Survey of Results in Compressed Sensing using Total Variation Minimization

Kevin Chow

December 15, 2015

1 Introduction

Applications of compressed sensing are often concerned with the recovery of images from linear measurements (MRI, radar). Some of the earliest papers in this subject [1,2] illustrate the power of compressed sensing by recovering an image using only a small number of the its Fourier coefficients. The paper [2] further exposed the possibility of *stable* and *robust* recovery of signals and images by suitably chosen sensing mechanisms.

A great deal of mathematical theory has since then been developed to address this phenomenon. The notion of a restricted isometry property (RIP) has proven to be extremely fruitful in establishing and generalizing stable and robust recovery results for large classes of sensing mechanisms (eg. [2, 3]). Subgaussian random matrices, Fourier ensembles, and more general bounded orthonormal systems, are just a few examples of mechanisms that have been studied under the RIP framework. Satisfying the RIP, optimal recovery bounds are available to solutions by a quadratically constrained ℓ_1 minimization program.

In this survey, we will focus on recovery guarantees of 1D and 2D signals using *total variation minimization.* The guarantees are relevant to a large class of sensing mechanisms in both uniform and non-uniform recovery settings. We then further focus on the case where measurements are taken with a subsampled Fourier operator. This latter case is of great interest since it is closely related to the aforementioned applications in medical imaging. In fact, for this reason, despite the difficulties presented in analyzing TV minimization, efforts persist in developing recovery guarantees. Empirical studies indicate that TV regularization recovers images of a higher quality than ℓ_1 minimization (say of the Haar wavelet coefficients) even when using the same set of measurements.

Our survey will start with a brief and not so comprehensive discussion of the general premise and principles of compressed sensing. Relevant notation and terminology will be defined. We will then proceed to examine recovery results in [7–9] and auxiliary results from [1,6]. Finally, we will perform numerical experiments that illustrate stable and robust recovery via TV minimization. We limit ourselves in the numerical experiments to Fourier measurements and recovery by the split Bregman algorithm [4,5], but one should not take that to be a limitation of the available theoretical results.

2 Basics of Compressed Sensing

2.1 Introduction to the ℓ_1-minimization problem

A central tenet of compressed sensing is that a sparse signal, $x \in \mathbb{C}^N$, can be recovered from a number of measurements, m, that is far fewer than the dimension N. A first formulation of a typical compressed sensing problem may be as follows: Suppose an unknown signal $x \in \mathbb{C}^N$ is s-sparse, i.e. the support of x has $|\mathrm{supp}(x)| \leq s \ll N$. From a set of linear measurements of x of the form $\{y_k = \langle a_k, x\rangle \mid 1 \leq k \leq m\}$, can we reconstruct x? Further writing the measurement acquisition as a linear system and adopting the more realistic scenario of noisy measurements and compressibility (rather than sparse x), it is proved in [2] that the solution $x^\sharp$ to the ℓ_1-minimization problem

$$\min_z \|z\|_1 \quad \text{subject to} \quad \|y - Az\|_2 \leq \epsilon, \qquad (\ell_1 \text{ min})$$

where $y = Ax + e$, $\|e\|_2 \leq \epsilon$, represents noisy measurements, satisfies

$$\|x - x^\sharp\|_2 \lesssim \epsilon + \frac{\|x - x_{\mathbf{s}}\|_1}{\sqrt{s}}. \qquad (1)$$

This result holds as long as the sensing (or measurement) matrix has the restricted isometry property (RIP) of order $\mathcal{O}(s)$ with sufficiently small restricted isometry constants (further details can be found in, eg. [3, Chapter 6]).

A crucial generalization of the program (ℓ_1 min) is to allow for an additional change of basis. Signals of interest are often not sparse in the canonical basis, but is so under a suitable transformation. An additional orthonormal transformation, say $z \leftarrow Uz$, poses some additional constraints on compatible sensing mechanisms, but are otherwise well studied. The error bound (1) otherwise holds under the same conditions.

2.2 Notation

In the preceding section we have introduced notation that will be common throughout this article. For $x \in \mathbb{C}^N$, $x^\sharp$ will denote a minimizer to a minimization problem, e.g. a minimizer to (ℓ_1 min). When subscripted with $\mathbf{s}$, $x_{\mathbf{s}}$ is the projection of x to an index set of its s-largest absolute entries and similarly, x_S is the restriction of x to indices in S. These conventions will be use analogously with 2D objects. Furthermore, when we discuss objects in 2D (typically images), we will denote as $X \in \mathbb{C}^{N\times N}$, and $\|X\|_p$ will be in the sense of vector p-norm. That is, $\|X\|_p^p = \sum_{i,j=1}^N |X_{ij}|^p$. To explicitly distinguish between actions on 1D and 2D signals, we will use $\mathcal{A}$ to denote a linear operator, $\mathcal{A} : \mathbb{C}^{N\times N} \to \mathbb{C}^m$, that acts on images. For example, when taking measurements of an images we may write $y = \mathcal{A}X$.

The character ϵ will be used exclusively as a bound on the ℓ_2-norm of measurement noise e. When we say $a \lesssim b$, it indicates that there exists some constant C independent of the variables involved such that $a \leq Cb$. We will also equate $\{1, \ldots, N\}$ with $[N]$.

Finally, for brevity, from here on we will use sparse and gradient sparse interchangeably. Whether we are referring to the sparsity of the signal or the sparsity of its gradient will be clear from context. Likewise we will use compressible in the same sense.

2.3 TV seminorm

To proceed, we must now define a discrete gradient operator and the associated total variation seminorm. Let $x \in \mathbb{C}^N$. Defining our discrete gradient operator as

$$\nabla : \mathbb{C}^N \to \mathbb{C}^{N-1}, \quad (\nabla x)_j = x_{j+1} - x_j, \quad \text{for } j \in [N-1],$$

the total variation seminorm is then

$$\|x\|_{TV} = \|\nabla x\|_1 = \sum_{j=1}^{N-1} |x_{j+1} - x_j|.$$

For an image $X \in \mathbb{C}^{N \times N}$, we define first vertical and horizontal difference operators,

$$\begin{aligned} D_1 X : \mathbb{C}^{N \times N} &\to \mathbb{C}^{(N-1)\times N}, \quad (D_1 X)_{ij} = X_{i+1,j} - X_{ij}, \\ D_2 X : \mathbb{C}^{N \times N} &\to C^{(N-1)\times N}, \quad (D_2 X)_{ij} = X_{i,j+1} - X_{ij}. \end{aligned}$$

Our gradient operator is $\nabla : \mathbb{C}^{N \times N} \to \mathbb{C}^{N \times N \times 2}$,

$$(\nabla X)_{ij} = \begin{cases} ((D_1 X)_{ij}, (D_2 X)_{ij}), & i, j \in [N-1], \\ (0, (D_2 X)_{Nj}), & j \in [N], \\ ((D_1 X)_{iN}, 0), & i \in [N], \\ (0, 0), & i = j = N, \end{cases} \tag{2}$$

and the TV seminorm is

$$\|X\|_{TV} = \|\nabla X\|_1 = \sum_{i,j=1}^{N} |X_{ij,1}| + |X_{ij,2}|. \tag{3}$$

The expression in (3) is known as anisotropic TV. One may instead define an isotropic TV and derive bounds similar to those presented in section 3 (see [7]). We also note that this but one way to define a the gradient operators. An alternative is to define them with periodic boundaries. This has the advantage of commuting with the discrete Fourier operator and admits advantages computationally. Results assuming periodicity of the gradient operators will also be detailed in section 3.

Similar in spirit as to how we arrive at (ℓ_1 min), we ask whether a minimizer $X^\sharp$ to

$$\min_Z \|Z\|_{TV} \quad \text{subject to} \quad \|y - \mathcal{A}X\|_2 \leq \epsilon \tag{TV min}$$

admits a bound of a form similar to (1).

Before proceeding further, we quickly address why the discussion in section 2.1 does not encompass (TV min). The difficulties lie with the discrete gradient operator, ∇. It is not an orthonormal transformation and is not even invertible without restriction to eg. mean-zero images.

3 Compressed Sensing with TV Regularization

3.1 A stable and robust uniform recovery guarantee

Let us open this section with a theorem from Needell and Ward ([8, Theorem 2]).

