n} = \frac{\tilde{c}(j,n)}{q^p_n}, \quad 1 \leq n \leq 2N+1, \quad 1 \leq j \leq 2N+1,
\]

where

\[
\tilde{c}(j,n) = \begin{cases} \begin{align*} c_{s(j,n)} & \text{ if } 0 < s(j,n) \leq p + 1, \\ 0 & \text{ otherwise,} \end{align*} \end{cases}
\]

with \( c_{s(j,n)} \) defined by (17). For example, when \( p = 2 \) we have

\[
\mathbf{P}^2 = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & -1 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ -1 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix}.
\]

The derivation of (19) and other examples for \( p \) can be found in [2].

2.4. Sample sets. We assume the Fourier coefficients of a piecewise periodic smooth function, \( f \), are drawn from a set \( \hat{U} = \{ \hat{f}_k : -N \leq k \leq N \} \), and that these Fourier coefficients may be subject to complex Gaussian noise. Each test case is characterized by a set of parameters. We let SNR represent the signal to noise ratio, with \( \text{SNR} = \infty \) representing no noise. The initial set of coefficients, \( \hat{U} \), may be sub-sampled as controlled by parameters \( \gamma \), \( \beta \), and \( \zeta \) to form

\[
\hat{V} = \hat{F} \cup \hat{R},
\]

where \( \dim(\hat{V}) = \gamma(2N+1) \),

\[
\hat{F} = \{ \hat{f}_k : -\beta N \leq k \leq \beta N \},
\]

with \( 0 \leq \beta \leq 1 \), and \( \hat{R} \) is a set of randomly selected coefficients \( \hat{f}_k \), \( |k| > \beta N \) (\( |k| \leq N \) when \( \beta = 0 \)) from a normal distribution.

Now, for each test we construct \( q = 1, \cdots, Q \) subsets from \( \hat{V} \) as

\[
\hat{T}_q = \hat{F} \cup \hat{R}_q,
\]

where \( \hat{R}_q \) has \( \zeta(\gamma - \beta)(2N+1) \) randomly selected values of \( \hat{R} \). We will apply the numerical algorithms in this investigation to a set of test cases with parameters specified in Table 2.

Different values of \( N \) will demonstrate the various features of our new algorithm. In applications where the sampling is limited to the small wave numbers, that is small \( N \), we do not expect to use much compression. However, as discussed in the introduction, in applications such as MRI, some sampling trajectories are designed to oversample the low frequency modes while sparsely sampling in the high frequency range. Moreover, if very large data sets are obtained, compression could be used to process the data more efficiently. Our numerical experiments give insight into both situations. Similarly, we demonstrate that our method is robust when the given data are noisy.

\[\text{Numerical experiments were also performed for uniformly distributed sets with no noticeable difference for } \beta \geq 0.3 \text{ and various choices of } \gamma. \text{ However, for } 0 < \beta < 0.3 \text{ and } \gamma < 0.1, \text{ it becomes imperative to keep the low modes of the given data set.}\]
<table>
<thead>
<tr>
<th>Test Case</th>
<th>SNR</th>
<th>$\beta$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\infty$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>$\infty$</td>
<td>0.3</td>
<td>0.75</td>
</tr>
<tr>
<td>3</td>
<td>12.5dB</td>
<td>0.3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>12.5dB</td>
<td>0.3</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Parameters selected for each test case. Unless otherwise specified, we also choose $N = 64$, $Q = 30$ and $\zeta = .5$.

3. Edge detection for under-sampled Fourier data.

3.1. The concentration factor edge detection method. We first examine the direct application of the concentration factor edge detection method, given in (7). As shown in Figure 3(a), this method works well when given the full set of $2N + 1$ noiseless Fourier coefficients, but it loses its effectiveness as either the signal to noise ratio decreases, (Figure 3(c)), or the number of available Fourier samples decreases, (Figure 3(b) and (d)). Oscillations appear in the smooth regions, and as they increase in magnitude, effective thresholding becomes more difficult.

