COMMUNICATIONS IN COMPUTATIONAL PHYSICS
Vol. **5**, No. 2-4, pp. 326-349

Commun. Comput. Phys.
February 2009

Segmentation of Images from Fourier Spectral Data

Anne Gelb*and Dennis Cates

Department of Mathematics and Statistics, Arizona State University, Tempe, Arizona 85287 USA.

Received 24 September 2007; Accepted (in revised version) 3 November 2007

Available online 1 August 2008

Abstract. This paper designs a segmentation method for an image based on its Fourier spectral data. An edge map is generated directly from the Fourier coefficients without first reconstructing the image in pixelated form. Consequently the internal boundaries of the edge map are not blurred by any (filtered) Fourier reconstruction. The edge map is then processed with an edge linking segmentation algorithm. We include examples from magnetic resonance imaging (MRI). Our results illustrate some potential benefits of using high order methods in imaging.

AMS subject classifications: 42A50, 65D17, 68U10

Key words: Fourier coefficients, edge detection, segmentation.

1 Introduction

Fourier spectral data is often the source of image information. For example, magnetic resonance imaging (MRI) and synthetic aperture radar (SAR) sensors measure the values of the Fourier coefficients of an image. Techniques such as the filtered Fourier reconstruction are then used to create the image. In many instances, clinicians and diagnosticians are not interested in the detailed structure of the image, but rather the shapes that outline a specific region of interest. Typically in such cases the Fourier coefficients are used to reconstruct the image at specified pixel points so that a standard segmentation algorithm, e.g. [7,19,22,23,25,26], can be applied. There are clear downsides to this approach, however. Most notably, since the image to be segmented is necessarily only piecewise smooth, the reconstruction is polluted by the Gibbs ringing artifact at the jump discontinuities (edges) of the image, [6,8,17,21], making it difficult to determine where the boundaries of internal regions lie. Smoothing filters are used to reduce the oscillations, but have the undesirable consequence of "blurring" the edges over several pixel points. In [2], the

*Corresponding author. *Email addresses:* ag@math.la.asu.edu (A. Gelb), mayfairfarm2@cox.net (D. Cates)

Gegenbauer reconstruction algorithm, [18], was adapted to resolve a brain MRI free from Gibbs ringing and without blurring the boundaries of each tissue region. Consequently, standard segmentation algorithms used to extract features of interest returned dramatically improved results, [1]. Unfortunately high order reconstruction methods, such as the Gegenbauer reconstruction method, can be cost prohibitive for large data sets. This investigation therefore adopts a different strategy based on the following observations: (i) image reconstruction is not necessary for region extraction; (ii) without using costly reconstruction algorithms, converting the Fourier coefficients into a pixelated image either produces Gibbs oscillations that clutter the edges, or damping, which may blur the edges beyond recognition; and (iii) vital information about the internal boundaries of an image stored in the Fourier data can be easily extracted. Hence we seek to develop a segmentation algorithm that directly uses Fourier information to extract the internal edges of an image. We emphasize the distinction between our proposed method, which starts with the Fourier coefficients of an image, and standard segmentation algorithms, which start with pixelated data.

Once a high quality edge map of the image is generated, we design a segmentation algorithm that links the points on the edge map to create closed contour regions. Specifically, the algorithm produces sequences of ordered pairs which in turn can be used as an initial guess in algorithms that parameterize contour regions for each feature of interest, [27]. We note that the intensity values of our edge map approximate the magnitude of the jump discontinuities of the image, and *not* the underlying image values at the internal boundaries. Although we are unable to determine a-priori error estimates, our examples demonstrate the techniques proposed here provide a good starting point for designing fast segmentation algorithms from Fourier data, and therefore illustrate the benefits of using high order methods in imaging.

This paper is organized as follows. In Section 2 we review the concentration method developed in [13–15] which locates jump discontinuities (edges) in piecewise smooth functions directly from Fourier spectral data. The method is extended to two dimensions in Section 3 to produce the edge map of an image. In Section 4 we present our segmentation algorithm that generates (closed) parameterized contours from the edge map. The output can now be used to initiate contour minimization routines which can accurately estimate and extract particular features of interest, [9, 27]. We summarize our results in Section 5 and discuss possible future applications.

2 Edge detection from Fourier spectral data

Suppose we are given the Fourier spectral data of an image. In order to extract any particular feature of interest, we must first be able to visualize its internal boundaries. One way to do this is to first reconstruct the image on a prescribed set of pixels, for instance by using filtered Fourier algorithms, [21]. However, as stated in the introduction, this type of projection often causes blurring at the internal boundaries, making segmentation

very difficult. High order reconstruction methods that do not blur the edges, [2, 18], are too expensive to implement on large data sets. As an alternative we will construct an edge map of the image directly from its Fourier coefficients, and use this information to segment the image in a cost efficient manner. The method is described below.

2.1 The concentration method

Consider a 2π periodic and piecewise smooth function $f(x)$ on an interval $[-\pi, \pi)$. For ease of presentation we assume there is a simple jump discontinuity at $x=\xi$, although the analysis holds for any finite number of jumps. Suppose that the one sided k^{th} derivatives, $f^{(k)}(x)$, exist at each jump discontinuity location for $k=0,1,\cdots,p$ and $p>0$. We define the corresponding jump function of $f(x)$ as

$$[f](x) := f(x^+) - f(x^-), \tag{2.1}$$

where $f(x^+)$ and $f(x^-)$ are presumably well defined right and left side limits of $f(x)$ at x respectively. If $f(x)$ is continuous at a point $x=\xi$, then the value of the jump function is zero. On the other hand, if $f(x)$ experiences a jump discontinuity at $x=\xi$, then the jump function value is $[f](\xi)$.

Assume we are given the complex Fourier coefficients

$$\hat{f}_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-ikx}\,dx, \qquad \text{for } k=-N,\cdots,N. \tag{2.2}$$

Information about the edges of $f(x)$ can be obtained directly from the Fourier coefficients, e.g. [3–5, 11, 13, 20]. The concentration method, [13–15], has some advantages over other edge detection algorithms. In particular, it is less sensitive to errors in the Fourier data and does not require a priori knowledge of the number of jump locations in the underlying function. The concentration method takes the form

$$\tilde{S}_N^{\sigma}[f](x) = i\sum_{k=-N}^{N} sgn(k)\sigma(\frac{|k|}{N})\hat{f}_k e^{ikx}. \tag{2.3}$$

Here

$$\sigma(\eta) = \sigma\left(\frac{|k|}{N}\right), \quad \eta\in(0,1]$$

is called a "concentration factor" since it forces (2.3) to concentrate at the discontinuities of $f(x)$. In [13] it was shown that admissible concentration factors, satisfying the properties

$$i)\,\frac{\sigma(\eta)}{\eta}\in C^2(0,1), \quad \text{and} \quad ii)\int_0^1 \frac{\sigma(\eta)}{\eta}\,d\eta = \pi, \tag{2.4}$$

yield the concentration property

$$\tilde{S}_N^{\sigma}[f](x) \longrightarrow [f](x) \qquad \text{as} \quad N\to\infty.$$

Several types of admissible concentration factors were studied [13–15] [†]. We consider three concentration factors described in [13, 14] given by

$$\sigma_{Gibbs}(\eta) = \frac{\pi \sin \pi\eta}{Si(\pi)}, \tag{2.5}$$

$$\sigma_{Poly}(\eta) = \pi\eta, \tag{2.6}$$

$$\sigma_{Exp}(\eta) = \gamma\eta \exp\Big(\frac{1}{\alpha\eta(\eta-1)}\Big), \tag{2.7}$$

where

$$\gamma = \frac{\pi}{\int_{\epsilon}^{1-\epsilon} \exp\big(\frac{1}{\alpha\tau(\tau-1)}\big) d\tau}$$

normalizes $\sigma_{Exp}(\eta)$. In our examples we chose $\alpha = 6$ and $\epsilon = \frac{1}{N}$.

The following example demonstrates the ability of the concentration method to detect jump discontinuities. Observe the close proximity of some jump locations.

Example 2.1.

$$f(x) = \begin{cases} 0 & -\pi \le x < -3\pi/4, \\ 1+\cos 3x & -3\pi/4 \le x < -\pi/2, \\ 0 & -\pi/2 \le x < -\pi/4, \\ 1-2\cos(5x+3\pi/2) & -\pi/4 \le x < \pi/8, \\ 0 & 0 \le x < 3\pi/8, \\ 2\cos(x-3\pi/4)-3 & 3\pi/8 \le x < 7\pi/8, \\ 0 & 7\pi/8 \le x \le \pi. \end{cases}$$

Fig. 1(a)-(c) illustrates that the concentration method, (2.3), does indeed recover the corresponding jump function, $[f](x)$, of Example 2.1. The high order exponential concentration factor, depicted in Fig. 1(c), produces very fast convergence to zero away from the jump discontinuities. Method induced oscillations near the edges are particularly prominent in this case, however.