Theorem 1. *Let $X \in \mathbb{C}^{N\times N}$ and ∇X the discrete gradient as defined in (2). If we observed measurements $y = \mathcal{A}X + e$, $\|e\|_2 \leq \epsilon$, and the measurement operator $\mathcal{A}$ satisfies the RIP of order s, then the minimizer $X^\sharp$ to*

$$\min_Z \|Z\|_{TV} \quad \textit{subject to} \quad \|y - \mathcal{A}Z\|_2 \leq \epsilon \tag{4}$$

satisfies

$$\|X - X^\sharp\|_2 \lesssim \log\left(\frac{N^2}{s}\right)\left(\epsilon + \frac{\|\nabla X - (\nabla X)_\mathbf{s}\|_1}{\sqrt{s}}\right). \tag{5}$$

Within the same paper [8, Theorem 5], the authors give a slightly broader result that implies Theorem 1 by establishing ancillary results

$$\|\nabla X - \nabla X^\sharp\|_2 \lesssim \epsilon + \frac{\|\nabla X - (\nabla X)_\mathbf{s}\|_1}{\sqrt{s}}, \tag{6}$$

$$\|X - X^\sharp\|_{TV} \lesssim \sqrt{s}\epsilon + \|\nabla X - (\nabla X)_\mathbf{s}\|_1. \tag{7}$$

This is one of the first established results on stable and robust recovery by TV minimization. To fully comprehend the result of Theorem 1, and specifically the bound (5) , we first examine the bound (1).

In (1), we see that the error of the reconstruction is proportional to the noise level, ϵ, and is likewise comparable to if we had known and selected s largest absolute entries of the signal as the reconstruction. In fact, as argued in [1], one cannot hope to do better with $\mathcal{O}(s\log N)$ measurements; an error bound of this form is optimal (see, eg. [3, Chapters 4,11]). Note also when the signal is s-sparse, exact reconstruction with noiseless measurements is implied.

The preceding remarks imply (5) is near optimal, and improvement is possible only by reducing the log factor. To see this, note that (6) implies stable and robust gradient recovery, and that the error bound on this process is optimal. Now if we were able to improve on (5) (absent the log factor), then we would contradict the optimality of (6) since $\|\nabla X - \nabla X^\sharp\|_2 \lesssim \|X - X^\sharp\|_2$. For simplicity, we justify the last claim by proving an analogous result for signals in $\mathbb{C}^N$.

Let $x, x^\sharp \in \mathbb{C}^N$ and set $v = x - x^\sharp$. Then

$$\|\nabla v\|_2 = \|(v_{j+1} - z_j)_{j=1}^{N-1}\|_2 \leq \|(v_{j+1})_{j=1}^{N-1}\|_2 + \|(v_j)_{j=1}^{N-1}\|_2 \lesssim \|v\|_2.$$

The proof of Theorem 1 requires two main ingredients. The first step is to establish a tube constraint and a cone constraint satisfied by the difference $\nabla X - \nabla X^\sharp$. Provided these constraints are met, (6) and (7) is derived. The second ingredient is what Needell and Ward call their *strong Sobolev inequality* ([8, Theorem 9]), which is a bound of the form

$$\|X - X^\sharp\|_2 \lesssim \epsilon + \log\left(\frac{N^2}{s}\right)\frac{\|X - X^\sharp\|_{TV}}{\sqrt{s}}. \tag{8}$$

This, when combined with (7), gives stable and robust signal recovery. In other words, we first show under suitable conditions, establishing stable and robust gradient recovery can then imply stable and robust signal recovery.

The next proposition is the key to the first ingredient ([8, Proposition 3]:

Proposition 2. *Fix parameters $\gamma \geq 1$ and $\delta < 1/3$. Suppose $\mathcal{B}$ satisfies the RIP of order $5s\gamma^2$ and level δ, and suppose that the image V satisfies a tube constraint*

$$\|\mathcal{B}V\|_2 \lesssim \epsilon.$$

Suppose further that for a subset S, $|S| \leq s$, V satisfies the cone constraint

$$\|V_{S^c}\|_1 \leq \gamma\|V_S\|_1 + \beta.$$

Then

$$\|V\|_2 \lesssim \epsilon + \frac{\beta}{\gamma\sqrt{s}},$$

and

$$\|V\|_1 \lesssim \gamma\sqrt{s}\epsilon + \beta.$$

We can see that the last two inequalities are exactly (6), (7) with $\nabla(X - X^\sharp)$ in place of V, $\|\nabla X - (\nabla X^\sharp)_{\mathbf{s}}\|_1$ in place of β and $\gamma = 1$. Showing that $\nabla(X - X^\sharp)$ satisfies the tube constraint requires the more tedious set up of the multi-component linear operator of Theorem 5 in [8] which we will avoid here. The cone constraint follows from

$$\begin{aligned}
&\|(\nabla X)_S\|_1 + \|V_{S^c}\|_1 \\
&\qquad \le \|(\nabla X)_S - V_S\|_1 + \|V_S\|_1 + \|(\nabla X)_{S^c} - V_{S^c}\|_1 + \|(\nabla X)_{S^c}\|_1 \\
&\qquad = \|\nabla X - V\|_1 + \|(\nabla X)_{S^c}\|_1 + \|V_S\|_1 \\
&\qquad = \|\nabla X^\sharp\|_1 + \|(\nabla X)_{S^c}\|_1 + \|V_S\|_1 \\
&\qquad \le \|\nabla X\|_1 + \|(\nabla X)_{S^c}\|_1 + \|V_S\|_1,
\end{aligned}$$

where we've used $V = \nabla X - \nabla X^\sharp$ and the fact that $X, X^\sharp$ are feasible. The last step is to rearrange and identify S with an index set containing s largest absolute coefficients of ∇X.

The second ingredient is the strong Sobolev inequality (8). The derivation of this is nontrivial. Comparing with the classical discrete Poincaré (for mean-zero images, $\|X\|_2 \le \|X\|_{TV}$), we are asking in (8) for a significantly stronger bound (to hold for most $s < N$). This is facilitated with a heavy dose of the bivariate Haar transform. Specifically, one makes use of the fact that the decay rate of its $k$the largest absolute bivariate Haar coefficients of a mean-zero image X is at least $C\|X\|_{TV}/k$. The precise statement is the content of the next proposition ([8, Proposition 8]).

Proposition 3. *Suppose $X \in \mathbb{C}^{N\times N}$ is mean-zero and let $c_{(k)}(X)$ be the bivariate Haar coefficient of X have $k$the largest magnitude. Then for all $k \ge 1$,*

$$|c_{(k)}(X)| \lesssim \frac{\|X\|_{TV}}{k}.$$

The derivation of the strong Sobolev inequality is then the content of Theorem 9 of [8]. We include it here for completeness.

Lemma 4. *Let $\mathcal{B} : \mathbb{C}^{N\times N} \to \mathbb{C}^m$ be a linear map such that $\mathcal{B}\mathcal{H}^{-1}$ has the RIP of order $2s+1$ and level $\delta < 1$, where $\mathcal{H} : \mathbb{C}^{N\times N} \to \mathbb{C}^{N\times N}$ is the bivariate Haar transformation. If $V \in \mathbb{C}^{N\times N}$ satisfies $\|\mathcal{B}V\|_2 \le \epsilon$, then*

$$\|V\|_2 \lesssim \epsilon + \log\left(\frac{N^2}{s}\right)\frac{\|V\|_{TV}}{\sqrt{s}}.$$

The final step is simply to combine (7) with the strong Sobolev inequality applied to

$X - X^\sharp$:

$$\begin{aligned}\|X - X^\sharp\|_2 &\lesssim \epsilon + \log\left(\frac{N^2}{s}\right) \frac{\|X - X^\sharp\|_{TV}}{\sqrt{s}} \\ &\lesssim \epsilon + \log\left(\frac{N^2}{s}\right) \frac{\sqrt{s}\epsilon + \|\nabla X - (\nabla X)_{\mathbf{s}}\|_1}{\sqrt{s}} \\ &\lesssim \epsilon + \log\left(\frac{N^2}{s}\right) \frac{\|\nabla X - (\nabla X)_{\mathbf{s}}\|_1}{\sqrt{s}},\end{aligned}$$

as long as appropriate RIP conditions are satisfied.