![Fig. 3. Concentration factor edge detection method, $S_{N}^{\sigma}[f](x)$](image)

The minmod algorithm was used in [7] to exploit the variability in the jump responses apparent in Figure 2. Specifically, the results in (7) using admissible concentration factors, $\sigma_1, \cdots, \sigma_n$, were jointly post processed as

\[
S_{N}^{MM}[f](x) = \minmod\{S_{N}^{\sigma_1}[f](x), S_{N}^{\sigma_2}[f](x), \ldots, S_{N}^{\sigma_n}[f](x)\},
\]

where

\[
\minmod\{a_1, a_2, \ldots a_n\} = \begin{cases} s \min(|a_1|, |a_2|, \ldots|a_n|) & \text{if } \sgn(a_1) = \cdots = \sgn(a_n) = s \\ 0 & \text{otherwise.} \end{cases}
\]

7
The results of (22) using the set \( \{ \sigma_G, \sigma_P^1, \sigma_P^2, \sigma_E^1, \sigma_E^2 \} \) are displayed in Figure 4. Although some artificial oscillations are reduced, these improvements become less evident as more noise and less data are used. In some cases, adding more concentration factors in (22) may help in further reducing the magnitude of the oscillations. That said, although (22) is not an effective means of differentiating the results using various concentration factors in these cases, it still is possible that the variation resulting from different concentration factors may be exploited. In Section 3.4 we introduce our new variance signature algorithm with this in mind.

![Graphs of various test cases](image)

**Fig. 4.** \( S_{MMN}[f_1](x) \) in (22)

### 3.2. The sparsity enforcing edge detection method.

The sparsity enforcing edge detection method for determining edges in noisy and/or sub-sampled environments using regularization was developed in [17] and given by

\[
S_{\sigma}^{SE}[f] = \arg\min_\hat{g} \left( \| \hat{g} - S_\sigma^N[f] \|_2^2 + \lambda \| \hat{g} \|_1 \right).
\]

Figure 5 displays the results of (23) using \( \sigma_G \). Comparing Figures 3, 4, and 5, we see that (23) yields better results than (7) when given all \( 2N + 1 \) noiseless coefficients, although post processing with the minmod algorithm, (22), is more effective. However, using the regularization in (23) is more robust in cases where data are noisy and under-sampled, although the results inherently depend on the regularization parameter chosen. Thus we observe a tradeoff between artificial jump suppression and the failure to detect true edges. Nonetheless, this method provides an effective jump response that varies from those generated using the concentration factor edge detection method in (7), which may be exploited in the procedure described below.

### 3.3. Generating an edge map using thresholding.

Let us call \( \hat{E}[f] \) the result acquired either by (7), (22), or (23) at the grid points \( x_j, j = 1, \ldots, 2N + 1 \). It is evident that some degree of post processing is needed to reduce the number of false positives and form an accurate edge map. One way to do this is to use thresholding. Rather than prescribe a cut off value related to \( |\hat{E}[f]_j| \), which may not be robust with respect to noise or sub-sampling, we appeal to the presumed sparsity of the underlying jump function, \( |f| \), to
Fig. 5. $S^{SE}_N[f_1](x)$ resulting from (23).

Fig. 6. Algorithm 1 using $S^{SG}_N[f_1](x)$, $S^{MM}_N[f_1](x)$, and $S^{SE}_N[f_1](x)$ and threshold $c = 7/8$. 
choose a threshold, \( \epsilon = \epsilon(c) \), such that \( \text{dim}(\{E[f]_j : |E[f]_j| < \epsilon, j = 1, \cdots, 2N + 1\}) = c(2N + 1) \). Typically \( c \in \left( \frac{2}{10}, 1 \right) \). Although the corresponding threshold is seemingly large, it is reasonable under the assumption that \([f]\) has a sparse representation, and that there should be relatively few values of \(|E[f]_j| > \epsilon\). Additionally, only the extrema of \(E[f]\) are considered as jumps in the post processed jump function approximation, \(E^{PP}[f]\), and we require that these extrema be separated by a distance of at least \(d\). That is, we define each local jump region to have a distance of \(d\), where \(d\) is dependent on the noise and amount of sub-sampling. The process of thresholding and extrema detection can then be combined as shown in Algorithm 1.

**Algorithm 1 Generating an edge map using thresholding**

Choose a jump function approximation method to construct \(E[f]\), a threshold \(\epsilon = \epsilon(c)\) as described above, and a distance \(d\) which defines the distance of each local jump region. Although not necessary for periodic functions, for simplicity assume that no jumps occur between \([x(1), x(d)]\) and \([x(2N+1-d), x(2N+1)]\).

