EdgeCS: Edge Guided Compressive Sensing Reconstruction

Weihong Guo\textsuperscript{a} and Wotao Yin\textsuperscript{b}

\textsuperscript{a}Case Western Reserve University, Dept. of Mathematics, Cleveland, OH, USA.
\textsuperscript{b}Rice University, Dept. of Computational and Applied Mathematics, Houston, TX, USA.

ABSTRACT

Compressive sensing (CS) reconstructs images from a small number of projections. We propose EdgeCS - edge guided CS reconstruction - to recover images of higher qualities from fewer measurements than the current state-of-the-art methods.

Accurate edge information can significantly improve image recovery quality and speed, but such information is encoded in the CS measurements of an image. To take advantage of edge information in CS recovery, EdgeCS alternatively performs CS reconstruction and edge detection in a way that each benefits from the latest solution of the other. EdgeCS is fast and returns high-quality images. It exactly recovers the $256 \times 256$ Shepp-Logan phantom from merely 7 radial lines (or 3.03% k-space), which is impossible for most existing algorithms. It accurately reconstructs a $512 \times 512$ magnetic resonance image from 21% noisy samples. Moreover, it is also able to reconstruct complex-valued images. Each took about 30 seconds on an ordinary laptop. The algorithm can be easily ported to GPUs for a speedup of more than 10 folds.

Keywords: Compressive sensing, edge detection, total variation, discrete Fourier transform, magnetic resonance imaging

1. INTRODUCTION

1.1 Background

Compressive sensing (CS) exploits the sparsity of an unknown signal to recover the signal from much fewer linear measurements than required by the Nyquist-Shannon sampling.\footnote{The fact that CS requires very few measurements makes it very useful to reduce sensing cost in a variety of applications.}

Let $\bar{u}$ be the underlying signal of interest and $\Psi \in \mathbb{C}^{m \times n}$ with $m < n$ (sometimes $m \ll n$) be a sensing (sampling) matrix. The “encoding” process employs a certain physical or digital means to collect data $b \in \mathbb{C}^m$ that is either $\Psi \bar{u}$ or $\Psi \bar{u} + r$ where $r$ is noise. The “decoding” process is to reconstruct the image $\bar{u}$ from $b$. To reconstruct a sparse $\bar{u}$, it is natural to solve the $\ell_0$ problem when there is no noise:

$$\ell_0 : \min_{\bar{u}} \{ \| \Phi u \|_0 : \Psi u = b \}. \quad (1)$$

The $\ell_0$ problem is combinatorially expensive.\footnote{It is more tractable to solve the following $\ell_1$ optimization problem (basis pursuit):}

$$\ell_1 : \min_{\bar{u}} \{ \| \Phi u \|_1 : \Psi u = b \} \quad (2)$$

$\ell_1$ minimization yields sparse solutions under certain conditions (see\textsuperscript{5–7} for explanations) and also have efficient algorithms such as.\textsuperscript{8–15} For (2), the minimum number of measurements for successful reconstruction is $m = O(k \log(n/k))$ and $O(k \log(n)^4)$ when $\Psi$ is a Gaussian random matrix and partial Fourier ensemble,\textsuperscript{1,16} respectively. These numbers are smaller than $n$ in order of magnitude. Comparing with (1), (2) is much easier to solve but requires significantly more measurements. Other CS decoders include those based on $\ell_p (p < 1)$,\textsuperscript{17–23} homotopic $\ell_0$ approximation,\textsuperscript{24} and others.

Send correspondence to the first author at wxg49@case.edu.
1.2 Contributions

In this paper, we present an edge guided CS (shortened as EdgeCS) reconstruction scheme to significantly reduce the measurement requirement and to improve recovery quality. Given undersampled and/or noisy data, EdgeCS reconstructs images with less error or artifact than the state-of-the-art method. EdgeCS was implemented based on RecPF,\(^{25}\) a very fast CS reconstruction code based on TV and \(\ell_1\) regularization, and compares favorably with RecPF on image recovery from incomplete Fourier samples at very low sampling rates and with varying amounts of noise.