2.2 Enhancing the concentration method

As is evident from Fig. 1, each concentration factor yields a different convergence rate to zero away from the jump discontinuities. They also produce oscillations that vary in sign and magnitude near the jumps. While these method induced oscillations are in themselves undesirable, the variation they produce can be exploited to reduce the overall impact of the oscillations. In fact, in [15] it was shown that by comparing the results of (2.3) using several different concentration factors, the artificial oscillations around the

[†]We direct readers to [13–15] for both a heuristic description of the concentration factors in (2.5) -(2.7) as well as the convergence analysis of the concentration method.

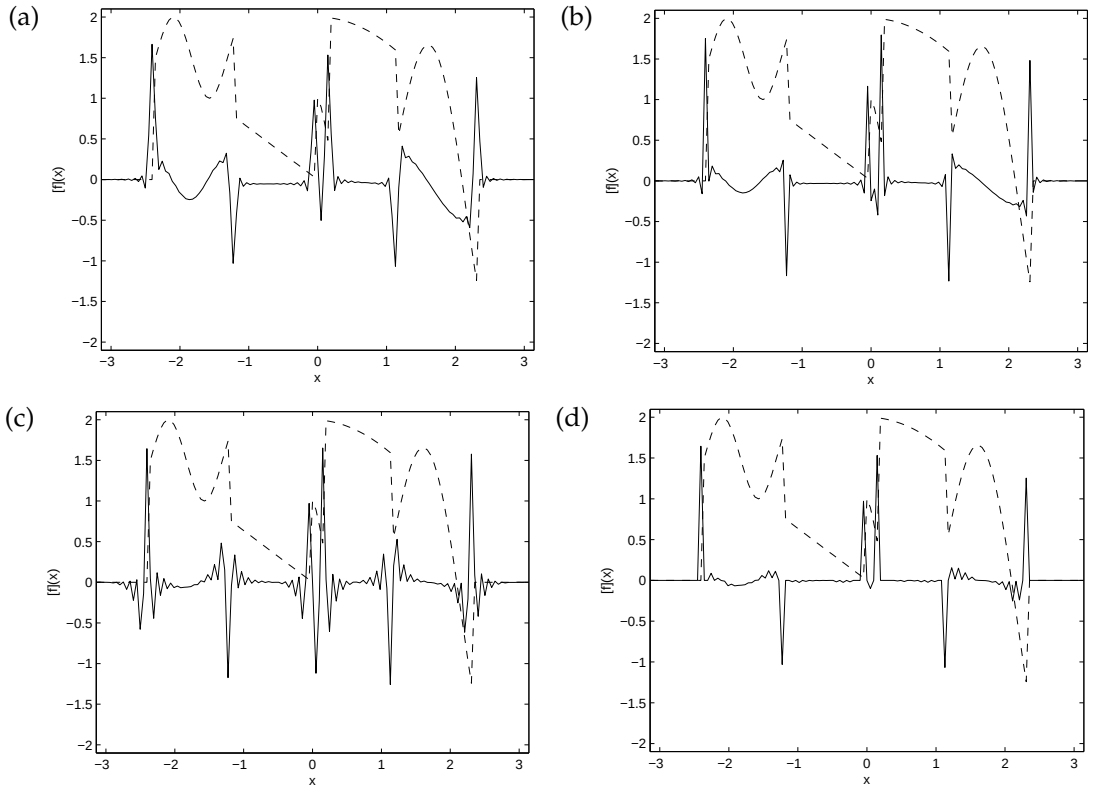


Figure 1: The concentration method applied to Example 2.1 for $N=64$ using (a) $\sigma_{Gibbs}(|k|/N)$, (b) $\sigma_{Poly}(|k|/N)$, and (c) $\sigma_{Exp}(|k|/N)$. (d) The *minmod* algorithm, (2.9), using concentration factors $\sigma_{Gibbs}(|k|/N)$, $\sigma_{Poly}(|k|/N)$, and $\sigma_{Exp}(|k|/N)$. The underlying function is dashed and the jump function approximation is solid.

jump discontinuities can be reduced. The method still converges rapidly to zero away from them. This was accomplished by applying the *minmod* function, given by

$$\text{minmod}(f_1,\cdots,f_j):=\begin{cases}\min(f_1,\cdots,f_j), & \text{if } f_1,\cdots,f_j>0,\\ \max(f_1,\cdots,f_j), & \text{if } f_1,\cdots,f_j<0,\\ 0, & \text{otherwise.}\end{cases} \tag{2.8}$$

It is pointed out that the minmod idea was also developed in this context in [5]. As a result we have what we will herein call the concentration method,

$$\tilde{E}_N^{\sigma}[f](x):=\text{minmod}(\tilde{S}_N^{\sigma_1}[f](x),\cdots,\tilde{S}_N^{\sigma_j}[f](x)), \tag{2.9}$$

where $\sigma_1,\cdots,\sigma_j$ are j admissible concentration factors. Fig. 1(d) shows the results of (2.9) on Example 2.1. Although the oscillations are significantly reduced, some outside thresholding based on the original underlying function is still needed to completely remove the remaining oscillations. The reliance on outside thresholding can be further reduced by

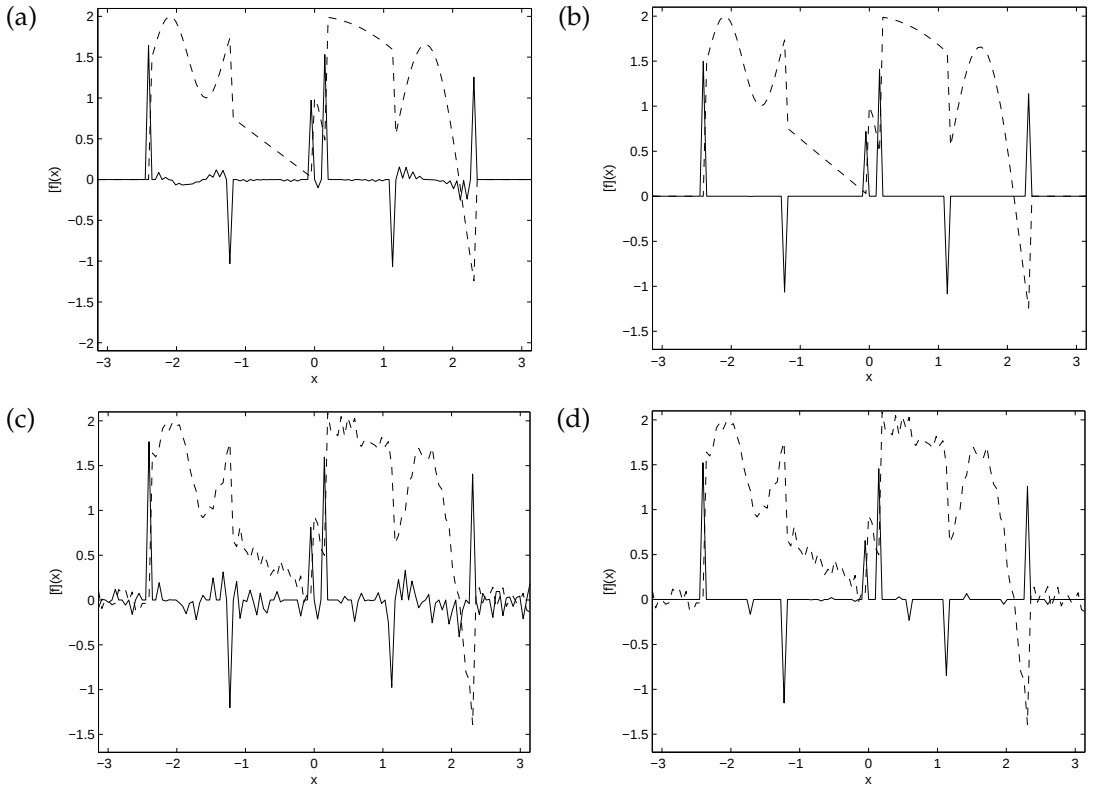


Figure 2: The concentration method applied to Example 2.1 for $N=64$. In (a) and (b) the image contains no noise. In (c) and (d) uniformly distributed random noise of range .15 has been added to the underlying function. (a) and (c): The results when the *minmod* algorithm is applied using concentration factors σ_{Gibbs}, σ_{Poly}, and σ_{Exp}. (b) and (d) The results when their corresponding matched waveform concentration factors σ_{match_1} and σ_{match_2} are also used. The underlying function is dashed and the jump function approximation is solid.

applying the matched waveform concentration factors, [12], given as

$$\sigma_{match_1}\Big(\frac{|k|}{N}\Big)=\frac{\pi}{k}\Big(\sigma\big(\frac{|k|}{N}\big)\Big)^2\Big/\sum_{l=1}^{N}\Big(\frac{\sigma\big(\frac{l}{N}\big)}{l}\Big)^2, \tag{2.10}$$

$$\sigma_{match_2}\Big(\frac{|k|}{N}\Big)=\pi k\Big(\sigma\Big(\frac{|k|}{N}\Big)\Big)^2\Big/\sum_{j=1}^{N}\Big(\sigma\Big(\frac{j}{N}\Big)\Big)^2, \tag{2.11}$$

where $\sigma(|k|/N)$ is any admissible concentration factor‡. Fig. 2 shows the application of the concentration method on Example 2.1 using (2.10) and (2.11). White uniform noise with zero mean is added to the underlying Fourier coefficients in Fig. 2(c) and (d). It was

‡Note that $\sigma_{match_2}(\eta)$ is not admissible as it is associated with a family of edge detectors that converge to the zero crossing approximation of the jump function rather than the jump function itself. However, its inclusion is beneficial since the local support of the corresponding jump function is much reduced, making it easier to distinguish closely spaced edges. See [12] for details.

also shown in [12] that the variance is always reduced when the refined concentration factors, (2.10) and (2.11), are included.