- for \(j = d, \cdots, 2N+1-d\)
- if \(|E[f]_j| > \epsilon\) and \(|E[f]_j| > |E[f]_l|\) for \(|x_j - x_l| < d\), then
  \[E^{PP}[f]_j = E[f]_j\]
- else
  \[E^{PP}[f]_j = 0\]
- end if
- end for

Figure 6 compares the thresholded edge maps generated by \(S^G_N[f_1](x)\), \(S^{MM}_N[f_1](x)\), and \(S^E_N[f_1](x)\), with \(c = \frac{5}{8}\). We note that a larger value for \(c\) would isolate the four true jumps, but as can be seen in test case 4, the scale of false jumps approaches that of true jumps. Without advance knowledge of the true jump heights, increasing the threshold, \(c\), risks eliminating true jumps.

**3.4. The variance signature of jump function responses.** As is evident from Figure 6, simple thresholding becomes less effective as noise is increased or as the data are further under-sampled. We now introduce a new method for detecting edges by defining the variance vector of the set of jump function approximations, \(E = \{E^{1}[f], \cdots, E^{n}[f]\}\), as

\[
\bar{v}(E)_j = \frac{1}{\text{dim}(E)} \sum_{E \in \bar{E}} (E^{\nu}[f]_j - \frac{1}{\text{dim}(E)} \sum_{E \in \bar{E}} E^{\nu}[f]_j)^2, \quad j = 1 \cdots 2N + 1.
\]

(24)

From the previous discussions, possible sets of jump function approximations evaluated at these grid points include

\[
E_1 = \{S^G_N \bar{f}\}_{\sigma \in G},
\]

(25a)

\[
E_2 = \{S^{G \nu}_N \bar{f}\}_{\sigma \in G} \cup \{S^E_N \bar{f}\},
\]

(25b)

where \(G\) is a set of concentration factors and \(S^E_N[f]\) is given in (23). The result of (24) with \(E_1\) for \(f(x) = r(x)\) is displayed in Figure 7. It is evident that in regions away from the neighborhood of the jump, \(\xi = 0\), \(\bar{v}(E_1)\) is small as desired. The largest variance is seen in the neighborhood of the jump discontinuities, but the variance at \(x = \xi\) is again small. This behavior is consistent with the oscillatory behavior observed in the jump responses for each concentration factor observed in Figure 2. Figure 8 displays the variance using \(E_1\) for the same concentration factors on \(f_1(x)\).

The results of (24) using \(E_2\) are shown in Figure 9. We see that including information from (23) does not seem to yield any more information that will help to isolate edges, so it is not utilized further in our investigation.

---

3We note that a related idea that exploits the jump responses in (13) using maximum likelihood estimators was developed in [12]. The main goal of that investigation was to produce the best ROC curve in noisy environments when given the first \(2N+1\) Fourier coefficients of a piecewise smooth function.
3.5. **Post processing edge detection using the variance signature.** The variance computed in (24) allows us to distinguish where an edge is likely to occur from an artifact caused by the oscillatory response of the jump function approximation. Below we provide an algorithm that allows us to convert the variance in (24) to an edge map.

As in Algorithm 1, we first determine which points \( x_j, j = 1, \cdots, 2N + 1 \), describe local jump regions in the domain \([-1, 1]\). It is evident from Figures 8 and 9 that each edge occurs between two locations where \( \tilde{v}(E) \) has a local maximum, and indeed we determine the jump location as that point in the jump region where \( S^N[f]^\sigma \) takes on its largest absolute value. Algorithm 2 describes how local jump regions are obtained using the variance signature in (24).

We note that Algorithm 2 will require thresholding to avoid false local region detections. We threshold \( \tilde{v}(E) \) in a manner similar to that applied in Algorithm 1, appealing to the presumed sparsity of the underlying jump function \( [f] \), and choosing a threshold, \( \epsilon(c) = \epsilon \), such that \( \dim(\{\tilde{v}(E)_m : \tilde{v}(E)_m < \epsilon, 1 \leq m \leq 2N + 1\}) = c(2N + 1) \). Typically \( \frac{3}{4} \leq c < 1 \). Algorithm 3 is then used to generate an edge map \( M \), prescribed on the grid points \( x_j \), by isolating the edges from within each jump region.
Algorithm 2 Local Jump Region Determination

Choose a variance threshold, $\epsilon$ as described above, and a maximum distance between peaks threshold, $\delta$, needed to establish two neighboring peaks. Although not needed for periodic functions, we also choose $d$ so that no jump occurs between $[x(1), x(d)]$ or $[x(2N + 1 - d), x(2N + 1)]$.