We remark that the derivation and the form of (8) suggests that the log factor, $\log(N^2/s)$, is but an artifact of the proof technique. This is conjectured by Needell and Ward and we will seek verification of this claim in our numerical experiments.

A secondary result in their paper ([8, Theorem 6]) removes the extra log factor in the error bound. However, this comes at a cost of requiring the sensing matrix to have the RIP of order $\mathcal{O}(s \log^3 N)$. To our knowledge, these are to date, the best proven results.

We stress here that while the results of Theorem 1 is formulated for signals $X \in \mathbb{C}^{N \times N}$, an extension to signals $X \in \mathbb{C}^{N^d}$, $d \geq 2$, has been made by Needell and Ward in [7]. The proof strategy is of the same flavor as in the 2D case. The arguments follow the same structure as outlined above, but does require some work to generalize each step to hold in higher dimensions. We would recommend the interested reader to follow carefully the arguments laid out in [8] before moving to the higher dimensional case.

3.2 Stable and robust recovery from Fourier measurements

The next result that we will examine is from a paper by Poon [9]. It is a highly specialized in the sense that a subsampled Fourier operator is the only sensing mechanism that we consider. Below we state her main result that applies to 1D signals ([9, Theorem 2.1]):

Theorem 5. *For $N \in 2^J$, $J \in \mathbb{N}$, let F be the discrete Fourier transform and let ∇_p be the discrete gradient operator with periodic boundary, i.e.*

$$\nabla_p : \mathbb{C}^N \to \mathbb{C}^n, \quad (\nabla_p z)_j = \begin{cases} z_{j+1} - z_j, & j \in [N-1], \\ z_1 - z_{N-1}, & j = N. \end{cases}$$

Let $\delta \in (0,1)$ and let $S \subseteq [N]$, $|S| = s$. Let $x \in \mathbb{C}^N$. Let $\Omega_1, \Omega_2 \subseteq \{-N/2+1, \dots, N/2\}$ be such that $|\Omega_1| = |\Omega_2| = m$ with $m \gtrsim s \log(N)(1 + \log(\delta^{-1}))$.

Let Ω_1 be chosen uniformly at random and let $\Omega_2 = \{\omega_1, \dots, \omega_m\}$ have indices chosen i.i.d. according to the distribution

$$\Pr(\omega_k = n) = p(n), \quad \textit{where} \quad p(n) = \frac{C}{\log(N) \max\{1, |n|\}}, \quad n = -N/2+1, \dots, N/2,$$

and C is chosen so that $\sum p(n) = 1$. Let $\Omega = \Omega_1 \cup \Omega_2$, and suppose $y = P_\Omega F x + e$, $\|e\|_2 \leq \sqrt{m}\epsilon$.

If $x^\sharp$ is a minimizer of

$$\min_z \|z\|_{TV,p} \quad \textit{subject to} \quad \|y - P_\Omega F z\|_2 \leq \epsilon\sqrt{m},$$

then with probability at least $1-\delta$,

$$\|\nabla x - \nabla x^\sharp\|_2 \lesssim \epsilon\sqrt{s} + \mathcal{L}_2 \frac{\|\nabla x - (\nabla x)_{\mathbf{s}}\|_1}{\sqrt{s}}, \qquad \frac{\|x - x^\sharp\|_2}{\sqrt{N}} \lesssim \mathcal{L}_1 \left(\frac{\epsilon}{\sqrt{s}} + \mathcal{L}_2 \frac{\|\nabla x - (\nabla x)_{\mathbf{s}}\|}{s} \right), \tag{9}$$

where $\mathcal{L}_1 = \log^2(s)\log(N)\log(m)$ and $\mathcal{L}_2 = \log(s)\log^{1/2}(m)$.

(The seminorm $\|\cdot\|_{TV,p}$ above comes from using the discrete gradient operator ∇_p, i.e. $\|z\|_{TV,p} = \|\nabla_p z\|_1$. The operator P_Ω is a restriction operator that projects onto the entries indexed by Ω.)

The optimality of (9) is argued in the succeeding remark in the paper and will not be repeated here. In short, the bound is optimal up to log factors.

Now to get to the heart of this result. Again, it is stable and robust recovery by TV minimization. However, there notable key differences in the hypotheses of Theorem 5 and Theorem 1. Theorem 1 is a uniform recovery result and applies to sensing matrices that satisfies the RIP of order $\mathcal{O}(s)$. In Theorem 5, the sensing mechanism is a subsampled Fourier operator. Moreover, the sampling is not of the uniform variety. Theorem 5 uses a variable density sampling scheme in drawing the indices for Ω. The probability density p adopted in Theorem 5 was analyzed in [6] in conjunction with the partial Fourier operator. It was showed there (although under a nonstandard noise model) that this sampling scheme allows the use of the Fourier operator in combination with the Haar wavelet transform despite the coherence between the two bases.

Interestingly, in the same publication [9], Poon derives a weaker result when using purely uniform random sampling (otherwise with conditions identical to Theorem 5). Ignoring log factors, the error bound on the signal recovery, $\|x - x^\sharp\|_2$, is greater by a factor of s ([9, Theorem 2.3]). It is further conjectured that this is a fundamental limit on uniform sampling patterns, i.e. uniform sampling is inherently inferior to the variable sampling strategy given in Theorem 5.

For further details and an analogous 2D result to Theorem 5, the interested reader may refer to [9, Theorem 2.2] and other content within the same publication.

4 Numerical Experiments

Below we will present a collection of numerical examples. All examples in this section perform the recovery process by solving (TV min) (or the analogous 1D formulation) with the split Bregman algorithm [5]. See also Appendix A for a quick and easy introduction to the algorithm.

4.1 1D Experiments

In this section, we will consider reconstruction of signals $x \in \mathbb{C}^N$. For all experiments here, $N = 1024$ will be the signal length, the (gradient) sparsity will be $s = 20$, and the reconstruction done with m samples, where m is such that $m/N \approx 0.14$. To further describe our procedure, we turn to our first example. Codes used in our numerical experiments are provided in Appendix B.

4.1.1 Example 1

In our first example, we construct a gradient sparse signal and add to it i.i.d mean-zero Gaussian noise with standard deviation $\sigma_1 = 1/5$. We then take measurements with a subsampled Fourier operator and added to the measurements i.i.d mean-zero Gaussian noise with standard deviation $\sigma_2 = 1/20$. The command

```
N = 1024;stable_n_robust_1d(N, round(N/50), 140/N, ...
    'uniform',1e-3,'noise',1*5,0.05,'global', 8)
```

in MATLAB produces the plots in Figure 1.

To generate the plots in Figure 1, we must specify parameters such as noise level (σ_1, σ_2), signal length (even), sparsity level, sampling ratio, as well as the sampling strategy that determines the restriction operator P_Ω. Here, we have sampled uniformly the Fourier data of the signal (top left plot of Figure 1). The sampling map is provided in the third plot of the second column. Note the sampling ratio. Our procedure allows each frequency to be sampled with probability 0.14, so each instance will produce a difference value.

Figure 1 is but our first demonstration of recovery by TV. Next, we will examine more broadly aspects of 1D signal recovery by TV minimization under four different sampling strategies.

4.1.2 Sampling strategies

The quality of the recovery using four different sampling strategies will be compared.