3 Edge map generation

Now assume we are given two dimensional Fourier spectral data. In this section we generate an edge map which we will later use to extract contour regions of interest.

3.1 Edge detection in two dimensions

The two dimensional edge map of an image is determined by combining the results from (2.9) in each perpendicular direction. We assume that the Fourier spectral data is known (as is the case in MRI, [21]) and is given by

$$\hat{f}_{k,l}=\frac{1}{4\pi^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}f(x,y)e^{-ikx}e^{-ily}dxdy, \tag{3.1}$$

for $k,l=-N,\cdots,N$. We then compute the Fourier coefficients for each fixed slice in the opposite Cartesian direction as

$$\tilde{f}_k(y_j)=\sum_{l=-N}^{N}\hat{f}_{k,l}e^{ily_j} \quad \text{and} \quad \tilde{f}_l(x_\nu)=\sum_{k=-N}^{N}\hat{f}_{k,l}e^{ikx_\nu}, \tag{3.2}$$

for

$$y_j=-\pi+\frac{\pi j}{N}, \quad x_\nu=-\pi+\frac{\pi\nu}{N},$$

with $j,\nu=0,\cdots,2N$. We apply the concentration method (2.9) directly on (3.2) to generate two edge maps

$$\tilde{E}_N^\sigma[f](x,y_j) \qquad \text{and} \qquad \tilde{E}_N^\sigma[f](x_\nu,y). \tag{3.3}$$

Although these maps can be generated for any x and y, we use $x=x_{\nu+1/2}$ and $y=y_{j+1/2}$ for $\nu,j=0,\cdots,2N-1$, to uniformly cover both dimensions. Fig. 3 shows how the edge map is constructed for a simple binary circle test pattern with $N=8$. The physical image is displayed just for reference, as we are given only Fourier spectral data. The small filled squares in Fig. 3 are the grid points populated with $\tilde{E}_N^\sigma[f](x,y_j)$ (horizontal scan), and the larger open squares in the diagram are the grid points populated with $\tilde{E}_N^\sigma[f](x_\nu,y)$ (vertical scan). The grid points marked by $+$ in Fig. 3 are populated later in Section 3.3 to "strengthen" the edge map

Note that since the one dimensional approximations capture the sign of the jump discontinuity, they contain both positive and negative values. However, no knowledge of the jump direction can be made when converting this information to two dimensions. A specific example of this is illustrated in the lower left portion of Fig. 3(b). Here the horizontal scans (arbitrarily chosen to progress from left to right) find positive jumps,

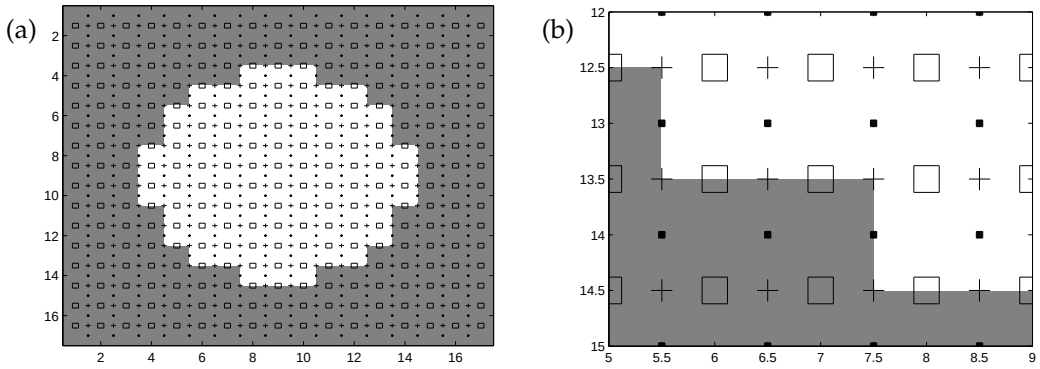


Figure 3: Simple circle test pattern illustrating the manner in which the edge map is constructed from (3.3). (a) The entire image. (b) A close up of one area. Here $N=8$.

but the vertical scans (arbitrarily chosen to progress from top to bottom) find negative jumps at the same boundary intersection. Therefore we only record the absolute values from each one dimensional scan.

3.2 Edge map noise thresholding

Before proceeding further, we must reduce the clutter in the edge map created by noise. For MRI processing, it is reasonable to assume that no edge information exists near the boundary of the image. Hence any non-zero value (or jump value) in this region is strictly due to noise. We thus define the noise threshold, *noisethreshold*, as the maximum (reconstructed) value in this region. Any jump values less than *noisethreshold* are automatically set to zero. This has the added benefit of reducing the effects of the oscillations close to the jump discontinuities that are inherent to the concentration method (see Fig. 2).

3.3 Edge map strengthening

We now "strengthen" the edge map by determining where edges may be forming perpendicular to the one dimensional scan directions (marked by $+$ in Fig. 3). An edge perpendicular to the horizontal scan direction (in which $\tilde{E}_N^\sigma[f](x,y_j)$ is being generated) is determined to exist along the line between $(x_{\nu+1/2},y_j)$ and $(x_{\nu+1/2},y_{j+1})$ if the values of the edge map at those positions are both nonzero. Hence we populate the point $(x_{\nu+1/2},y_{j+1/2})$ with a nonzero value. Edges perpendicular to the vertical scan direction are similarly determined. In this way, the intermediate grid points (marked by $+$) are populated with values that indicate whether or not two neighboring values constitute a probable edge. Algorithm 3.1 is used to evaluate $\tilde{E}_N^\sigma[f](x_{\nu+1/2},y_{j+1/2})$ for $j,\nu=0,\cdots,2N-1$.

Our edge map is now defined by

$$\tilde{E}_N^\sigma[f](X), \tag{3.4}$$

Algorithm 3.1: Edge Map Strengthening

- If

$$\min(\tilde{E}_N^\sigma[f](x_\nu,y_{j+\frac{1}{2}}),\tilde{E}_N^\sigma[f](x_{\nu+1},y_{j+\frac{1}{2}}))>0,$$
$$\min(\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_j)),\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_{j+1})=0,$$

  then $\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_{j+\frac{1}{2}})=\frac{1}{2}(\tilde{E}_N^\sigma[f](x_\nu,y_{j+\frac{1}{2}})+\tilde{E}_N^\sigma[f](x_{\nu+1},y_{j+\frac{1}{2}}))$.
- If

$$\min(\tilde{E}_N^\sigma[f](x_\nu,y_{j+\frac{1}{2}}),\tilde{E}_N^\sigma[f](x_{\nu+1},y_{j+\frac{1}{2}}))=0,$$
$$\min(\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_j),\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_{j+1}))>0,$$

  then $\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_{j+\frac{1}{2}})=\frac{1}{2}(\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_j)+\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_{j+1}))$.
- If

$$\min(\tilde{E}_N^\sigma[f](x_\nu,y_{j+\frac{1}{2}}),\tilde{E}_N^\sigma[f](x_{\nu+1},y_{j+\frac{1}{2}}))>0,$$
$$\min(\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_j),\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_{j+1}))>0,$$

  then

$$\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_{j+\frac{1}{2}})=\max\Big\{\frac{1}{2}(\tilde{E}_N^\sigma[f](x_\nu,y_{j+\frac{1}{2}})+\tilde{E}_N^\sigma[f](x_{\nu+1},y_{j+\frac{1}{2}})),$$
$$\frac{1}{2}(\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_j)+\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_{j+1}))\Big\}.$$

- If

$$\min(\tilde{E}_N^\sigma[f](x_\nu,y_{j+\frac{1}{2}}),\tilde{E}_N^\sigma[f](x_{\nu+1},y_{j+\frac{1}{2}}))=0,$$
$$\min(\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_j),\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_{j+1}))=0,$$

  then $\tilde{E}_N^\sigma[f](x_{\nu+\frac{1}{2}},y_{j+\frac{1}{2}})=0$.

where $X=(x,y)$. Note that (3.4) has dimension $(4N+1)\times(4N+1)$.