- Set counter for the number of potential edges, $\nu = 0$.
- for $j = d, \ldots, 2N + 1 - d$
  - if $\bar{v}(E)_j > \bar{v}(E)_{j-1}$ and $\bar{v}(E)_j > \bar{v}(E)_{j+1}$ and $\bar{v}(E)_j > \epsilon$, then
    1. $\nu = \nu + 1$
    2. $y_\nu = x_j$;
  - end if
- end for
- Set counter for the number of jump regions, $\ell = 0$.
- for $n = 1, \ldots, \nu - 1$
  - if $\text{dist}(y_n, y_{n+1}) < \delta$ then (we are in a jump region – otherwise, the large variance indicated a false edge)
    1. $\ell = \ell + 1$
    2. $B_\ell = \{ j : y_n \leq x_j \leq y_{n+1} \}$
  - end if
- end for

Figure 10 demonstrates application of Algorithm 3 on $f_1(x)$ using $E_1$ in (24). While the algorithm is effective in noiseless environments with the first $2N + 1$ Fourier coefficients given, it is evident that adding noise and reducing the size of the sampling set makes it difficult for the algorithm to recover true edges without producing false ones. Indeed, in test cases 3 and 4, the edge at $x = -0.75$ is completely missed while false edges occur in other parts of the domain. This is not so surprising given the results in Figure 8, which shows large variance in smooth regions. The convergence of the concentration factor edge detection method in smooth regions given the first $2N + 1$ Fourier coefficients is such that $\bar{v}(E_1)$ is a valid predictor.
Algorithm 3 Edge Map Generation

Input variables: Jump regions $B_\ell, \ell = 1, \cdots, L$, and $E_{PP}[f]$ from Algorithm 1.

Output variables: Edge map $\vec{M}$

- for $j = 1, \cdots, 2N + 1$
  - $\vec{M}_j = 0$
- end for
- for $\ell = 1, \cdots, L$
  - $\xi = \text{arg max}_{x_j \in B_\ell} |E_{PP}[f]_j|$
  - Define $k : x_k = \xi$
  - $\vec{M}_k = E_{PP}[f]_k$
- end for

of where edges are. However, no such proof of convergence exists for the concentration factor method when the data are sub-sampled as in (20). Hence we seek to develop an algorithm where the variance between the jump discontinuities, that is, in smooth regions, remains small, even when the data are sub-sampled. This is accomplished in Section 3.6.

3.6. Determining edges from regularized reconstruction using the variance signature. As observed in Section 3.5, the limitations on post processing the jump function approximation using (24) using either (25a) or (25b) are due to too much variability away from the edges in smooth regions. However, as will be shown below, using (24) may still be an effective tool when the input set, ($E$ in (24)), has the property that each element in $E$ differs in convergence properties only within each jump region, that is, the convergence is similar in smooth regions. Thus we seek a new set $E$ for which this property holds. As it turns out, the method developed in [2], which approximates piecewise smooth functions from under-sampled Fourier data using $l^1$ regularization, leads to such a set. The method is briefly reviewed below.

Once again we are given the (noisy) Fourier coefficients of a piecewise smooth function from the set
V in (20). For each sub-sampled set $T_q$ in (21), we reconstruct $f$ on a set of uniform grid-points $x_j, j = -N, \cdots, N$, as

$$\tilde{f}_q = \text{argmin}_{\tilde{u}} \|F_q \tilde{u} - \tilde{f}_q\|^2_2 + \lambda \|P^p \tilde{u}\|_1.$$  