1. Uniform random sampling: This is the most common (but certainly not the best) sampling strategy studied in the compressed sensing literature. The sampling set,

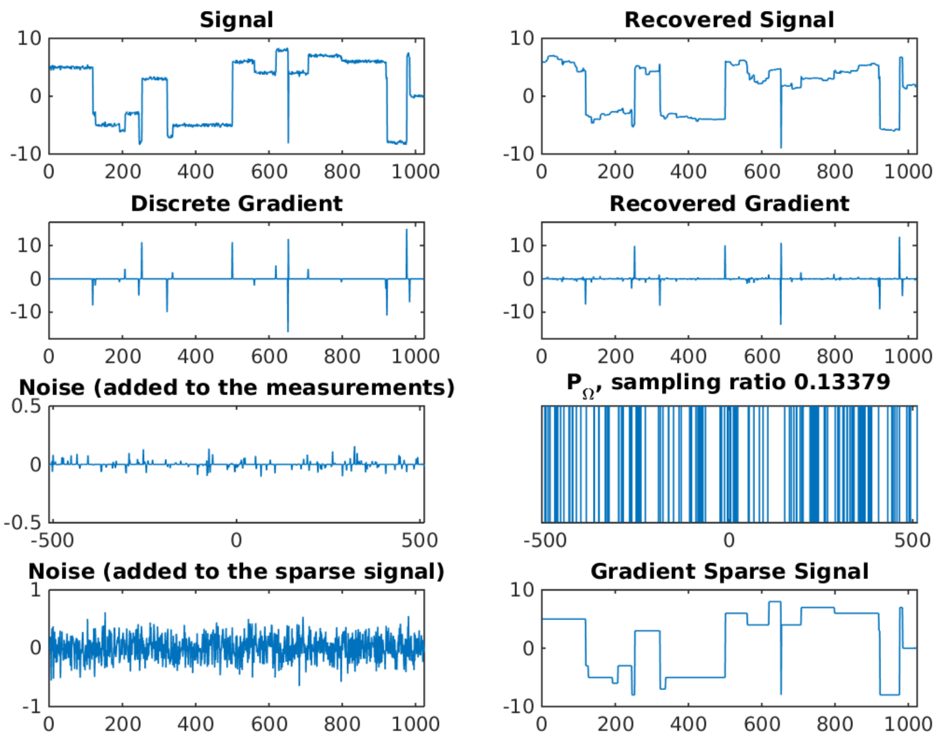

Figure 1: Reconstruction of the 1D signal (top left) using 13.4% of its Fourier coefficients. The coefficients have been chosen uniformly at random (third plot, second column) and is corrupted by noise (third plot, first column). The reconstructed signal (shown top right) has a relative error of 0.368.

which we denote Ω_U, is selected via the probability distribution

$$\Pr(k \in \Omega_U) = \frac{m}{N}, \quad \text{for } k = -\frac{N}{2}+1, \dots, \frac{N}{2}.$$

2. Low frequency sampling: This is the most straightforward. We simply sample the frequency k whenever $-m/2 \leq k \leq m/2 - 1$, and disregard frequencies outside this range. We denote this sampling set Ω_L.

3. Uniform and power decay density sampling: This is the strategy in the statement of Theorem 5. When we say power decay density, we are referring to the probability density p is the theorem. We get resp. m_1 and m_2 samples accordingly and form the sample set Ω_H. This we will refer to as our hybrid sampling strategy.

4. Power decay density sampling: Again, power decay density refers to the probability distribution in Theorem 5. We will investigate whether combination with a uniform sampling set is necessary, or whether it may be an artifact of the proof provided in [9], which uses the uniform sampling set to establish stable gradient recovery. We will refer to this strategy as power density sampling and denote the sampling set as Ω_P.

We also stipulate that in each strategy we sample the zero frequency.

Figure 2 shows a representative sampling map from each of the four sampling strategies.

4.1.3 Example 2

The results of this example are summarized in Table 1. We show two measures of error. The first is a direct comparison with the signal x. The second computes a gradient error. Let x' be the reconstructed signal. The quantities l2error and l2∇error are then defined as

$$\text{l2error} := \frac{\|x - x'\|_2}{\|x\|_2}, \quad \text{l2}\nabla\text{error} := \frac{\|\nabla x - \nabla x'\|_2}{\|\nabla x\|_2}.$$

For each sampling strategy, we run 200 experiments. Each experiments constructs a gradient sparse signal (similar to bottom right of Figure 1), samples, and reconstructs. The signal is sparse and no noise is added to the measurements.

The results in Table 1 say that capturing only the low frequencies is the weakest strategy. Losing all high frequency information hinders gradient and signal recovery. Comparison between the power density sampling and hybrid sampling shows a trade off between capturing fewer high frequency modes for more low frequency modes. The signal error is slightly reduced at the cost of an increase in the gradient error.

A natural followup then is to test with a compressible rather than sparse signal. This we do by repeating the above experiment with an additional step where random Gaussian

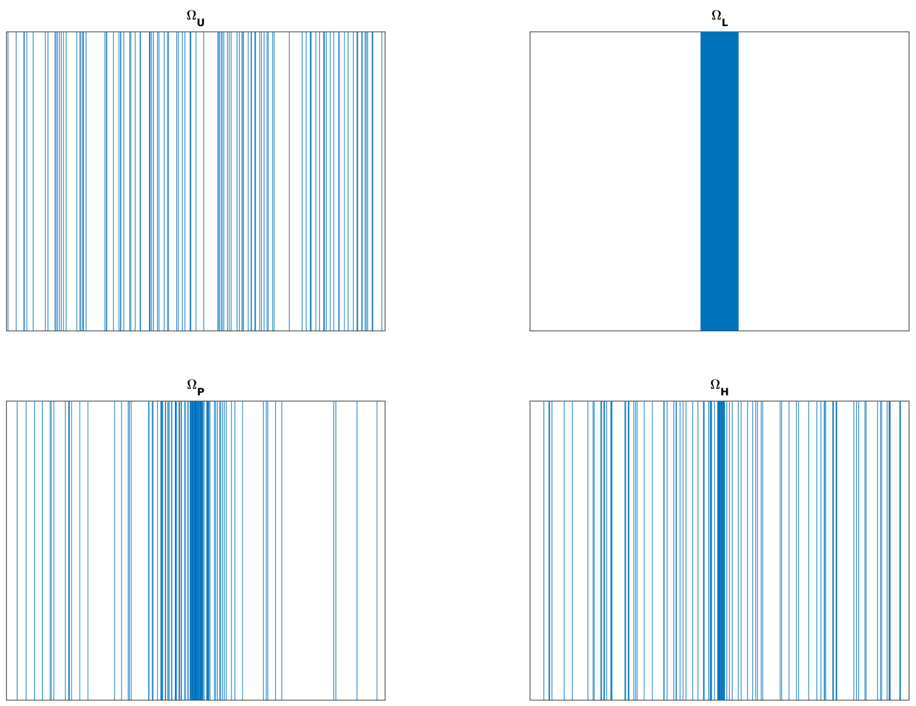

Figure 2: Sampling maps of the four sampling strategies. Each map was constructed with the target ratio $m/N \approx 0.10$. Dense sampling near the zero frequency is seen in Ω_L, Ω_P and Ω_H. Note also that Ω_H samples more evenly through the medium and high frequencies than Ω_P.

	Average number of samples	l2error ($\times 10^{-3}$)	l2∇error ($\times 10^{-3}$)
Ω_L	140	77.0	518
Ω_U	141	9.6	5.5
Ω_H	139	8.7	19.9
Ω_P	140	7.1	26.5

Table 1: Reconstruction of a gradient sparse signal.

	Average number of samples	l2error ($\times 10^{-3}$)	l2∇error ($\times 10^{-3}$)
Ω_U	141	367	275
Ω_H	139	107	240
Ω_L	140	81.9	478
Ω_P	140	66.4	225

Table 2: Reconstruction of a compressible signal.

	Average number of samples	l2error ($\times 10^{-2}$)	l2∇error ($\times 10^{-2}$)
Ω_U	140	370	299
Ω_H	140	96.6	249
Ω_P	140	57.2	205
Ω_L	140	29.6	116

Table 3: Reconstruction of a gradient sparse signal from noisy measurements.

noise (with standard deviation $\sigma_2 = 1/4$) is added to a sparse signal. A summary of the errors is in Table 2. Signal recovery by uniform sampling is now the worse by our error measure, whereas the error with low frequency sampling stayed relatively constant. However, in this scenario, we find power decay sampling to be the best in both signal recovery and gradient recovery. This indicates that sampling densely the low frequencies improves stability of the reconstruction.

For our last test of this section, we reconstruct gradient sparse signals from noisy measurements. Random Gaussian noise with $\sigma_1 = 5$ is added to the Fourier samples. The results in Table 3 now suggest complete low frequency sampling is the best. However, this is not entirely unexpected since the additional of noise disproportionately affects the higher frequencies. In other words, the high frequency modes are excited by the additional of noise, thus causing spurious oscillations. Dense sampling of the low frequencies circumvents this problem.

4.2 2D Experiment

In this section, experimentation will be with images under analogous sampling strategies outlined in Section 4.1.2. For concreteness, we presented representative sampling maps in Figure 3.