Observe that only the points $(x_{\nu+1/2},y_{j+1/2})$ (marked by $+$) are examined in this manner. Populating other intermediate points, such as $(x_{\nu+1},y_{j+1})$, would produce possible smearing, so that the segmentation algorithm might lose the ability to distinguish between closely spaced edges. Another benefit seen here is the somewhat "rounding out" of the edges. That is, if a sharp corner is developing at a grid point location, a zero is assigned to the relevant intermediate location because only one value of the edge map on either side of the corner would be non zero. This effect subsequently enhances the abil-

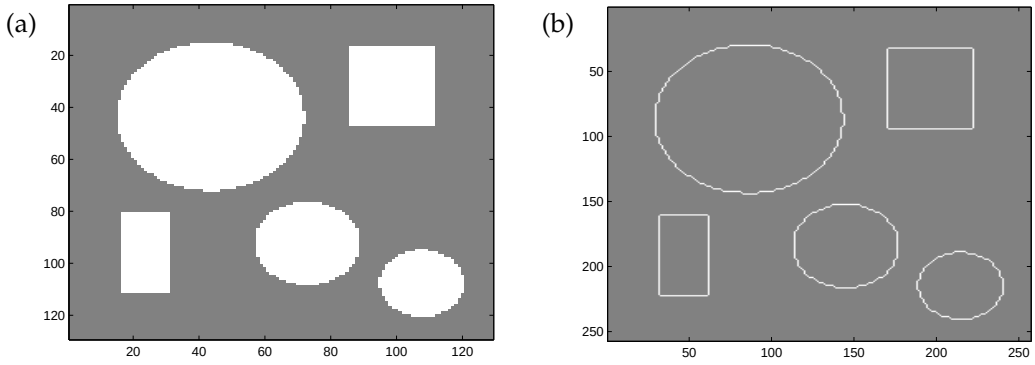


Figure 4: (a) Test Pattern A. (b) Edge map generated from 128×128 Fourier coefficients.

ity of the segmentation algorithm, described in Section 4, to accurately generate closed contours around features.

Fig. 4 displays the corresponding edge map of a binary piecewise constant image (Test Pattern A). Recall that only the Fourier coefficients (3.1) are used to produce the edge map. Here we have used the two dimensional FFT to artificially generate the Fourier coefficients of the image.

4 Segmentation algorithm

There are existing numerical algorithms that extract contour regions of images from an edge cloud and a good initial guess of each feature's shape, [9, 27]. With an edge map in hand, our goal is to create sequences of parameterized ordered pairs, $(x(t),y(t))$, and then generate each feature's shape in the image by attempting to close each sequence. Techniques such as contour following and edge linking, [19], have been used to segment images when the internal boundary *function values* are known. We follow these ideas, but modify them to be applicable when the internal boundary *jump values* are given instead. Generating closed contours from an edge map is difficult, and we must divide our work into features that we can close immediately and those that require additional processing.

4.1 Generation of sequences

The concentration method yields varying jump values associated with the strength of each edge. This distinguishes it from other edge detection methods, such as [7, 16, 19, 26], that typically produce a binary output as their edge map. We take advantage of the variation in our edge map construction, (3.4), by making the reasonable assumption that the internal boundary edges of an image will maintain similar strength locally and possibly around an entire feature. Our segmentation algorithm therefore adds the constraint of grouping edges of similar strength together.

We first divide the edge map into different jump levels where each level, JLT_i, is defined as

$$JLT_i = M - \left(\frac{M - noisethreshold}{\#levels}\right)(\#levels - i + 1), \tag{4.1}$$

for $i = 1, \cdots, \#levels$. Here $M = \max_X \tilde{E}_N^\sigma[f](X)$ in (3.4), and the lowest jump level threshold is *noisethreshold*, so that all edge map values falling below it are automatically discarded as noise. Every grid point on the edge map is now assigned the appropriate integer value jump level as

$$\tilde{E}_N^\sigma[f](X) = \max\left\{i : \tilde{E}_N^\sigma[f](X) > JLT_i\right\},$$

for $i = 1, \cdots, \#levels$. If

$$\tilde{E}_N^\sigma[f](X) < JLT_1,$$

we set

$$\tilde{E}_N^\sigma[f](X) = 0.$$

This generates a map of integer values ranging from zero to $\#levels$. We used $\#levels = 10$ in our experiments.

Assuming that many edges maintain a near constant jump magnitude around its contour, the segmentation algorithm attempts to link high jump magnitude edges first, and then works down to lower magnitude values. If an edge does in fact change its magnitude as its position is changed, causing it to fall to a different jump level, the algorithm will still be able to parameterize the contour at a later step.

The sequencing begins by first selecting the grid points with the highest jump level values. Since no history is associated with this first point, the direction in which to search for a second point within the same jump level is arbitrary, and is based solely on proximity. Once the second point has been located, a direction history is established. A contour following algorithm requires that we (i) minimize the kinks in the resulting edge and (ii) minimize the distance between adjacent points on an edge path. As is typical in path following algorithms, we construct a (weighted) average of the direction history to help determine subsequent grid point locations that minimize the angular direction change as the segmented edge is being generated. Specifically, we define the historical angular direction change as

$$\theta_{i+2} = \alpha\theta_{i+1} + \beta\theta_i, \tag{4.2}$$

$i = 1, \cdots,$. The sequence ends when the edge is segmented. We use $\theta_1 = \theta_2$ to initialize (4.2). The parameters α and β are user defined such that $\alpha + \beta = 1$, for example $\alpha = \beta = \frac{1}{2}$. Clearly, θ_{i+1} would be weighed more heavily if $\alpha > \beta$, however, no attempt was made here to optimize these parameters. The angular direction reference is arbitrary, but must be maintained within the range $-\pi \leq \theta < \pi$.

In addition to the above conditions, we impose the following restrictions to ensure a viable segmentation: (iii) single usage of an edge point within a jump level; (iv) not choosing edge points from lower jump levels; and (v) not considering edges that are "close" to a previously found closed contour.

These three additional requirements are used so that the edges stay on their intended paths and so that the path does not inadvertently or prematurely double back on itself. In particular, (v) helps to "thin" the edges, [19]. Of course it also means that less intense features might not be recovered if they are very near to stronger features. We note that the procedure does not limit itself to adjacent grid points, but may skip over a grid point or two if needed to meet these requirements.

We pause here to note the number of parameters already introduced in our segmentation algorithm. This is typical for contour following and edge linking techniques. While we can easily validate our results for simple contours, validation in (brain) MRI is usually done in the form of probability maps that determine the various tissue types throughout the image (see [1] for details). This paper aims to provide an algorithm to show that segmentation from a Fourier based edge map is indeed possible.

To advance the contour, all nearby grid points are assigned a priority value based on their proximity to the current ending contour grid point and the historical direction, θ_{i+2}, in (4.2). Fig. 5 displays the reference grid generated around the current contour ending grid point, labeled as P_{start}.

P_{23}	P_{24}	P_9	P_{10}	P_{11}
P_{22}	P_8	P_1	P_2	P_{12}
P_{21}	P_7	P_{start}	P_3	P_{13}
P_{20}	P_6	P_5	P_4	P_{14}
P_{19}	P_{18}	P_{17}	P_{16}	P_{15}

Figure 5: Orientation of nearby grid points about the current contour end grid point, labeled P_{start}.

The historical direction of the developing contour can be translated into the nearby reference grid point from which the contour is developing. As an example, if the segment is developing from below the current end point of the contour, P_{start}, then the historical direction is "upwards" with $\theta_{i+2} \approx \frac{\pi}{2}$ (see Fig. 5). In this case, the contour is developing from the grid point location P_5. Note that this does not mean that the previous grid point value in the parameterized list of the developing contour is necessarily P_5.

The nearby reference grid points are all ranked based on the developing direction of the contour. As an example, if the contour is developing from grid point P_5, then the priority list is

$$P_1, P_2, P_8, P_9, P_{10}, P_{24}, P_{11}, P_{23}, P_3, P_7, P_{12}, P_{22}. \tag{4.3}$$

All remaining nearby reference grid points are excluded from consideration in this case. Similar priority lists are determined for each different development direction at the beginning of the algorithm, which minimize proximity while maintaining smoothness. Furthermore, we reduce the possibility of a growing contour doubling back on itself prematurely.