Here $\tilde{f}_q$ is the vector of Fourier coefficients formed from the sub-sampled set $T_q$, and $F_q$ is the discrete Fourier transform generating the coefficients to match $\tilde{f}_q$. The polynomial annilation transform, $P^p$ in (19), is used as a sparsifying operator in the penalty term. We choose $p = 2$ in our numerical experiments. We note that when $p = 1$, which is equivalent to using total variation (TV) as the $l^1$ transform term, the reconstruction does not exhibit the needed variance in the neighborhood of edges, causing a shift in edge locations in (26). Consequently, (27) is not effective in this case. Conversely, choosing $p > 2$ creates more oscillations near the jump discontinuities, leading to an extended variance region with multiple peaks in each jump region, making it difficult to isolate the edges. Figure 11 shows the approximation of $f_1(x)$ for each of the four test cases and the corresponding pointwise errors. Observe that for the first two test cases, (26) converges accurately away from the discontinuities but is not accurate in the neighborhoods of the internal edges. For test cases 3 and 4, (26) is less accurate. Nonetheless, as we will show shortly, the solutions $\tilde{f}_q, q = 1, \cdots, Q$, vary very little in smooth regions, while they exhibit more variation near the discontinuities. As before, we will exploit this variation in the approximation to determine the edge locations of $f$. However, now we choose $E$ from (24) to contain the approximations of $f$ given in (26). Moreover, instead of choosing different approximation parameters, e.g. $\sigma$ in (7), we consider different sampling sets, $\hat{T}_q, q = 1, \cdots, Q$, in (21).

Thus for $Q = \{\tilde{f}_q\}_{q=1}^Q$, where each $\tilde{f}_q$ is calculated on a set of points $x_j, j = 1, \cdots, 2N + 1$, we have

$$\bar{v}(Q)_j = \frac{1}{Q} \sum_{q=1}^Q (\tilde{f}_q j - \frac{1}{Q} \sum_{q=1}^Q \tilde{f}_q j)^2, \quad j = 1 \cdots 2N + 1.$$  

Figure 12 show the results of (27) for $f_1(x)$ using each test case in Table 2. Once (27) is calculated, we can apply Algorithms 2 and 3 to recover an edge map, in this case with (27) replacing (24). Figure 13 shows these results.

4. Determining the two dimensional edge map using the variance signature. Below we describe how the methods developed in Section 3 can be expanded to two dimensional functions. In this case we assume that $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ is a periodic piecewise smooth function on $[-1, 1]^2$, and that we are given Fourier

---

4We note that it is possible to vary the sampling sets for $E$ in (24) as well. Unfortunately, for the same reasons as stated at the end of Section 3.5, using different sampling sets did not reduce the variance sufficiently in the smooth regions of the domain.
Fig. 12. $\mathbf{v}(\mathbf{Q})$ for $f_1$ using the parameters in Table 2

Fig. 13. Algorithms 2 and 3 applied to $\mathbf{v}(\mathbf{Q})$ for $f_1$ using the parameters in Table 2. Here $\delta = 5$ and $c = \frac{7}{8}$. 
coefficients

\begin{equation}
\hat{f}_{k,l} = \frac{1}{4} \int_{-1}^{1} \int_{-1}^{1} f(x,y) e^{-i\pi(kx+ly)} dydx,
\end{equation}

drawn from a set \( \hat{U} = \{ \hat{f}_{k,l} : -N \leq k, l \leq N \} \). For ease of presentation, we will consider the same test cases as those displayed in Table 2, with the equivalent parameters used in each direction.

![Fig. 14. \( f_2(x,y) \)](image)

The following function, displayed in Figure 14, will be used to test our algorithms:

\begin{equation}
f_2(x,y) = \begin{cases} 
\frac{1}{3} (1 - x^3) + \frac{1}{2} (xy - y^2) & \text{if } \sqrt{(x - \frac{1}{2})^2 + (y - \frac{1}{2})^2} < \frac{1}{4} \\
\frac{2}{5} \left( \frac{2}{5} (x^2 + y^2) \right) & \text{if } \sqrt{x^2 + y^2} < \frac{1}{4} + \frac{1}{20} \sin(\arctan(y/x)) \\
\frac{1}{4} (x^4 - 2x^2 + 1) (y^4 - 2y^2 + 1) & \text{otherwise.}
\end{cases}
\end{equation}

We seek to recover the jump function approximation, defined in two dimensions as

\begin{equation}
[f](x,y) := \sum_{j=1}^{M} [f](P_j) \chi_{P_j}(x,y),
\end{equation}

where the \( x \) and \( y \) components of each discontinuity \( P_j, j = 1, \cdots, M \), are given by \((\xi, \eta)\), and \( \chi_{P_j}(x,y) \) is the two dimensional extension of (3), which has value 1 at each cell containing \( P_j \) and 0 everywhere else. As in the one dimensional case, we observe that there are only a sparse number of nonzero coefficients in (30).