4.3 Example 3

We are now set to test reconstruction of images using the four sampling strategies: uniform, low frequency, hybrid, and power density sampling. In Figures 4 and 5, are reconstructions using uniform sampling and the hybrid sampling, resp. Note that we have chosen not to show the results from low frequency and power density sampling because they are

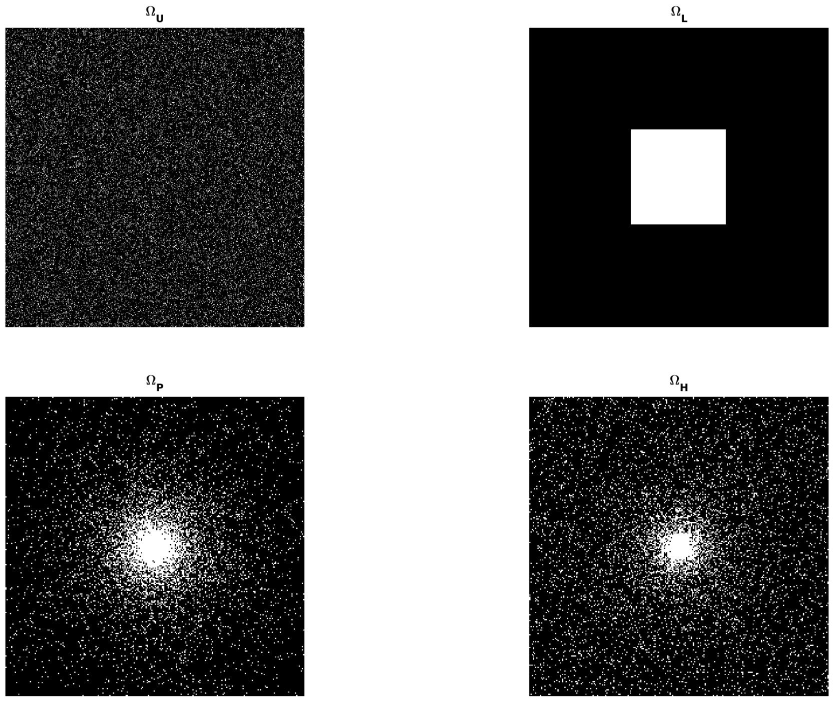

Figure 3: Sampling maps of the four sampling strategies. Each map was constructed with the target ratio $m/N^2 \approx 0.10$, where N now represents the number of pixels in the horizontal and vertical dimension. Dense sampling near the zero frequency is seen in strategies Ω_L, Ω_P and Ω_H.

nearly indistinguishable from hybrid sampling (fine features are blurred slightly with low frequency sampling). In Table 4, the l2error from Example 2 is computed and summarized. The results show that power density sampling outperforms the other strategies under consideration.

m/N^2	Ω_U	Ω_L	Ω_H	Ω_P
0.1	0.202	0.028	0.024	0.019
0.2	0.118	0.022	0.013	0.012
0.3	0.049	0.019	0.009	0.008

Table 4: Errors in image reconstruction. Each strategy sees a decrease in error when increasing the sampling ratio. At moderate sampling ratios, the performance of Ω_H, Ω_P are equal.

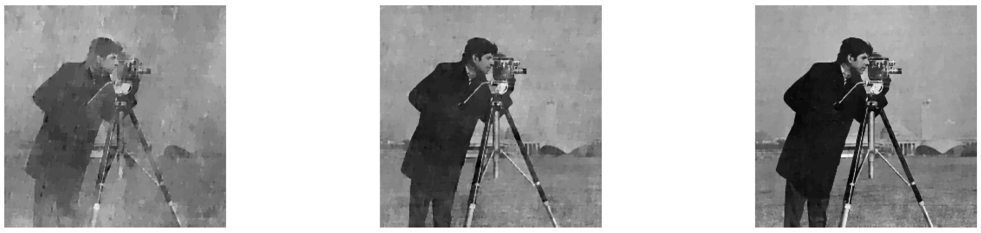

Figure 4: Image reconstruction from uniform random samples. Sampling ratios (from left to right): 0.10, 0.20, 0.30. Blurring and other visual artifacts are visible in each reconstruction.

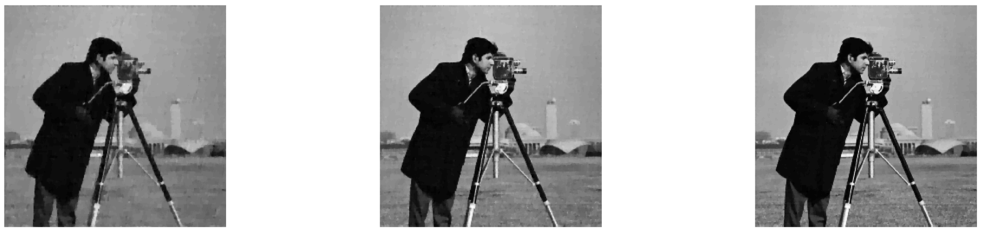

Figure 5: Image reconstruction from the hybrid sampling. Sampling ratios (from left to right): 0.10, 0.20, 0.30. Image quality is vastly superior to those in Figure 4.

5 Conclusion

Our survey examined some current results in compressed sensing by total variation minimization. Of particular interest was the role this regularizer plays in image processing applications in the compressed sensing setting, and why we can expect reasonable image reconstruction. We examined the claim made in [9] that we get improved stability by dense sampling of the low frequencies and found that it is indeed the case. Moreover, our testing suggests that combining power density sampling with uniform sampling as in Theorem 5 is unnecessary, and that uniform sampling was only in place to facilitate the proof.

A The Split Bregman Algorithm

We give here an brief description of the split Bregman algorithm. For simplicity, we will outline the details for signals in 1D. For details in 2D, we refer to reader to Section 4.2 of [5].

The split Bregman algorithm is a method for finding solutions to certain classes of ℓ_1-minimization problems. In this survey, the minimization problem of interest is of the form

$$\min_z \|\nabla z\|_1 \quad \text{such that} \quad \|y - P_\Omega F z\|_2 \leq \epsilon, \tag{10}$$

where P_Ω is the restriction operator that projects onto the indices in Ω, F is the Fourier operator, and $\nabla \equiv \nabla_p$, as defined in Theorem 5.

First, the problem (10) is reformulated as a sequence of unconstrained problems,

$$z^{k+1} = \operatorname*{argmin}_z \|\nabla z\|_1 + \frac{\mu}{2}\|y^k - P_\Omega F z\|_2^2, \tag{11}$$

$$y^{k+1} = y^k + y - P_\Omega F z^{k+1}. \tag{12}$$

Note the introduction of a parameter μ. Next, we rewrite the minimization problem in (11) to fit the split Bregman framework. This requires the introduction of the variable $d \sim \nabla z$ and a Bregman parameter b. This gives

$$\min_{z,d} \|d\|_1 + \frac{\mu}{2}\|y - P_\Omega F z\|_2^2 + \frac{\lambda}{2}\|d - \nabla z - b\|_2^2.$$

This last expression is now suitable for decomposition into subproblems using the split Bregman algorithm. The resulting subproblems are

$$\begin{aligned} z^{k+1} &= \operatorname*{argmin}_z \frac{\mu}{2}\|y^k - P_\Omega F z\|_2^2 + \frac{\lambda}{2}\|d^k - \nabla z^k - b^k\|_2^2 \\ d^{k+1} &= shrink(\nabla z^k + b^k, 1/\lambda), \end{aligned}$$

where the shrinkage operator is defined as

$$shrink(x, \alpha) = \frac{x}{\|x\|_1} \max(\|x\|_1 - \alpha, 0).$$

The first subproblem can be shown to be equivalent to solving

$$Cz^{k+1} := (\mu F^T P_\Omega^T P_\Omega F - \lambda \triangle + \gamma I) z^{k+1} = \mu F^T P_\Omega^T y^k + \lambda \nabla^T (d^k - b^k) =: rhs^k, \quad (13)$$

where $\triangle$ represents a discrete Laplacian operator. Herein lies one of the key efficiencies of the split Bregman. Due to the splitting of the ℓ_2 and ℓ_1 quantities into separate subproblems, the first is differentiable and admits a solution that can be solved for efficiently. The system to be inverted on the left-hand side of (13) is independent of k, and is circulant and thus diagonalized by the DFT. The second subproblem is a routine calculation.