We then examine the values of each point on the priority list in order. If the grid point under consideration satisfies (iii), (iv), and (v), it is attached to the end of the segment,

and the sequencing process continues by shifting the new end point to the center of the grid. In our example, P_1 would be tested first. If its value satisfies the criteria (the correct jump level, not already used, etc.), then it would be attached to the end of the segment, the search would end, and the process would begin again with P_1 as P_{start}. On the other hand, if its value does not satisfy the criteria, then P_2 would be checked, then P_8 and so on. As the edge contour develops, a parameterized list, $\big(x(t),y(t)\big)$, is being generated.

4.2 Closed contour segmentation

The sequencing routine continues until it cannot find another point on the priority list ((4.3) in our example) that satisfies the edge path criteria. We then terminate the contour and attempt to close it. However, requirement (iii) does not allow a grid point to be used twice in the same edge path. This, of course, creates a problem as there is then no way for the segment to close itself. It is handled as follows:

Let $\{X_1,\cdots,X_p\}$ denote the set of points that determines a completed edge path and define δ_1 to be the minimum closed contour length. From the starting path point X_1, we move along the path to the point X_r, which is the first point on the path such that

$$|X_r-X_1|\geq\delta_1.$$

We then measure the distances from each subsequent grid point on the edge path to the starting point $|X_j-X_1|$, $j=r+1,\cdots,p$, and define

$$X_m:=\{X\in\{X_r,\cdots,X_p\}:|X-X_1|=\min_{r<j\leq p}|X_j-X_1|\}. \tag{4.4}$$

If $|X_m-X_1|\leq\delta_2$, where δ_2 is the user prescribed maximum connection distance for a single segment, then the path is considered closed. We now proceed to close the contour: (1) The edge is terminated at X_m. (2) X_1 is added after X_m in the parameterized list to close the segment. (3) Any remaining points that are not part of the closed portion of the segment, $\{X_{m+1},\cdots,X_p\}$, are returned to the edge map to be reused by other possible contours. (4) The routine parades along the closed contour, $\{X_1,\cdots,X_m\}$, and removes all nearby points within a distance δ_3 (minimum distance between two contours) that are not part of the contour. These points are not considered again. These actions do a few things: First, they close segments. Second, they eliminate the possibility of double edges, that is they "thin" the edges. However, weaker edges within a distance δ_3 from a closed contour are lost. Finally, if a contour has inadvertently crossed over itself, this process may re-establish closed curves by pinching off the remainder of the contour for later use. Table 1 summarizes the segmentation parameters.

Once the routine is completed for a particular segment, we look for a new starting point within the same jump level, (that is not already a part of an existing contour), and begin the sequencing process again. The whole algorithm is repeated until no starting points remain within that jump level. All edge map values within a jump level that were not included in a closed path are then reset so that they can be reused by linking to grid

Table 1: Sequencing algorithm user defined parameter summary. The typical values are given as the number of pixels, where each pixel distance is π/N.

Name	Purpose	Typical Value
δ_1	Minimum closed contour length	4
δ_2	Maximum connection length for a single segment	2
δ_3	Minimum distance between two contours	1
δ_4	Minimum length of an admissible open segment	3
δ_5	Maximum connection length between two open segments	3
δ_6	Maximum length to close connected segments	4

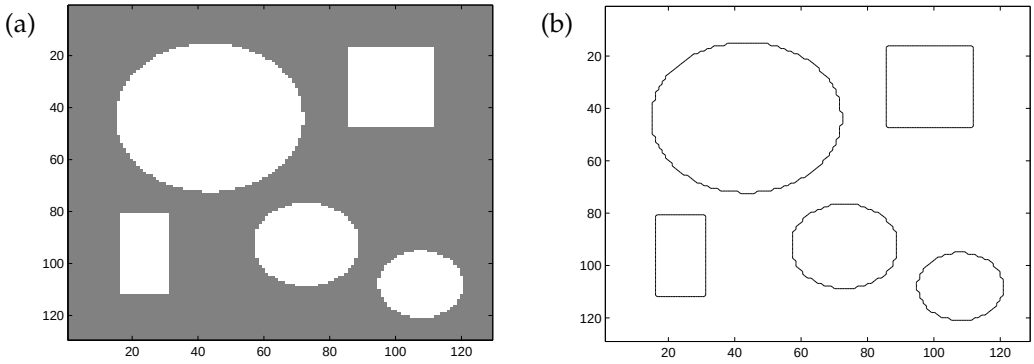


Figure 6: (a) Test Pattern A on $128X128$ grid points. (b) Segmentation results.

points at a lower jump level, JLT_{i-1} in (4.1). The procedure is then started again at the next lower jump level, and is repeated through all possible jump levels of the edge map.

Fig. 6 shows the segmented image of Test Pattern A. The edge map clearly defines the boundaries of every feature in the image, and the segmentation algorithm generates five separate parameterized lists of points in the order in which each contour progresses. Throughout this article we used a simple "connect-the-dots" plot routine, which linearly connects successive grid points, to illustrate the segmentation output.

Fig. 7 demonstrates the results of our segmentation algorithm on a more challenging binary image that includes snakelike patterns and closely spaced features with thinly defined edges. In this case the discrete Fourier coefficients are used to generate the edge map. Among other obstacles, notice how two neighboring features have similar edge directions. It is important that our segmentation algorithm not jump from one feature to the next. Several features turn sharply so that minimizing the kinks without following the other requirements would surely lead to incorrect segmentations. We include geo-

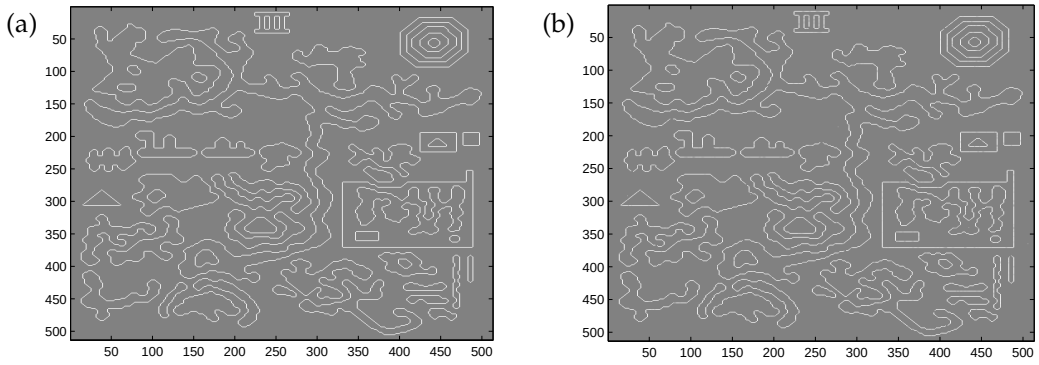


Figure 7: (a) Test Pattern B on 256×256 points. (b) Edge map generated when 20% uniformly distributed random noise added to the underlying data.

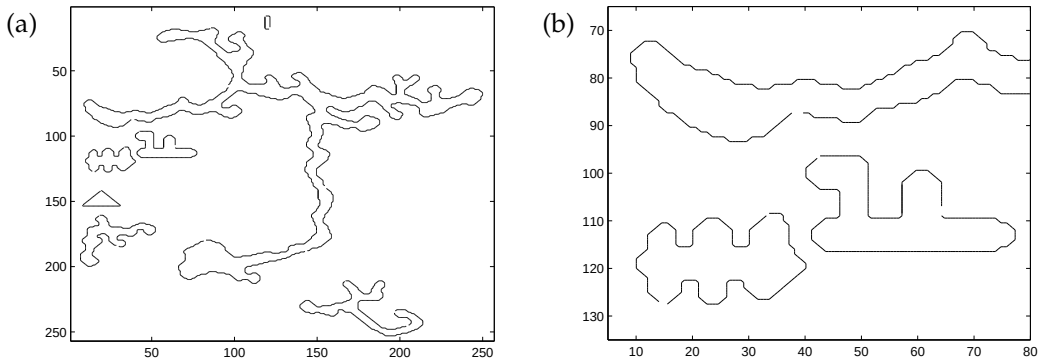


Figure 8: (a) Segmentation results showing the remaining open segments of Test Pattern B when 20% uniformly distributed noise is added to the physical image. (b) Close up of one region illustrating the fragmented open contours.

metric features even though they are unlikely to be present in a real image, since these shapes present additional challenges for the segmentation algorithm. The scaling of the smallest geometries can be seen in the Roman numeral III feature at the top of the image in which all feature dimensions are three pixels. When no noise is present, the segmentation algorithm generates all 51 features in this case (not shown here). In Fig. 7(b), the underlying physical data has been corrupted by uniformly distributed white noise on $[0,.2]$. Although the edge map from the concentration method is fairly well organized, we cannot successfully close all of the contours in the image. Fig. 8 shows the remaining open segments. A clean up strategy is devised in Section 4.3 to piece together remaining open segments that may be connected to form a closed feature.