2. PROPOSED METHODS

Edge detection is a process to capture the significant properties of objects in the image. There are different types of edges, including step edge (discontinuity in intensity), line edge (local extrema), and junction (where at least two edges meet). In this paper, we focus on step edges which are the most common type of edge encountered, and apply edge detection to enhance CS reconstruction for one and two dimensional signals. A one-dimensional demonstration is shown in the next subsection, in which connection with previous work that explains the theoretical foundations is laid out. Subsection 2.2 below focuses on two-dimensional image reconstruction.

2.1 One-dimensional EdgeCS

We begin our exposition with a simple demo of reconstructing a piece-wise constant signal from its random measurements. The signal \(\bar{u}\) has \(n = 200\) entries and 25 randomly located jumps. The jump sizes are sampled from the identically independently distributed (iid) Gaussian distribution. Let \(\Psi\) be a \(60 \times 200\) Gaussian random matrix and \(b := \Psi \bar{u}\). The problem is to recover \(\bar{u}\).

Define \(TV(u) = \sum_{i=1}^{n-1} |u_{i+1} - u_i|\). We compare the solutions of total variation (TV) minimization,

\[
\min \{TV(u) : \Psi u = b\},
\]

and weighted TV minimization,

\[
\min \left\{ \sum_{i=1}^{n-1} g_i |u_{i+1} - u_i| : \Psi u = b \right\},
\]

where weights \(g_i\) are updated by jump detection. \(u\) and \(g\) are iteratively updated in the following algorithm:

**1D EdgeCS: Jump Guided TV Minimization for 1D signals**

**Input:** \(\Psi, b, n\).

1. Set the iteration number \(k \leftarrow 1\) and initialize \(g_i = 1\), \(\forall i\);
2. While the stopping condition is not met, do
   (a) Subproblem: \(u^{(k)} \leftarrow \) solve weighted TV minimization with weights \(g_i\);
   (b) Jump detection: \(I^{(k)} \leftarrow \{i : |u_{i+1} - u_i| > 2^{-k} \max \{|u_{j+1} - u_j|, j = 1, \ldots, n\}\}\);
   (c) Weight update: \(g_i \leftarrow 0\), \(\forall i \notin I^{(k)}\); \(g_j \leftarrow 1\), \(\forall j \notin I^{(k)}\);
   (d) \(k \leftarrow k + 1\).

In our test, the above algorithm returned the true signal \(\bar{u}\) in merely six iterations, but TV minimization failed to return \(\bar{u}\). Since \(g_i \equiv 1\) initially, the first iteration of EdgeCS is TV minimization. The solution of TV minimization and those at the end of 2nd, 4th, and 6th iterations are depicted in Figure 1. Subfigures (b), (c), and (d) highlight the detected jumps including both the true and false ones. Relative errors are computed as \(\|u^{(k)} - \bar{u}\|_2/\|\bar{u}\|_2\). The quality of these jumps are given in quadruplets “(total,det,good,bad)”, which are defined as follows:

- total: the total number of jumps in current \(u^{(k)}\).
- det: the number of detected jumps, equal to |\(I^{(k)}\)|.
- good: the number of true jumps.
- bad: the number of false jumps.

\(^*\)Extensions to three and higher dimensional signals are straightforward.
The TV solution (1st iteration) roughly matches the true signal but misses many small jumps. It contains false jumps and artifacts that are relatively small in size, but most of its large jumps are preserved at their exact locations. Hence, thresholding at $2^{-1} \max \{|u_{j+1} - u_j|\}$ identifies six of them (shown for $k = 1$ in the table) with no false detections. By setting the corresponding $g_j$ to 0 for the second iteration, a smaller reconstruction error is obtained. The improved solution of iteration 2 is then used for detecting more jumps. As EdgeCS iterates, the solutions improve and the thresholds reduce, so more jumps are detected. Note that at iteration 4, the detection includes 3 false jumps; however, since it also introduces 4 true jumps, iteration 5 yields a reduced reconstruction error. Generally, as long as false detections are relative few, more detections lead to lower reconstruction errors. The detection at iteration 5, though including false jumps, has included enough true jumps to allow an exact reconstruction at iteration 6. The solution of iteration 6 exactly recovers all jumps and has a tiny error. All subproblems were solved with MATLAB’s linear programming solver “linprog” with the default parameters.