Adding in updates of the Bregman parameter b and the quantity y^k, we have the following algorithm:

```
Inputs:  $z^0 = F^T y, y^0 = y, d^0 = b^0 = 0.$
While $\|y^k - P_\Omega F z^k\|_2^2 > \epsilon^2$
    For $i = 1$ to $n$
        Solve $Cz^{k+1} = rhs^k$
        $d^{k+1} = shrink(\nabla z^{k+1} + b^k, 1/\lambda)$
        $b^{k+1} = b^k + \nabla z^{k+1} - d^{k+1}$
    end
    $y^{k+1} = y^k + y - P_\Omega F z^{k+1}$
end
```

Codes implementing the split Bregman is openly available on Goldstein's web page ([4]) and is also presented as `sb_1d` and `sb_2d` in our codes section.

For more details see [5] and reference therein.

B Codes

B.1 1D Codes

```
function[] = stable_n_robust_1d(N, s, m, sampling, epsilon, noise, ...
    scale, std, sig_type, cmpct)
% stable_n_robust_1d.m
%    Tests stability and robustness of recovery of 1D signals by TV
%    minimization
%
% Inputs:
%   N: signal length, N must be even
%   s: (gradient) sparsity of the signal
%   m: sampling ratio
%   sampling strategy. See test_mrics1d.m for supported strategies.
%   epsilon: recon stopping crit, ie. stop when norm(Az-y) < epsilon.
%   noise: adding noise to the signal, 'no', 'noise'
%   scale: scale magnitude of noise, noise <- noise/scale.
%   std: standard deviation of (gaussian) measurement noise
%   sig_type: global or local signal, 'global', 'local'
%   cmpct: local signal support length; given as multiple of s
% Outputs:
%   Generates a collection of plots
%
% Ex. N = 1024;
% stable_n_robust_1d(N, round(N/50), 140/N, 'low', 1e-3, 'noise', 1*5, ...
    0.1, 'global', 8)
% stable_n_robust_1d(N, round(N/50), 140/N, 'uniform', 1e-3, 'no', 1, 0, ...
    'local', 8)
%
% Dec 2015, Kevin Chow
%
if nargin < 5
    error('Specify first 5 inputs');
elseif nargin < 7
    noise = 'no'; scale = 1;
elseif nargin < 8
    std = 0;
elseif nargin < 10
    sig_type = 'global'; cmpct = 0;
end
if mod(N, 2) == 1
    error('Even signal length only')
end

% % % % % % % % % % % % % % % % % % % % % % % % 1D gradient sparse signal:
switch sig_type
    case 'global'
        sig = grad_sparse(s, N, 10, 0);
    case 'local'
        sig = cmpct_grad_sparse(s, cmpct, N, 10, 0);
end

sig0 = sig;
sig_grad = sig(2:end)-sig(1:end-1);

switch noise
    case 'noise'
```

```
        noise = randn(size(sig)); noise = noise/scale;
        sig = sig+noise;
    case 'no'
        noise = zeros(size(sig));
end

% % % % % % % % % % % % % % % Call test_mrics1d to sample and reconstruct:
[recovered, R, mnoise, outer]=recovery_1d(sig, sampling, m, epsilon, std);
recovered = real(recovered);
rec_grad = recovered(2:end)-recovered(1:end-1);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % Some statistics:
imgName = '1D gradient sparse';
l2error = norm(recovered - sig, 2)/norm(sig, 2);
l2error_grad = norm(rec_grad - sig_grad, 2)/norm(sig_grad, 2);
m = nnz(R);
disp([imgName, ':', 10, 9, 'Sampling ratio = ', num2str(m/numel(sig), ...
    3), ...
    10, 9, 'l2error = ', num2str(l2error), ...
    10, 9, 'grad_l2error = ', num2str(l2error_grad)]);

% % % % % % % % % % % % % % % % % % % % build a figure to display results:
figure;
% suptitle('TV recovery')
ax1 = subplot(4,2,1); plot(sig);
axis([0 N min(sig0)-2 max(sig0)+2]); title('Signal');

ax3 = subplot(4,2,3); plot(sig_grad);
axis([0 N min(sig_grad)-2 max(sig_grad)+2]); title('Discrete Gradient');

ax5 = subplot(4,2,5); plot(-N/2+1:N/2, fftshift(mnoise));
axis([-N/2+1 N/2 min(1.1*min(mnoise),-0.5) max(1.1*max(mnoise), 0.5)]);
title('Noise (added to the measurements)');

ax7 = subplot(4,2,7); plot(noise);
axis([0 N min(1.1*min(noise),-1) max(1.1*max(noise),1)]);
title('Noise (added to the sparse signal)');

ax2 = subplot(4,2,2); plot(recovered);
axis([0 N min(sig)-2 max(sig)+2]); title('Recovered Signal');

ax4 = subplot(4,2,4); plot(rec_grad);
axis([0 N min(sig_grad)-2 max(sig_grad)+2]); title('Recovered Gradient');

ax6 = subplot(4,2,6); stem(-N/2+1:N/2, fftshift(R),'d','marker','none');
axis([-N/2+1 N/2 0 1]); title(['P_\Omega, sampling ratio ', ...
    num2str(m/numel(sig))]);
set(ax6,'ytick',[])

ax8 = subplot(4,2,8); plot(sig0);
```

```
axis([0 N min(sig0)-2 max(sig0)+2]); title('Gradient Sparse Signal');

linkaxes([ax1,ax2,ax8],'xy');
linkaxes([ax3,ax4],'xy');
```

```
function[recovered,R,mnoise,outer]=recovery_1d(sig,sampling,m,epsilon,std)
% recovery_1d.m
%   Given a 1D signal, get measurements by a subsampled Fourier operator
%   and perform CS reconstruction by TV min. using the split Bregman (SB).
%
% Inputs:
%   sig: 1D signal
%   sampling: sampling strategy
%   m: sampling ratio = # of measurements / signal length
%   epsilon: tolerance, halts SB when norm(Az - y) < epsilon
%   std: standard deviation of gaussian noise added to measurements
% Outputs:
%   recovered: signal recovered by TV minimization using the SB
%   R: sampling map.
%   mnoise: noise added to measurements
%   outer: # of iters of the SB outer loop
%
% Modified from test_mrics.m, Dec 2015, Kevin Chow
%
if nargin < 4
    error('First 4 arguments must be specified');
elseif nargin < 5
    std = 0;
end
% % % % mu, lambda, epsilon are parameters for the split Bregman algorithm:
% If one finds outer (see mrics_mod1d.m) is large, adjust mu, lambda.
mu = 1.1;
lambda = 0.1;
N = length(sig);

% % % % % % % % % % % % % % % % % % % % % % % Build the sampling matrix, R:
% 3 cases:
%   uniform random sampling             'uniform'
%   low freq sampling                   'low'
%   power law density sampling          'power'
%   unif + power law density sampling   'unif power'
%
% Might have to adjust first parameter of sample.m to get the right sampling
% ratio.
switch sampling
    case 'uniform'
        R = rand(size(sig));
        R = double(R < m);
    case 'low'
        low = round(m*N/2);
        R = zeros(size(sig));
        R(1:low) = 1;
        R(end-low+1:end) = 1;
    case 'power'
        C = pn1dC(N);
        R = sample_m(round(2*m*N), N, @(n) pn1d(n,N,C));
    case 'unif power'
        C = pn1dC(N);
        R = rand(size(sig));
        R1 = double(R < m/2);
        R2 = sample_m(round(m*N), N, @(n) pn1d(n,N,C));
        R = double(R1|R2);
end
R(1) = 1; % zero frequency sample

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % Form the CS data:
F = R.*fft(sig)/sqrt(numel(sig));

% % % % % % % % % % % % % % % % % % % % % % % Add noise to the measurements:
% We add iid gaussian with mean zero and standard deviation std.
mnoise = R.*randn(size(F))*std; F = F+mnoise;

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % Recover the signal:
[recovered, outer] = sb_1d(R, F, mu, lambda, 10, epsilon);
% outer
```

```
function[u, outer] = sb_1d(R, f, mu, lambda, nInner, sigma)
% sb_1d.m
%   Perform CS reconstruction on 1d signal by the split Bregman (SB).
%
% Inputs:
%  R: Sampling map. R(k)=1 if the frequency k is known, and 0 otherwise.
%  F: Fourier data. The values of this vector should have R.*R == F
%           in this matrix properly, then you should have (R.*F==F).
% mu- The parameter on the fidelity term in the Split Bregman method.  The
%           best choice will depend on how the data is scaled.