We wish to recover (30) from (28) at a finite set of uniform grid points, \((x_n, y_m)\), for \( n, m = 1, \cdots, 2N+1 \). Determining \([f]\) is considerably more difficult in two dimensions, because its non-zero values depend upon how each edge is approached. Since we are essentially interested in generating an edge map on a Cartesian grid, we will say that each jump value, \([f](\xi, \eta)\), is approximated as the difference of \(f\) in \( x \) and \( y \) across an internal boundary curve at the point \((\xi, \eta)\). This interpretation allows us to define the two dimensional concentration factor edge detection method as [11]:

\begin{equation}
S_N^\sigma[f](x,y) = - \sum_{l=-N \tau \neq 0}^{N} \sum_{k=-N \neq 0}^{N} \hat{f}_{k,l} \text{sgn}(k) \text{sgn}(l) \sigma\left(\frac{|k|}{N}\right) \sigma\left(\frac{|l|}{N}\right)e^{i\pi(kx+ly)}.
\end{equation}

Using (31) has some inherent limitations, however. In particular, if the underlying image has an edge that consists of a straight line in either coordinate direction, the method would not “see” it in that direction, as (31) would return zero. As discussed in [11], one option to improve the performance in these cases is to apply (31) twice, once in the Cartesian coordinate system and once again after rotating the image, which will detect edges that align with the coordinate axes. However, for generating an edge map on a Cartesian
grid, we have found that applying (7), (24), and Algorithm 3 dimension by dimension, is efficient and robust. To this end we write the dimension by dimension equivalents of (7) as

\[ S_{\sigma}^N[f](x, y_m) = i \sum_{k=-N}^{N} \sum_{l=-N, l \neq 0}^{N} \hat{f}_{k,l} \text{sgn}(k)\sigma \left( \frac{|k|}{N} \right) e^{i \pi (kx + ly_m)}, \]

for each \( x_n = -1 + \frac{2(n-1)}{2N} \), and \( y_m = -1 + \frac{2(m-1)}{2N} \), \( 1 \leq n, m \leq 2N + 1 \). We compute (32a) and (32b) on \( x_n \) and \( y_m \) respectively. For algorithmic purposes, we will use \( S_{\sigma}^N[F]_{m,n} \) to represent (32a) applied row by row or in the \( \vec{x} \) direction and correspondingly \( S_{\sigma}^N[F]_{m,n} \) to represent (32b) applied column by column or in the \( \vec{y} \) direction. We can then combine these results to form

\[ S_{\sigma}^C[N][F]_{m,n} = \max \left( \left| S_{\sigma}^N[F]_{m,n} \right|, \left| S_{\sigma}^N[F]_{m,n} \right| \right). \]

As in Algorithm 1, an edge map, \( S_{\sigma}^{PP}[F] \), can be obtained by thresholding \( S_{\sigma}^C[N][F] \) as demonstrated below.

Algorithm 4 Two dimensional edge map generation using thresholding

Input: \( S_{\sigma}^C[N][F] \).

- for \( j = 1, \ldots, 2N + 1 \)
  1. let \( \vec{u} \) be the thresholded jump map generated by Algorithm 1 using \( S_{\sigma}^C[N][F]_{j,1...2N+1} \)
  2. for \( k = 1, \ldots, 2N + 1 \)
     (a) let \( \vec{v} \) be the thresholded jump map generated by Algorithm 1 using \( S_{\sigma}^C[N][F]_{1...2N+1,k} \)
     (b) if \( |\vec{v}_j| > |\vec{u}_k| \)
         i. \( S_{\sigma}^{PP}[F]_{j,k} = \vec{v}_j \)
     (c) else
         i. \( S_{\sigma}^{PP}[F]_{j,k} = \vec{u}_k \)
  3. end for
- end for

\[ S_{\sigma}^{C,N}[F_2] \] and a cross section at \( S_{\sigma}^{C,N}[F_2]_{m,N+1} \), \( m = 1, \ldots, 2N + 1 \).

\[ \text{Fig. 15.} \]

---

\( ^5 \)We also performed our numerical tests using the 2-norm \( S_{\sigma}^C[N][F]_{m,n} = \sqrt{\left( S_{\sigma}^{x,N}[F]_{m,n} \right)^2 + \left( S_{\sigma}^{y,N}[F]_{m,n} \right)^2} \), but observed no noticeable difference in the results.
Figure 15 and 16 show $S_{C,N}^G[F_2]$ and the thresholded result respectively. Thresholding is difficult even for test case 1, and becomes less effective with fewer available coefficients or more noise, as oscillations appear both at nearby edge points as well as in smooth regions.