### 2.1.1 Theoretical justification

In this section, we discuss some theoretical results from our previous work\(^{26}\) to justify our approach. For simplicity, we assume that the unknown signal has sparse spikes, rather than sparse jumps. Consequently, we discuss the results in terms of weighted $\ell_1$-minimization rather than weighted TV minimization. The results for the latter are analogous. Let $I \subseteq \{1, \ldots, n\}$ be an index subset, and $T := I^C$. The underlying model for recovering a sparse vector $\tilde{u}$ is

<table>
<thead>
<tr>
<th>Itr $k$</th>
<th>Jumps</th>
<th>Relative Error $\frac{|x^{(k)} - \bar{u}|_2}{|\bar{u}|_2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TV 1</td>
<td>60</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>60</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>59</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>59</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>59</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>
Similar to the algorithm of EdgeCS for 1D signals, \( I \) consists of detected nonzeros and is initially set to \( \emptyset \). At each iteration \( k \), (3) is first solved for \( T := I^C \), and the solution \( u^{(k)} \) is used for support detection to generate a new \( I \). Below we summarize the results in the following aspects: (i) an exact recovery condition, (ii) support detection performance and practical detection methods, and (iii) solution quality measurement and algorithm stopping, and (iv) performance in terms of reconstruction errors for compressible signals and noisy measurements.

For exact reconstruction, one needs the \( t \)-NSP (truncated null space property)\(^{26} \) of the sensing matrix \( \Psi \). Given \( T \), the property essentially says that no \( \eta \in \mathcal{N}^y(\Psi) \), the null space of \( \Psi \), has a mass concentrating over any \( L \) components within \( T \); in other words, \( \eta \) has to be spread in \( T \). The spreading is measured by \( \gamma \). For simplicity, we use \( t \)-NSP\((t,L,\tilde{\gamma})\) to denote the \( t \)-NSP of order \( L \) for \( \tilde{\gamma} \) and \( t \). The \( t \)-NSP\((|T|, L, \tilde{\gamma})\) of \( \Psi \) with \( \tilde{\gamma} < 1 \) is sufficient for (3) to recover any sparse signal \( \tilde{u} \) with \( \| \tilde{u}_T \|_0 \leq L \).

At the beginning, \( T = \{1, \ldots, n\} \) and \( L = \| \tilde{u} \|_0 \). If \( \Psi \) satisfies \( t \)-NSP\((|T|, L, \tilde{\gamma})\) for \( \tilde{\gamma} < 1 \), (3) returns the true signal \( \tilde{u} \). Otherwise, its solution is subject to support detection. Let the numbers of correct and wrong detections be denoted by \( d_c \) and \( d_w \), respectively. Let \( t' = n - d_c - d_w \) and \( L' = \| \tilde{u} \|_0 - d_c \). If \( (L - L') > \tilde{\gamma}(t - t' - (L - L')) \) or equivalently \( d_c > \tilde{\gamma}d_w \), then the new \( \tilde{\gamma}' \) is smaller than \( \tilde{\gamma} \). If \( \tilde{\gamma}' < 1 \), then (3) in the next iteration returns \( \tilde{u} \). Otherwise, the algorithm continues. Here, \( d_c > \tilde{\gamma}d_w \) means that the number of correct detections is at least \( \tilde{\gamma} \) times larger than that of the false detections. This result means that the iterative support detection approach is robust to a small number of wrong detections as long as the number of correct detections is larger by a certain amount. This explains why in the demo in Figure 1, exact recovery is achieved through iterative jump detections that include false detections.