4.3 Open segment extraction

After finding all of the initial closed contours in the image, we begin the sequencing and segmentation procedures once again. We develop priority lists, (4.3), for the (remaining)

points on the edge map at each jump level threshold, JLT_i, i=1,$\cdots$,#*levels*, to find all of the remaining open segments. Since we are unable to close these remaining segments using our prior algorithm, we apply the same requirements with more "relaxed" parameters to try to connect as many segments as possible. Then we can try once again to close these new open segments, following the same closed contour algorithm but again with more relaxed parameters.

We call an open segment admissible if it is at least of length δ_4, and also define the maximum open segment connection length as $\delta_5 > \delta_2$ (see Table 1). Let us consider the current segment,

$$segment_c = \{X_1^c, \cdots, X_p^c\}, \tag{4.5}$$

and suppose there are n remaining admissible open segments, with each $segment_i$ defined by $\{\bar{X}_1^i, \cdots, \bar{X}_{q_i}^i\}$, for $i = 1, \cdots, n$. We test both ends of all n admissible open segments as possible appendages to $segment_c$. For ease of presentation, we define $segment_j$, $j = n+1, \cdots, 2n$, to be the same n admissible open segments in reverse order, given by

$$segment_j = \{\bar{X}_{q_i}^i, \cdots, \bar{X}_1^i\}, \quad \text{for } j = n+i \quad \text{and } i = 1, \cdots, n.$$

We begin by calculating the historical direction for $segment_c$, $\theta_c \in [-\pi, \pi)$, using (4.2) with corresponding values X_{p-2}^c, X_{p-1}^c and X_p^c. We then similarly determine the historical directions for each $segment_i$, $\theta_i \in [-\pi, \pi)$ using $\bar{X}_1^i$, $\bar{X}_2^i$, and $\bar{X}_3^i$, $i = 1, \cdots, 2n$.

We now use the average end position of $segment_c$,

$$X_{end}^c = \frac{1}{3}(X_{p-2}^c + X_{p-1}^c + X_p^c),$$

and the average start position of $segment_i$,

$$\bar{X}_{start}^i = \frac{1}{3}(\bar{X}_1^i + \bar{X}_2^i + \bar{X}_3^i),$$

to determine the offset angle between the end of $segment_c$ and the start of $segment_i$. Specifically, if $\bar{X}_{start}^i = (x_i, y_i)$ and $\bar{X}_{end}^c = (x_c, y_c)$ then the offset angle, $\rho_i \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, is given by

$$\rho_i = \arctan\left(\frac{y_i - y_c}{x_i - x_c}\right). \tag{4.6}$$

The end points are averaged to minimize the effect of a nonuniform segment end and to reduce the effect of any outliers.

The properties of $segment_c$ and $segment_i$ are now evaluated using the previous conditions (i)-(v) to determine if they can be connected. The proximity is easily quantified by calculating the distance between the end of $segment_c$ and the start of $segment_i$ from the average end positions of the two segments as

$$d_i = |X_{end}^c - \bar{X}_{start}^i|, \quad \text{for } i = 1, \cdots, 2n.$$

The alignment of $segment_c$ and $segment_i$ is evaluated by combining the contributions from θ_c, θ_i, and ρ_i as

$$\alpha_i = \big|\theta_c - (\theta_i + \pi)\big| + \big|\rho_i - \frac{1}{2}(\theta_c + \theta_i + \pi)\big|, \quad \text{for } i = 1, \cdots, 2n.$$

If α_i is small, then the two segment ends are in good alignment. This requires

$$\theta_c \approx (\theta_i + \pi) \quad \text{and} \quad \rho_i \approx \theta_c.$$

The idea here is to encourage contour growth in the historical direction of the current contour by causing admissible open segments that exist in that direction to appear closer. This is accomplished by integrating the proximity and alignment quantifiers into a single weighted value as

$$w_i = d_i\big(c_1\alpha_i + c_2\big), \tag{4.7}$$

where c_1 and c_2 are user defined constants chosen so that when α_i is small, $w_i < d_i$, and when α_i is large, $w_i > d_i$. We used $c_1 = .30$ and $c_2 = .25$ in our tests but did not attempt to optimize the parameters. Hence we see that the segment corresponding to

$$\min\{w_i\} < \delta_5, \quad i = 1, \cdots, 2n,$$

should be attached to $segment_c$ to connect two open segments (see Table 1), bearing in mind that the order of elements of the adjoining segment might be reversed.

Since preference is given to open segment pieces that are in the same general path of an existing contour, our procedure minimizes kinks. The weights are chosen so that a developing connected open segment will stretch further in a direction in which it has been progressing than in any other to locate a likely continuation segment. However, a contour may still advance in a direction that is not in alignment with its historical direction when a smoother appendage is not available. Similar ideas are used for contour following and edge linking methods, see [19] and references within.

A growing contour is continued in this way, appending as many of the remaining open segments as possible, until no further matches are possible. The process is then repeated by beginning at the opposite end of the initial segment, $segment_c$, so that we are assured the current contour has grown as large as possible in both directions. Once this process has been exhausted, another initial open segment is chosen, and the process repeats until an attempt has been made to connect all open segments. When we have connected as many open segments as possible, we can once again begin the segmentation procedure described in Section 4.2 to close the segments. Here, however, we choose $\delta_6 > \delta_2$, as the maximum distance used to close the contour (see Table 1). Fig. 9 illustrates the process of connecting open segments on Test Pattern B with noise. Recall from Fig. 8 that several features were not initially closed by the segmentation algorithm. We are now able to successfully parameterize closed contours around all of the features in the image. Our complete segmentation algorithm is provided in the appendix.

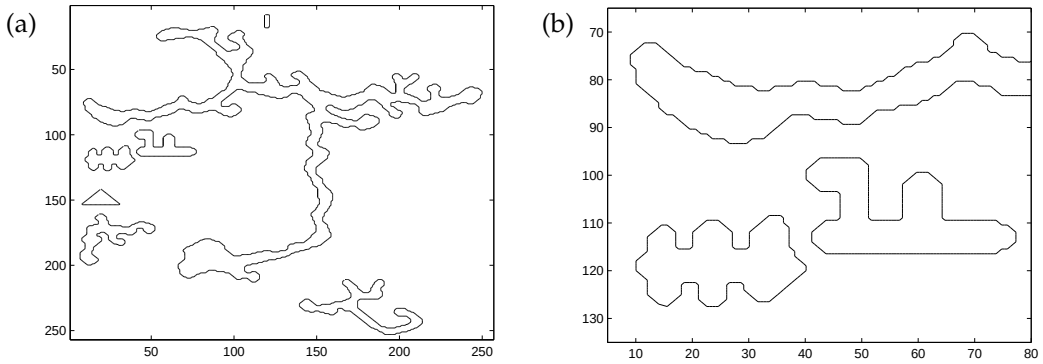


Figure 9: (a) Segmentation results showing the newly closed segments of Test Pattern B when 20% uniformly distributed random signal is added to the physical image. (b) Close up region illustrating the newly closed segments. Here $N=128$.

4.4 Signal variation

Since standard edge detection and segmentation techniques are typically based on thresholding algorithms, they may have difficulty when the underlying image contains some variation. On the contrary, the concentration method is a *high order method*, and is therefore well suited for such problems. We demonstrate this by overlaying Test Pattern B with a continuously varying signal given as

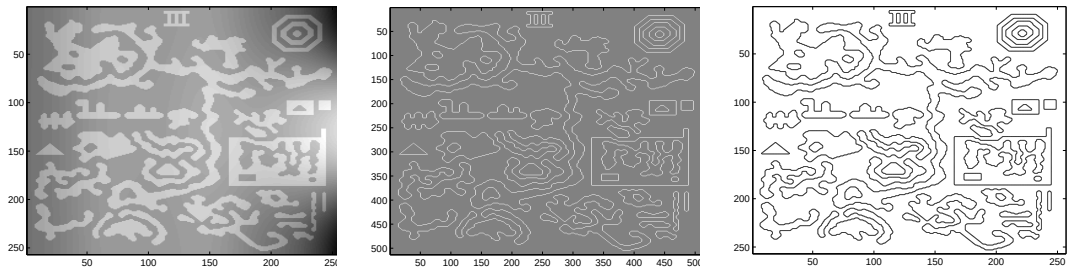
Example 4.1.

$$f(x,y)=\frac{1}{4}\left(e^x\cos y+\sin 3x+\cos 5y-x^2-y^2\right).$$

Fig. 10 shows the segmentation output for Example 4.1. As expected, the addition of a continuously varying signal underlying the image has little effect on the ability of the method to extract all of the features in the image. Similar results are obtained when the image contains noise.