As in the one dimensional case, the primary advantage of the minmod algorithm, (22), is that it reduces oscillations that result from the so called “side lobes” of any particular concentration factor. Indeed, the internal oscillations seen in test case 1 of Figure 16 are greatly reduced. However, this advantage becomes negligible as the data are increasingly noisy or under-sampled. The sparsity enforcing edge detection method, (23), is also readily extended to multiple dimensions using a dimension by dimension approach, and the results are analogous to the one dimensional case. Since neither offers a significant improvement over the standard concentration factor edge detection method, the results are not included here.

4.1. Variance of jump function responses. Analogously to the one dimensional case in (24), we define the two dimensional variance on the set of two dimensional jump function approximations, $E = \{E^1F, \cdots, E^\nu F\}$, where each $E^jF$, $j = 1, \cdots, \nu$, is calculated on a set of points $(x_n, y_m)$, with $1 \leq m, n \leq 2N + 1$, as

$$V(E)_{m,n} = \frac{1}{\dim(E)} \sum_{E^j \in E} (E^jF_{m,n} - \frac{1}{\dim(E)} \sum_{E^k \in E} E^kF_{m,n})^2.$$  

Here

$E = \{S_{C,N}^G[F]\}_{\sigma_k \in G}.$

Algorithm 5 describes how to generate a two dimensional edge map from (33). As with Algorithm 2, there is a need for thresholding. We now choose $\epsilon = \epsilon(c)$ such that

$$\dim (\{V(E)_{m,n} : V(E)_{m,n} < \epsilon, 1 \leq m, n \leq 2N+1\}) = (1 - c)(2N + 1)^2.$$  

We note that Algorithm 5 actually returns an indicator of relative edge heights rather than a pair of directional jump values. These can be determined if necessary by referencing $S_{C,N}^G[F]$ and $S_{G,N}^G[F]$ at edge points.

The variance of jump response, $V(E)$, using the concentration factor set $G = \{\sigma_G, \sigma_p, \sigma_E, \sigma_E^2\}$, is shown in Figure 17 with the results from Algorithm 5 shown in Figure 18. The high variance regions in the neighborhood of edges are well defined with the full set of noise free Fourier coefficients as well as when those coefficients are sub-sampled. However, it is evident that large variance values appear in smooth regions in the presence of noise. Thresholding $V(E)$ in Algorithm 2 causes some false jump regions to be identified and some true jump regions to be discarded, leading to false positives and missed edges in Algorithm 5 (Figure 18(c) and (d)).

4.2. Determining edges by regularized reconstruction. Algorithm 5 works well when the behavior of the variance can properly identify local jump regions. As in the one dimensional case, the variance of the regularized reconstruction method, (26), augments the edge detection criterion further by recognizing the
Algorithm 5 Two dimensional edge map generation

Input: $S_{C,N}^\sigma[F], V(E).

\begin{itemize}
\item for $j = 1, \ldots, 2N + 1$
  \begin{itemize}
  \item $\vec{u}_j = V(E)_{j,1\ldots2N+1}$
  \item $\vec{w}_j = S_{C,N}^\sigma[F]_{j,1\ldots2N+1}$
  \item determine $B_{\ell}$ from $\vec{u}$ by Algorithm 2
  \item determine $(E_X)_{j,1\ldots2N+1}$ from $B_{\ell}$ and $\vec{w}$ by Algorithm 3
  \end{itemize}
\item end for
\item for $j = 1, \ldots, 2N + 1$
  \begin{itemize}
  \item $\vec{u}_j = V(E)_{1\ldots2N+1,j}$
  \item $\vec{w}_j = S_{C,N}^\sigma[F]_{1\ldots2N+1,j}$
  \item determine $B_{\ell}$ from $\vec{u}$ by Algorithm 2
  \item determine $(E_Y)_{1\ldots2N+1,j}$ from $B_{\ell}$ and $\vec{w}$ by Algorithm 3
  \end{itemize}
\item end for
\item for $j = 1, \ldots, 2N + 1$
  \begin{itemize}
  \item for $k = 1, \ldots, 2N + 1$
    \begin{itemize}
    \item $E_{j,k} = |(E_X)_{j,k}| + |(E_Y)_{j,k}|$
    \end{itemize}
  \item end for
  \end{itemize}
\item end for
\end{itemize}
The results of applying Algorithm 5 to $V(E)$ using $G = \{\sigma_G, \sigma_{P_1}, \sigma_{E_2}, \sigma_{E_3}\}$, maximum distance between peaks, $\delta = 7$, threshold $c = 15/16$.