For some common types of signals such as sparse Gaussian and power–law decaying signals, it is easy to detect true nonzero entries and keep false detections low by two methods based\(^{26} \) on thresholding. The first is as simple as letting the detection set \( I \leftarrow \{ i : |u^{(k)}_i| > \|u^{(k)}\|_\infty / \rho^k \} \), where \( \rho > 1 \) is a constant. The second method, looking for a so-called first significant jump, is more robust. It sorts \( |u^{(k)}_i| \), \( i = 1, 2, \ldots, n \) increasingly into \( |u^{(k)}_i| \)'s and looks for the smallest index \( (i) \) such that \( |u^{(k)}_{i+1}| - |u^{(k)}_i| > \kappa \), where \( \kappa \) can be set as, for example, \( \|u^{(k)}\|_\infty / k \). The detection set \( I \) is set corresponding to \( \{ (i+1), \ldots, (n) \} \). A typical plot of sorted components are given in Figure 2, in which the smaller components are clustered and increase slowly and the larger ones are spread and increase sharply. Hence, a gap (first significant jump) right below the solid line naturally forms.

It is reliable to stop the iterations by checking the tail size of \( u^{(k)} \), defined as the fraction \( \sum_{i \in T}|u^{(k)}_i|/\sum_{j \in T}|u^{(k)}_j| \), i.e., the thought–nonzero divided by the thought–nonzero. It can be shown that if \( u^{(k)} = \tilde{u} \), the fraction is zero; if \( \tilde{u} \) is not sparse but compressible and \( T \) contains the smaller components, the fraction depends on the decay rate and will be small when the tail of \( \tilde{u} \) is small. The tail size measures how well \( u^{(k)} \) and \( T \) matches the measurements and expected sparseness/compressibility. Typically, no more than ten iterations are needed before stopping.

Most practical signals are not exactly sparse but compressible (i.e., the nonzeros follow a fast decaying distribution). Based on the best \( L \)–term approximation error: \( \sigma_L(y)_1 := \inf \{ \| y - u \|_1 : \| u \|_0 \leq L, u \in \mathbb{R}^{\dim(y)} \} \), it can be shown that if the decay of \( \tilde{u} \) is faster and/or \( T \) contains more components of \( \tilde{u} \) that are relatively smaller, then the solution of (3) has smaller errors.

### 2.2 Two-dimensional EdgeCS

The 2D EdgeCS algorithm is the same as the 1D version except that model (4) below is solved in Step 2(a) to return an image \( u^{(k)} \), which is subject to edge detection in Step 2(b). Let \( u = (u_{i,j}) \in \mathbb{R}^{m \times n} \) denote a grey-scale image. We use the weighted TV model

\[
\text{Truncated } \ell_1 \text{ minimization: } \min \{ \| u_T \|_1 : \Psi u = b \}. \tag{3}
\]
†for the 4-neighbor, all weights equal with. 4/8/16-neighborhoods are depicted in Figure

The isotropic discretization of TV is

\[ \min_{\mu} WTV(u) + \alpha \| \Phi u \|_1 + \frac{1}{2} \| \Psi u - b \|_2^2, \]  

where \( WTV(u) \) stands for a certain discretization of TV with weights (see below for details) and \( \alpha \| \Phi u \|_1 \) is an optional regularization term if \( u \) is sparse under the orthonormal basis \( \Phi \). Below, we explain how to formulate weighted TV with isotropic and anisotropic discretizations and use edges to determine the weights.

The isotropic discretization of TV is

\[ TV^{iso}(u) = \sum_{i,j} g_{i,j} \| D_{i,j} u \|_2 := \sum_{i,j} g_{i,j} \sqrt{\| u_{i+1,j} - u_{i,j} \|^2 + \| u_{i,j+1} - u_{i,j} \|^2}, \]

where \( g_{i,j} \)'s are weights. In the unweighted version, \( g_{i,j} \)'s are set to a constant such as 1. Anisotropic discretizations of TV have different forms according to the number of neighbors each pixel \( (i,j) \) is taken finite differences with. 4/8/16-neighborhoods are depicted in Figure 3. A standard anisotropic discretization of TV has the form

\[ TV^{aniso}(u) = \sum_{\alpha} g_{\alpha} \| D_{\alpha} u \|_2 := \sum_{\alpha} g_{\alpha,\sim(k,l)} \| u_{i,j} - u_{k,l} \|, \]

where the sum is taken over all pairs \( \alpha \) of neighbors \( (i,j) \) and \( (k,l) \), and \( g_{\alpha,\sim(k,l)} \) is set to proper values\(^\dagger\). In our algorithm EdgeCS, each weight \( g_{i,j} \) or \( g_{\alpha} \) has two possible values: it original value or 0.