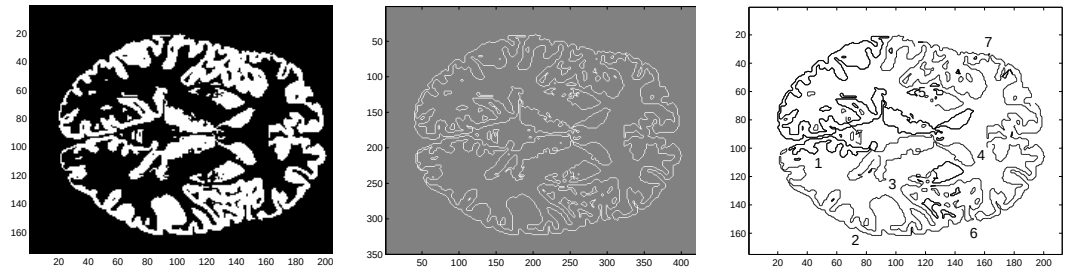
4.5 Simulated MRI brain image

As a final test of the edge detection and segmentation routines, we consider a simulated MRI brain image, obtained from a database of the McConnell Brain Imaging Center, [24]. These simulated images are often used to qualitatively compare segmentation algorithms, [10]. The physical data associated with this phantom has nearly piecewise constant values of either one or zero. There is some small variation from these values, however. Recall that the concentration method does not use the physical data of an image. The data from MRI are typically given as Fourier coefficients, and hence the concentration method can be applied directly. In this test case, however, the data is given as physical spatial data. We therefore first perform a two dimensional FFT to artificially generate the Fourier spectral data.



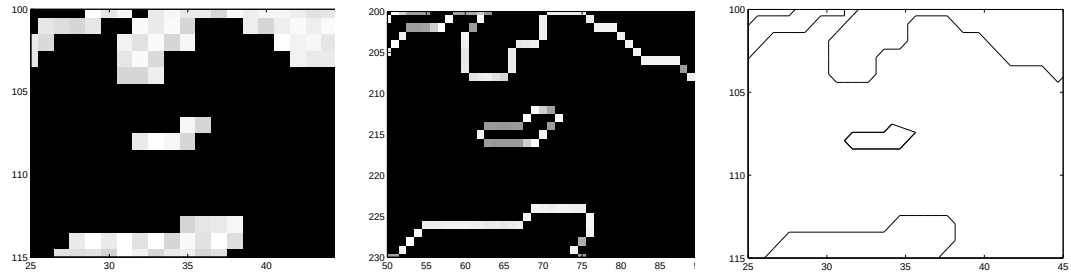
(a) (b) (c)

Figure 10: (a) Segmentation results for Test Pattern B with a continuously varying function given in Example 4.1 underlying the entire physical image. (b) Edge map generated from the concentration method. (c) Final segmentation routine result showing all closed contours. Here $N=128$.



(a) (b) (c)

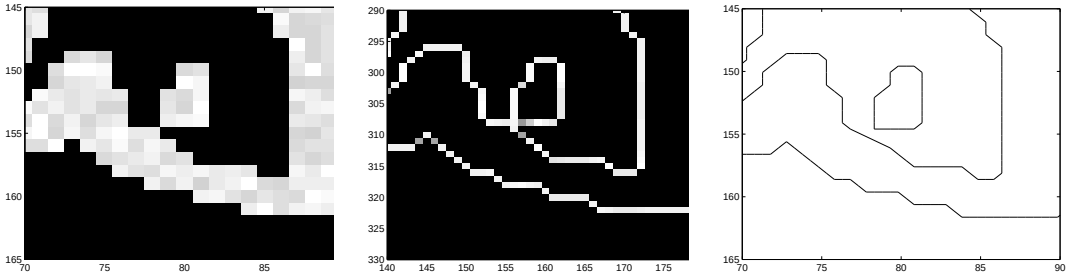
Figure 11: (a) Simulated MRI brain on 128×128 grid points. (b) Edge map produced from concentration method. (c) Segmentation including open and closed contours.



(a) (b) (c)

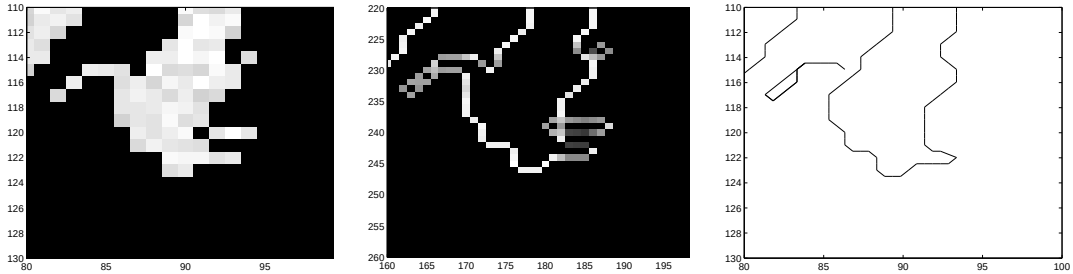
Figure 12: Close up of brain MRI Location 1: (a) Underlying physical image. (b) Edge map of the area. (c) Final segmentation result.

Fig. 11 shows the simulated MRI brain, the edge map produced by the concentration method, and the segmentation result that includes both open and closed contours. Although the edge map accurately depicts the internal boundaries, the segmentation algorithm has trouble extracting some of the smaller features and those with difficult geometries. Many feature sizes of one pixel occur within the image, and in some cases, they are contained within another feature. We examine some of the "severe" features in the MRI labeled in Fig. 11(c) to gain some insight into why our segmentation algo-



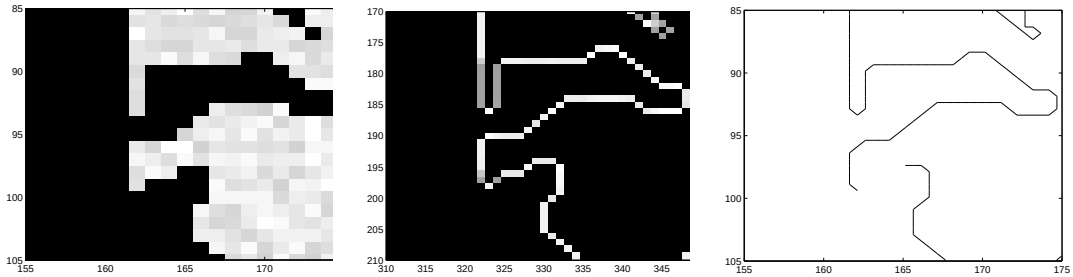
(a) (b) (c)

Figure 13: Same as Fig. 12, except for Location 2.



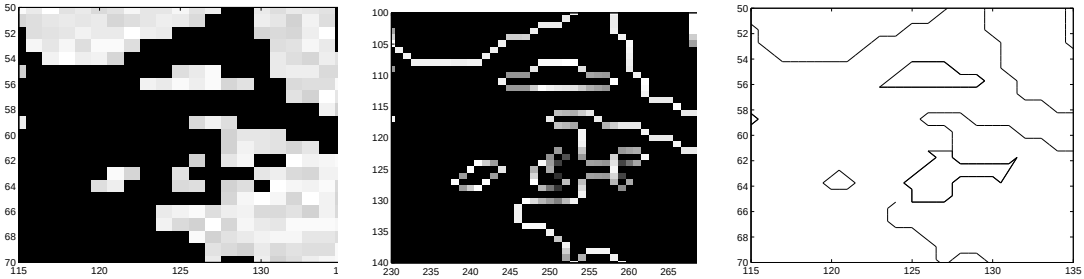
(a) (b) (c)

Figure 14: Same as Fig. 12, except for Location 3.



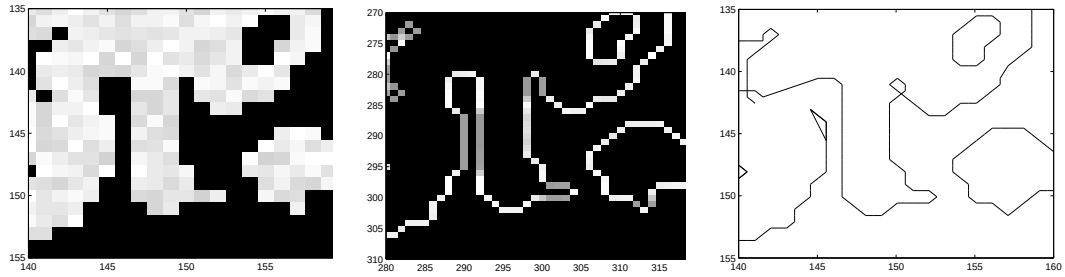
(a) (b) (c)

Figure 15: Same as Fig. 12, except for Location 4.