% lambda - The coefficient of the constraint term in the Split Bregman
%       model. For most problems, I suggest using lambda=mu.
% nInner - This determines how many "inner" loops the Split Bregman method
%       performs (i.e. loop to enforce the constraint term).  I suggest
%       using nInner = 30 to be safe.  This will usually guarantee good
%       convergence, but will make things a bit slow.  You may find that you
%       can get away with nInner = 5-10
% sigma -
%
% Modified from mrics.m, Dec 2015, Kevin Chow
```

```
%

    [rows,cols] = size(f);

% % % % % % % % % % % % % %   % Reserve memory for the auxillary variables
    f0 = f;
    u = zeros(rows,cols);
    x = zeros(rows,cols);
    bx = zeros(rows,cols);

% % % % % % % % % % % % % % % % % % % % % % % % % % % %   % Build Kernels
    scale = sqrt(rows*cols);
    murf = ifft(mu*(conj(R).*f))*scale;

    uker = zeros(rows,cols);
    uker(1) = 2; uker(2) = -1; uker(end) = -1;
    uker = mu*(conj(R).*R) + lambda*fft(uker);

% % % % % % % % % % % % % % % % % % % % % % % % % % % %   % Reconstruction
% Split Bregman inner and outer iterations
    outer = 0; % count while loop its
    while norm(R.*fft(u)/scale-f0, 2)/norm(f0, 2) > sigma && outer < 1000
        for inner = 1:nInner;
          % update u
            rhs = murf+lambda*Dxt(x-bx);
            u = ifft(fft(rhs)./uker);

          % update x
            dx = Dx(u);
            x = shrink(dx+bx, 1/lambda);

          % update bregman parameter
            bx = bx+dx-x;
        end

        f = f+f0-R.*fft(u)/scale;
        murf = ifft(mu*R.*f)*scale;
        outer = outer+1;
    end
end

function d = Dx(u)
  % Gradient
    d = zeros(size(u));
    d(:, 2:end) = u(:, 2:end) - u(:, 1:end-1);
    d(:, 1) = u(:, 1) - u(:, end);
end

function d = Dxt(u)
  % Gradient "transposed"
```

```
    d = zeros(size(u));
    d(:, 1:end-1) = u(:, 1:end-1) - u(:, 2:end);
    d(:, end) = u(:, end) - u(:, 1);
end

function[xs] = shrink(x, lambda)
  % Shrinkage (4.4)-(4.6) of Split Bregman paper
    s = sqrt(x.*conj(x));
    ss = s-lambda;
    ss = ss.*(ss>0);

    s = s+(s<lambda);

    xs = ss.*x./s;
end
```

```
function[x, dx, v] = grad_sparse(s, NN, kmax, plt)
% grad_sparse.m
%   This creates a gradient sparse signal of length NN and sparsity s.
%   Signal deviation from zero is no more than kmax.
%
% Inputs:
%   s, sparsity level
%   N, signal length. Must have N > s.
%   kmax, positive integer.
%   plt, set as 1 to plot,
% Returns:
%   x, the gradient sparse signal
%   dx, discrete gradient of x. This is s-sparse.
%   v, location of jumps.
% Ex:
%   grad_sparse(10, 512);
%
% Created Nov 2015, Kevin Chow
%

if nargin < 2
    error('Require at least 2 inputs');
elseif nargin < 3
    kmax = 10; plt = 1;
elseif nargin < 4
    plt = 1;
elseif nargin > 4
    disp('Extra inputs ignored');
end
if s ≥ NN
    error('Desired sparsity level is greater than signal length')
end
```

```

    x = zeros(1, NN);
    v = [randsample(NN, s)' NN]; v = sort(v);
    kk = randi([-kmax, kmax], [1, s+1]);

    pp = 1;
    for jj = 1:NN
        if jj ≤ v(pp),
            x(jj) = kk(pp);
        else
            pp = pp+1;
            x(jj) = kk(pp);
        end
    end
    dx = [x(2:end)-x(1:end-1) 0];

    if plt == 1
        fignum = 234;
        figure(fignum)
        clf
        plot(x, 'x-', 'markersize', 4);
        hold on
        axis([0 NN min(min(x), min(dx))-2 max(max(x), max(dx))+2])
        plot(dx, 'ro')
        plot(v, dx(v), '+', 'markersize', 12);
        legend('Signal', 'Discrete gradient')
    end
end
```

```
function[x, dx, v] = cmpct_grad_sparse(s, k, NN, kmax, plt)
% cmpct_grad_sparse.m
%   This creates a gradient sparse signal of length N and sparsity s that
%   is supported over at most k*s indices. We'll require k*s < NN.
%   Signal deviation from zero is no more than kmax.
%
% Inputs:
%   s, sparsity level
%   k, positive integer.
%   N, signal length. Must have N > s.
%   kmax, positive integer.
%   plt, set as 1 to plot,
% Returns:
%   x, the gradient sparse signal
%   dx, discrete gradient of x. This is s-sparse.
% Ex:
%   cmpct_grad_sparse(10, 8, 512);
%
% Created Nov 2015, Kevin Chow
```

```
%

if nargin < 3
    error('Require at least 3 inputs');
elseif nargin < 4
    kmax = 10; plt = 1;
elseif nargin < 5
    plt = 1;
elseif nargin > 5
    disp('Extra inputs ignored');
end
if k*s ≥ NN
    error('Desired support length is greater than signal length')
end

    x = grad_sparse(s-2, k*s, kmax, 0);
    x0 = randi([0 NN-k*s]);
    x = [zeros(1, x0-1) x zeros(1, NN-x0+1-k*s)]; x = x(1:NN);
    dx = [x(2:end)-x(1:end-1) 0];
    v = find(dx≠0);

    if plt == 1
        fignum = 444;
        figure(fignum)
        clf
        plot(x, 'x-', 'markersize', 4);
        hold on
        axis([0 NN min(min(x), min(dx))-2 max(max(x), max(dx))+2])
        plot(dx, 'ro')
        plot(v, dx(v), '+', 'markersize', 12);
        legend('Signal', 'Discrete gradient')
    end
end
```

```
function[samples] = sample_m(m, N, prob)
% sample_m.m
%   Draw m samples from frequencies -N/2+1,...,N/2 according to the
%   distribution prob.
% Inputs:
%   N: frequencies -N/2+1,..,N/2,
%   m: sampling ratio of m/N.
% Outputs:
%   samples: vector from {0,1}^N. Approx m ones.
% ex.
%   N = 128;
%   abab = sample_m(10, N, @(n) pn1d(n, N));
%   stem(-N/2+1:N/2, fftshift(abab)); % to visualize.
%
```

```
    mm = rand(1, m); mm = sort(mm);
    samples = zeros(1, N);
    p = prob(0);
    idx_at = nnz(mm ≤ p)+1;
    if idx_at > 1, samples(1) = 1; end
    if idx_at > m, return; end
    for jj = 1:N/2-1
        pjj = prob(jj);
        loc = [jj+1 N-jj+1];
        for ii = loc;
            p = p+pjj;
            if mm(idx_at) < p
                samples(ii) = 1;
                while idx_at ≤ m && mm(idx_at) ≤ p
                        idx_at = idx_at + 1;
                end
                if idx_at > m, return; end
            end
        end
    end
    samples(N/2) = 1;
end
```

```
function[pn, n] = pn1d(n, N, C)
% pn1d.m
%   We have a 1d probability distribution.
%       p(n) = C*(log(N)*max(1, abs(n)))^{-1}, for n = -N/2+1, ... , N/2.
%   The constant C = C(N) is chosen so that p is a probability distribution.
%   The statement p(n) is the probability that frequency n is to be sampled.
%
% This function takes inputs n, a frequency, N, the number of frequencies
% under consideration and constant C, the normalizing constant, and returns
% p(n), the probability that we sample the frequency n.
%
% Ref1: Poon, On the role of TV in compressed sensing (2015)
% Ref2: Krahmer and Ward, Stable and robust sampling stragies for
%   compressive sensing (2014)
%
% Dec 2015, Kevin Chow
%
if nargin < 2
    error('Require minimum 2 inputs');
elseif nargin < 3
    C = log(N)/(1 + 2/N + 2*sum(1./(1:N/2-1)));
end
if max(n) > N/2 || min(n) < -N/2+1
    error('n outside of expected frequencies');
end
```

```

    n = abs(n);
    pn = C./(log(N).*max(1, n));
end
```

```
function[C] = pn1dC(N)
% pn1dC.m
%   Returns the normalizing constant needed for pn1d.m
%
if nargin < 1, error('Supply N'); end
    C = log(N)/(1 + 2/N + 2*sum(1./(1:N/2-1)));
end
```