The weights \( g_{i,j} \) in the isotropic TV are defined on pixels \( (i,j) \). We let \( g_{i,j} \leftarrow 0 \) if \( (i,j) \) is on an edge; otherwise, \( g_{i,j} \) keeps its original value. In an anisotropic discretization of TV, \( g_{\alpha} \) is defined for \( \alpha \), a pair of pixels, so we let \( g_{\alpha} \leftarrow 0 \) if the pair of pixels are on opposite sides of an edge; otherwise, \( g_{\alpha} \) keeps its original value. Anisotropic TV needs sub-pixel (i.e., between-pixel) edge detection.

With the original and sub-pixel (see next paragraph) versions of Canny edge detector, we found that a weighted anisotropic TV, even the simplest 4-neighbor version, performs much better than weighted isotropic TV for two simple reasons as follows. First, there are more weight parameters \( g_{\alpha} \) in anisotropic TV than \( g_{i,j} \) in isotropic TV, so the former has more degrees of freedom to adapt to given edges. Second, each isotropic TV weight \( g_{i,j} \) serves the two pairs of pixels under the corresponding square root, so it is not clear how to set \( g_{i,j} \) when exactly one of the pairs is cut by an edge, for example, \( u_{i,j} = u_{i+1,j} > 0 \) and \( u_{i,j+1} = 0 \).

Of course, another way to assign weights for isotropic TV is to give a weight for each pair of pixels in the form of \( \sqrt{g_a \cdot a^2 + g_b \cdot b^2} \). However, this may lead to complications that require further investigation. Assume \( g_a = 1 \). If \( g_b = 0 \), then \( \sqrt{g_a \cdot a^2 + g_b \cdot b^2} = |a| \) so the regularization on \( |a| \) is first-order. If \( g_b = 1 \), then

\(^\dagger\)For a cut metric,\(^\dagger\) for the 4-neighbor, all weights equal \( \pi/4 \); for the 8-neighbor type, nearest 4 neighbors have weights \( \pi/8 \) and next 4 neighbors have \( \sqrt{2\pi/16} \); for the 16-neighbor type, nearest 4 neighbors have weights \( \frac{1}{2} \tan^{-1}(\frac{1}{2}) \), next 4 neighbors have \( \sqrt{2\pi(\frac{3}{4} - \tan^{-1}(\frac{1}{2}))} \), and last 8 neighbors have \( \frac{\sqrt{2\pi}}{80} \).
the order of regularization on $|a|$ depends on $|b|$. If $|b| > 0$, the regularization is less than first-order (as $|a|/\sqrt{g_a \cdot a^2 + g_b \cdot b^2} < 1$). If $|b| \gg |a|$, there is little regularization on $|a|$. Therefore, whether $g_a$ (or $g_b$) equals 0 not only determines the regularization on $|a|$ (or $|b|$) but also affects the regularization on $|b|$ (or $|a|$).

We modified the classical Canny edge detector\(^{28}\) to give sub-pixel edges. The classical Canny method obtains edges through hysteresis thresholding on gradient magnitude $\|D_j u\|_2$. Our modification performs hysteresis thresholding on directional gradients $D_\alpha u$, e.g., $|u(i,j) - u(i+1,j)|$ and $|u(i,j) - u(i,j+1)|$ separately.