$V(Q)$ using $Q = 40$ and a cross section $V(Q)_{m,N+1}$, $m = 1, \ldots, 2N+1$.

shown in Figure 20. The identification of false edges in smooth but variable regions results from a lack of resolution in the given data. Thresholding reduces the false detects but also discards some true edges. Numerical experiments indicate that increasing the size of the initial data set $U$ defined in Section 2.4 for the polynomial annihilation transform of order 2 improves the performance of the algorithm. Figure 21 displays the results of each algorithm for $N = 128$ with the remaining parameters in Table 2 held constant. The threshold constant for Algorithm 1, $c$, is adjusted to reflect the increase in resolution. Figure 22 shows how each algorithm is affected when the initial data set $U$ is increased to $N = 128$, but the sample size for each test, $\gamma(2N + 1)^2$ is held fixed (that is, to the case when $N = 64$). Finally, Figure 23 demonstrates that for $N = 128$, similar results can be obtained with noisier data ($SNR = 8$) and more sub-sampling ($\gamma = .2$).

Figure 24 compares the concentration factor edge detection method to our variance signature technique quantitatively for each of the four test cases, as well as the examples from Figures 21, 22, and 23, by measuring their ability to properly classify points as edges. The figure displays the fraction of edge cells that were correctly detected, the count of false positives, and the fraction of false positives that lie in the 8-connected neighborhood of true edge cells. Observe that when the variance for the regularized reconstruction is used, most of the false positives are near the true edges.

5. Concluding Remarks. Assuming sufficient resolution, the concentration factor edge detection method, (7), effectively recovers the edges of a piecewise smooth function from its first $2N+1$ noiseless Fourier coefficients. The method is subject to many false positives if the data are noisy or if values from the set of $2N+1$ Fourier coefficients are missing or otherwise unusable. Thresholding helps, but the threshold value is problem dependent and is therefore not very robust to added noise or reduced sample size. Moreover, when jumps are small, thresholding may eliminate true edges. Processing the results via the minmod algorithm, (22), helps to eliminate some false edges that are artifacts of oscillatory jump function responses, but is still not robust when data are noisy or under-sampled. Finally, while the sparsity promoting $l^1$ regul-
The results of applying Algorithm 5 to $V(Q)$ with $Q = 30$, maximum distance between peaks, $\delta = 7$, threshold $c = 15/16$.

Comparison of Algorithm 5 results using $N=128$, SNR=12.5dB, $\gamma = 0.75$, $\beta = 0.3$, $\delta = 7$, threshold $c = 31/32$.

Our method is robust to increasing levels of noise and sub-sampling, and is efficient since each test can be performed in parallel. Future investigations will include a more rigorous study of parameter choices. Ultimately these will be application dependent, and in particular it is important that our algorithm is robust to a variety of input data sets. Future investigations will also include the case where the Fourier data are sampled non-uniformly, which occurs in applications such as propeller and parallel MRI. We believe that in these cases our algorithm will further demonstrate its computational efficiency, since in the non-uniform
Fig. 22. Comparison of Algorithm 5 results using N=128, SNR=12.5dB, γ = \frac{9}{16}, β = 0.09, δ = 7, threshold c = 31/32.

Fig. 23. Comparison of Algorithm 5 results using N=128, SNR=8dB, γ = \frac{1}{5}, β = 0.09 δ = 7, threshold c = 31/32.

In this case, the FFT is not as readily used, so reducing the amount of data needed becomes more critical.

REFERENCES

(a) Correct detections/edge cells

(b) False positives

(c) False positives adjacent to edge cells/false positives

Fig. 24. Comparison of methods using Algorithm 4 on $S_{C,N}^{G}$ and $S_{C,N}^{M}$ and Algorithm 5 on $V(E)$ and $V(Q)$ for the test cases in Table 2 and Figures 21, 22, and 23.