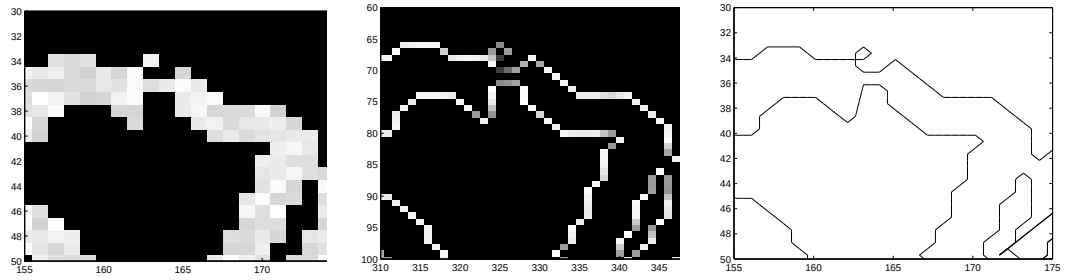
(a) (b) (c)

Figure 16: Same as Fig. 12, except for Location 5.



(a) (b) (c)

Figure 17: Same as Fig. 12, except for Location 6.



(a) (b) (c)

Figure 18: Same as Fig. 12, except for Location 7.

rithm sometimes fails. Close up views of the edge maps and resulting segmentation are displayed in Fig. 12 through 18. In most instances, the edge map accurately identifies even the very small features. The segmentation algorithm sometimes fails when it cannot determine the contour path of some of these severe features. The majority of the segmentation problems appear to be caused by small (one pixel) abruptly changing features, and in some instances the feature is a hole. Similar problems occur in edge linking and contour following algorithms. There are some difficulties that do not appear to be associated with the size of the feature, for instance see Fig. 15. This will be investigated in future work.

5 Conclusion

This paper presents a segmentation algorithm that extracts features from an edge map created by its Fourier spectral data. Several important imaging modalities collect Fourier spectral data directly, including MRI and SAR. There are several advantages in segmenting an image without first reconstructing it from its Fourier data. First, using the (filtered) Fourier reconstruction yields either Gibbs ringing or blurring at the internal boundaries. This in turn can cause difficulties for (standard) segmentation algorithms, [1, 2, 21]. On the other hand, high order reconstruction schemes that maintain the integrity of the internal boundaries can be costly, [18]. As illustrated by our examples, for applications where

only the shape that outline particular regions are of interest, image reconstruction is unnecessary; that is, only a high fidelity edge map is needed. The concentration method generates such an edge map directly from Fourier spectral data (i.e. *without* reconstruction). Our segmentation algorithm then produces sequences of ordered pairs and connects them to form contours. We note that the sequenced data also provides an initial guess for algorithms designed to parameterize contour regions for each feature of interest, [27]. Our approach is much different from typical techniques, [7, 22, 23, 25, 26], which rely on physical rather than Fourier data. These segmentation procedures use either a reconstructed image or an edge map of image intensity values. For this reason, valid comparisons of our methods with other edge detection and segmentation procedures are difficult. As presented, the sequencing and contour closing algorithms are somewhat ad hoc with several user defined parameters. Tuning of these parameters will clearly require some knowledge about the particular application. Nevertheless, our examples strongly suggest that the edge map produced by the concentration method provides a very accurate road map for closing many feature contours in an image. We may improve our technique by including high order reconstruction information only at the internal boundaries. We know from [1, 2] that segmentation algorithms benefit when the entire image is reconstructed. We anticipate that performing a less costly reconstruction (only at the boundaries) will allow us to effectively segment an image using other well established segmentation techniques, such as clustering. Finally, since our method is high order and works well when there is variation in the underlying function, we believe that it can be used effectively to determine solution domains for time dependent problems.

Acknowledgments

This work was partially supported by NSF grants CNS 0324957, DMS 0510813, DMS 0652833 and NIH grant EB 02553301 (AG) The first author would also like to thank the ICOSAHOM committee for the invitation to speak at this conference.

Appendix

Algorithm A.1: Edge Detection and Segmentation

1. Given (3.2), determine two edge maps, (3.3)using the concentration method, (2.3), enhanced by the *minmod* algorithm, (2.9).
2. Discard any edges that fall below *noisethreshold* (Section 3.2).
3. Apply Algorithm (3.1) to strengthen the edge map, which is now given by (3.4).
4. Divide the edge map (3.4) into 10 different jump levels.
5. Starting with the highest jump level, determine parameterized lists of edge points based on conditions (i)-(v) and close as many sequences as possible.
6. If open sequences still remain, repeat the previous step with more "relaxed" parameters to close as many remaining open sequences as possible.

References

[1] R. Archibald, K. Chen, A. Gelb, and R. Renaut, Improving tissue segmentation of human brain MRI through pre-processing by the Gegenbauer reconstruction method, *NeuroImage*, **20:1** (2003) 489-502.

[2] R. Archibald and A. Gelb, A method to reduce the Gibbs ringing artifact in MRI scans while keeping tissue boundary integrity, *IEEE Medical Imaging*, **21:4** (2002) 305-319.

[3] N. Banergee and J. Geer, Exponentially accurate approximations to piecewise smooth periodic Lipschitz functions based on Fourier series partial sums, *J. Sci. Comput.*, **13** (1998) 419-460.

[4] N. Bary, Treatise of TrigonometricSeries, The Macmillan Company, New York, 1964.

[5] R. Bauer, Band Filters for Determining Shock Locations, Ph.D. Thesis, Division of Applied Mathematics, Brown University, Providence, Rhode Island, (1995).

[6] J.P. Boyd, Chebyshev and Fourier Spectral Methods, Second Edition, Dover Publications, Mineola, New York, 2001.

[7] J. Canny, A computational approach to edge detection, *IEEE Trans. PAMI,* **8:6** (1986) 679-698.

[8] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T.A.Zang, Spectral Methods in Fluid Dynamics, Springer-Verlag, Berlin, 1988.

[9] K. Cheng, W. Wang, H. Qin, K. Wong, H. Yang, Y. Liu, Fitting subdivision surfaces to unorganized point data using SDM, *Proceedings of the* 12^{th} *Pacific Conference on Computer Graphics and Applications*, **00** (2004) 16-24.

[10] Z. Chi and Y. Li, MR Brain image segmentation based on self-organizing map network, *Int. J. Inform. Tech.*,, **11:8** (2005) 44-53.

[11] K.S. Eckhoff, Accurate reconstructions of functions of finite regularity from truncated series expansions, *Math. Comp.*, **64** (1995) 671-690.

[12] A. Gelb and D. Cates, Detection of edges in spectral data III – refinement of the concentration method, to appear in J. Sci. Comput., 2007.

[13] A. Gelb and E. Tadmor, Detection of edges in spectral data, *Appl. Comp. Harmonic Anal.*, **7** (1999) 101-135.

[14] A. Gelb and E. Tadmor, Detection of edges in spectral data II – nonlinear enhancement, *SIAM J. Numer. Anal.*, **38:4** (2000) 1389-1408.

[15] A. Gelb and E. Tadmor, Adaptive edge detectors for piecewise smooth data based on the MinMod limiter, *J. Sci. Comput.*, **28:2-3** (2006) 279-306.

[16] R. Gonzales and R. Woods, Digital Image Processing, Addison-Wesley Publishing Company, 1992.

[17] D. Gottlieb and S. Orszag, Numerical Analysisof Spectral Methods: Theory and Applications, SIAM, Philadelphia, 1977.

[18] D. Gottlieb and C.-W. Shu, On the Gibbs phenomenon and its resolution, *SIAM Review*, **30** (1997) 644-668.

[19] A.K. Jain, Fundamentals of Digital Image Processing, Prentice-Hall, 1986.

[20] G. Kvernadze, Determination of the jump of a bounded functionby its Fourier series, *J. Approx. Theory,* **92** (1998) 167-190.

[21] Z. Liang and P. Lauterbur, Principles of Magnetic Resonance Imaging: A Signal Processing Perspective, IEEE Press, 2000.

[22] S. Mallat and W. Hwang, Singularity detection and processing with wavelets, *IEEE Trans. Inform. Theor.,* **38** (1992) 617-643.

[23] D. Marr and E. C. Hildreth, Theory of edge detection, *Roy. Soc. London Ser. B,* **207** (1980)

187-217.
[24] `http://www.bic.mni.mcgill.ca/brainweb/`, McGill University's Montreal Neurological Institute.
[25] I. Sobel, An isotropic 3×3 image gradient operator, *Machine Vision for Three-Dimensional Scenes (H. Freeman editor)*, Academic Press, Boston, (1990).
[26] D. Vernon, Machine Vision, Prentice-Hall, 1991.
[27] W. Wang, H. Pottmann, Y. Liu, Fitting B-spline curves to point clouds by curvature-based squares distance minimization, ACM Transactions on Graphics, **25:2** (2006) 214-238.