3. ALGORITHM

For notational simplicity, we have assumed that the underlying signal is a real, grey-scale, 2D image. Generalizations to complex, higher-dimensional, multi-channel signals are straightforward. Moreover, the computation fits for anisotropic TV of an arbitrary order. As in the previous section, $\sum_\alpha g_\alpha |D_\alpha u|$ denotes weighted anisotropic TV. Our algorithm is applied to the reconstruction model

$$\min_u \mu \sum_\alpha g_\alpha |D_\alpha u| + \alpha \|\Phi u\|_1 + \frac{1}{2} \|F_p u - b\|^2_2, \quad (5)$$

where we have set the sensing matrix $\Psi = F_p := PF$, a Fourier transform $F$ subsampled by $P$. The algorithm is a combination of iterative edge detection with the recent solver RecPF,\(^{25}\) which skillfully applies variable splitting (introducing $w_\alpha = D_\alpha u$ and $z = \Phi u$) and the well-known alternating direction method of multipliers\(^{29}\) (ADM) for solving (5). It minimizes with respect to $(w, z)$ and then updates the multipliers, all done in a cyclic way. Since all subproblems can be analytically solved, the algorithm is ultimately fast. The $u$-subproblem involves both $F_p$ and the $D_\alpha$’s can be quickly solved by FFTs. The remaining $w$ and $z$ subproblems are component-wise separable and easy. Such variable splitting for TV regularization was perhaps first discovered in\(^{30}\) for image denoising and deblurring, and was recently generalized with improvements to multi-channel problems,\(^{31}\) the TV-$L^1$ model,\(^{32}\) and TV-based CS recovery in RecPF\(^{25}\) and Split Bregman.\(^{33}\) We apply simple modifications to RecPF to deal with the above weighted, anisotropic discretization of TV. The EdgeCS algorithm framework is given in Algorithm 3. Smoothing parameters are tuned for the edge detector (Step 4) in a way that as $k$ increases, smaller and smoother edges are detected.

**EdgeCS:** Given $b$, $\Phi$, $F_p$, $\alpha$, $\mu$

1. Set $k \leftarrow 1$.
2. Do
3. Call modified RecPF to solve (5); obtain solution $u^{(k)}$;
4. Perform sub-pixel edge detection on $u^{(k)}$; $g_\alpha \leftarrow 0$ over detected edges; otherwise set $g_\alpha$ to its original value;
5. If there is (nearly) no change in $g$, STOP; otherwise, $k \leftarrow k + 1$ and continue.

4. EXPERIMENTAL RESULTS

As the first algorithm that learns edges from CS reconstruction, it has no similar algorithms to compare with. We note a related paper,\(^{34}\) which compares three methods for edge detection from incomplete Fourier measurements. However, none of them produces images. Therefore, we compared EdgeCS with RecPF,\(^{25}\) a fast state-of-the-art reconstruction algorithm on recovering 2D images from incomplete Fourier samples. RecPF (6) returns an image as the solution of the following minimization problem:

$$\min_u \mu TV(u) + \alpha \|\Phi u\| + \frac{1}{2} \|F_p u - b\|^2_2, \quad (6)$$

where $TV(u)$ can be the isotropic or any anisotropic discretizations of TV. We used the isotropic TV for RecPF and 4-neighborhood anisotropic TV for EdgeCS for their best performance. EdgeCS and RecPF were compared on tests with very few samples, as well as those with highly noisy samples.

The parameters $\mu$ and $\alpha$ control the overall performance of RecPF and EdgeCS. We fixed $\alpha$ to 0 in all tests in order to focus on where the two models are different. Upon being called, both algorithms scale the input data.
Figure 4. RecPF (top row) versus EdgeCS (bottom row) on partial Fourier samples in varying sizes. Left, middle, right columns are recovered images from noiseless samples taken on 15, 8, 7 radial lines (6.44%, 3.98%, 3.03% of k-space samples) respectively. $\mu = 10^{-10}$ was used.

$$\sigma^2 = 0, \mu = 10^{-10} \quad \sigma^2 = 0.01, \mu = 5 \times 10^{-5} \quad \sigma^2 = 0.05, \mu = 10^{-4}$$

Figure 5. RecPF (top row) versus EdgeCS (bottom row) on partial Fourier samples with varying noise levels. Left, middle, right columns are images recovered from samples taken on 15 radial lines plus Gaussian noise with variances 0 (no noise), .01, and .05, respectively.

and parameter $\mu$ in the same way to normalize the effects due to varying image sizes, pixel intensity ranges, and sample sizes. Specifically, $\mu$ is multiplied by the sample size and divided by the square root of image size, and $b$ is divided by the intensity range. The returned images are denormalized by multiplying with the intensity range. All RecPF results were obtained with tight parameters to avoid loss of quality due to early stopping. All tests were run under Windows Vista and MATLAB v7.6 (R2008a) on a laptop with an Intel Core 2 Duo CPU at 2.0 GHz and 3 GB of memory.
Given sufficient samples, both EdgeCS and RecPF are able to exactly recover images. However, from the same set of samples that are less than enough for RecPF to exactly recover the images, EdgeCS returned much better images than RecPF. Figure 4 depicts the images of EdgeCS and RecPF recovered from the same three sets of samples that are 6.44%, 3.98%, 3.03% of k-space (or 15, 8, 7 radial lines) of the 256 × 256 Shepp-Logan phantom. The images of EdgeCS and RecPF from the 15-line samples are visually comparable. In terms of error and SNR, however, EdgeCS’s image is slightly better. From the 8-line samples, RecPF returned an image with a relative error of 37.15%, and EdgeCS returned a much better images in 30 seconds. From the 7-line samples, EdgeCS still achieved an almost exact recovery while RecPF returned an image with huge errors and lots of artifacts.

EdgeCS also returned better images than RecPF from noise-contaminated samples. Figure 5 depicts the images recovered from samples that are 6.44% of k-space (or 15, 8, 7 radial lines) added with Gaussian noise of varying variances $\sigma^2 = 0, .01, .05$ (the phantom has intensity values between 0 and 1). Corresponding to $\sigma^2 = 0$ (or no noise), RecPF and EdgeCS returned comparable images. Corresponding to $\sigma^2 = .01$ (or a low level of noise), EdgeCS produced an image with sharper edges (observable in the boxed areas). Corresponding to $\sigma^2 = .05$, EdgeCS returned a much better image than RecPF, which failed to recover the small shapes.

We also tested EdgeCS for recovering a 512 × 512 MR image from 20.87% k-space samples (100 radial lines) added with Gaussian noise of variances $\sigma^2 = 0, .05, .1$. Respectively, we used $\mu = 10^{-4}, 3 \times 10^{-4}, 10^{-3}$. Figure 6 depicts the results of back projections and EdgeCS.

Furthermore, EdgeCS is able to deal with complex-valued images and samples. This is demonstrated in Figure 7, which shows a complex-valued MR image recovered by EdgeCS. It remains easy to set the parameter $\mu$, which varies with the noise level but is independent of the sample size.

5. CONCLUSION AND DISCUSSION

We propose an edge guided CS reconstruction method to recover images of higher qualities from fewer measurements than the current state-of-the-art methods. Edge information is useful for image recovery, but this information is embedded in CS measurements. To extract edge information for CS recovery, we alternatively perform CS reconstruction and edge detection in a way that each benefits from the latest solution of the other. Anisotropic TV and sub-pixel edge detection are used in EdgeCS, which is fast and returns high-quality images. As an example, it is able to exact reconstruct the 512 × 512 Shepp-Logan phantom from only 3.03% samples in 30 seconds on an ordinary laptop. This is impossible for most of the existing methods.
The idea of using edge detection for CS recovery can be applied to reconstruct images from various samples/observations and used with various regularization methods/algorithms, for example $\ell_q, 0 < q < 1$, homotopic $\ell_0$, TV with varying orders, nonlocal TV, etc. More accurate edge detection will further improve the performance of EdgeCS. In particular, it would be interesting to incorporate edge detection methods\textsuperscript{34,35} in EdgeCS and evaluate the performance.

ACKNOWLEDGMENTS

The work of W. Guo was supported in part by Case Western Reserve University startup funds. The work of W. Yin was supported in part by NSF CAREER Award DMS-07-48839, ONR Grant N00014-08-1-1101, and an Alfred P. Sloan Research Fellowship.

REFERENCES


22. S. Foucart and M.-J. Lai, “Sparsest solutions of underdetermined linear systems via \( \ell_q \)-minimization for \( 0 < q \leq 1 \),” Applied and Computational Harmonic Analysis 26(3).