B.2 2D Codes

```
function[R] = st_n_rob_2d(img, m, sampling, epsilon, noise, scale, std)
% st_n_rob_1d.m
%    Tests stability and robustness of recovery of images by TV
%    minimization
%
% Inputs:
%   img: specify test image
%   m: sampling ratio
%   sampling strategy. See recovery_2d.m for supported strategies.
%   epsilon: recon stopping crit, ie. stop when norm(Az-y) < epsilon.
%   noise: adding noise to the image, 'no', 'noise'
%   scale: scale magnitude of noise, noise <- noise/scale.
%   std: standard deviation of (gaussian) measurement noise
% Outputs:
%   Generates a collection of plots
%
% Ex. N = 1024;
% st_n_rob_2d('pears', 0.15, 'power, 1e-3, 'noise', 1, 0);
% st_n_rob_2d('peppers', 0.15, 'uniform', 1e-3, 'no', 1, 0);
%
% Dec 2015, Kevin Chow
%
if nargin < 3
    error('Specify first 3 inputs');
elseif nargin < 4
    epsilon = 1e-3; noise = 'no'; scale = 1; std = 0;
elseif nargin < 5
    noise = 'no'; scale = 1; std = 0;
elseif nargin < 6
    scale = 1; std = 0;
```

```
elseif nargin < 7
    std = 0;
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % Select image:
% Available in Matlab distribution are
%   spine.tif, pears.png, ... see toolbox/images/imdata
switch img
    case 'spine'
        name=img; img=imread('spine.tif'); img=img(1:end-1,63:end-62);
    case 'pears'
        name=img; img=imread('pears.png'); img=rgb2gray(img);
    case 'peppers'
        name=img; img=imread('peppers.png'); img=rgb2gray(img);
    case 'boats'
        name=img; img=imread('boats.png');
    case 'cameraman'
        name=img; img=imread('cameraman.tif');
    otherwise
        error('No such image');
end
% 'squarify' the image; limitation of sample_m2d.m
N = min(size(img)); N = N-mod(N,2); img = img(1:N,1:N);
img = im2double(img)*255;
img0 = img;

% % % % % % % % % % % % % % % % % % % % % % % % % % Add noise to the image:
switch noise
    case 'noise'
        noise = randn(size(img)); noise = noise/scale;
        img = img+noise;
    case 'no'
        noise = sparse(size(img));
end

% % % % % % % % % % % % % % % % % Call recovery_2d to sample and reconstruct:
[rec, R, F, outer]=recovery_2d(img, sampling, m, epsilon, std);
% outer

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % Some statistics:
l2error = norm(rec - img, 2)/norm(img, 2);
m = nnz(R);
disp([name, ':', 10, 9, 'Sampling ratio = ', num2str(m/numel(img), 3), ...
    10, 9, 'l2error = ', num2str(l2error)]);

% % % % % % % % % % % % % % % % % % % % build a figure to display results:
figure;
% suptitle('TV recovery')
ax1 = subplot(2,2,1);
imagesc(abs(ifft2(F))); colormap('gray');
```

```
title('Image reconstruction with unknowns set to zero');
axis equal, axis tight, axis off

ax2 = subplot(2,2,2);
imagesc(abs(rec)); colormap('gray');
title('Split Bregman Recovery'); axis equal, axis tight, axis off

ax3 = subplot(2,2,3);
imagesc(abs(img)); colormap('gray');
title(name); axis equal, axis tight, axis off

ax4 = subplot(2,2,4);
imagesc(abs(fftshift(R))); colormap('gray');
title(['R, sampling ratio ', num2str(m/numel(img))]);
axis equal; axis tight; axis off

linkaxes([ax1, ax2, ax3],'xy')
```

```
function[recovered,R,F,outer]=recovery_2d(img,sampling,m,epsilon,std)
% recovery_2d.m
%   Given an image, get measurements by a subsampled Fourier operator
%   and perform CS reconstruction by TV min. using the split Bregman (SB).
%
% Inputs:
%   img: image
%   sampling: sampling strategy
%   m: sampling ratio = # of measurements / numel(img)
%   epsilon: tolerance, halts SB when norm(Az - y) < epsilon
%   std: standard deviation of gaussian noise added to measurements
% Outputs:
%   recovered: signal recovered by TV minimization using the SB
%   R: sampling map.
%   F: (Noisy) Fourier samples
%   outer: # of iters of the SB outer loop
%
% Modified from test_mrics.m, Dec 2015, Kevin Chow
%
if nargin < 4
    error('First 4 arguments must be specified');
elseif nargin < 5
    std = 0;
end
% % % % mu, lambda, epsilon are parameters for the split Bregman algorithm:
% If one finds outer (see sb_2d.m) is large, adjust mu, lambda.
mu = 0.5; lambda = 0.1;

% % % % % % % % % % % % % % % % % % % % % % % % Build the sampling matrix, R:
% 4 cases:
```

```
%   uniform random sampling              'uniform'
%   low freq sampling                    'low'
%   power law density sampling           'power'
%   unif + power law density sampling    'unif power'
%
% Might have to adjust first parameter of sample_m2d.m to get the right
% sampling ratio using 'power', 'unif power'.
switch sampling
    case 'uniform'
        R = rand(size(img)); R = double(R < m);
    case 'low'
        R = zeros(size(img));
        low = sqrt(m);
        km = round(low*size(img, 1)/2);
        kn = round(low*size(img, 2)/2);
        R(1:km, 1:kn) = 1;
        R(end-km:end, end-kn:end) = 1;
        R(1:km, end-kn:end) = 1;
        R(end-km:end, 1:kn) = 1;
    case 'power'
        N = size(img, 1); C = pn2dC(N);
        R = sample_m2d(round(m*N^2), N, @(n,k) pn2d(n,k,N,C));
    case 'unif power'
        N = size(img, 1); C = pn2dC(N);
        R = rand(N, N);
        R1 = double(R < m/2);
        R2 = sample_m2d(round(m/2*N^2), N, @(n,k) pn2d(n,k,N,C));
        R = double(R1|R2);
    otherwise
        error('Specify available sampling strategy');
end
R(1, 1) = 1; % zero frequency

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % Form the CS data:
F = R.*fft2(img)/sqrt(numel(img));

% % % % % % % % % % % % % % % % % % % % % % % Add noise to the measurements:
% We add iid gaussian with mean zero and standard deviation std.
mnoise = R.*randn(size(F))*std; F = F+mnoise;

% % % % % % % % % % % % % % % % % % % % % % % % % % % % Recover the image:
[recovered, outer] = sb_2d(R, F, mu, lambda, 10, epsilon);
```

```
function[u, outer] = sb_2d(R, f, mu, lambda, nInner, epsilon)
% sb_2d.m
%   Perform CS reconstruction on 2D image by the split Bregman (SB).
%
% Inputs:
```

```
%  R: Sampling map. R(n,k)=1 if the frequency (n,k) is known, and 0 o.w.
%  f: Fourier data. The values of this vector should have R.*R == f
%           in this matrix properly, then you should have (R.*f==f).
%  mu:The parameter on the fidelity term in the Split Bregman method.  The
%           best choice will depend on how the data is scaled.
%  lambda: The coefficient of the constraint term in the Split Bregman
%      model. For most problems, I suggest using lambda=mu.
%  nInner: This determines how many "inner" loops the Split Bregman method
%      performs (i.e. loop to enforce the constraint term).  I suggest
%      using nInner = 30 to be safe.  This will usually guarantee good
%      convergence, but will make things a bit slow.  You may find that you
%      can get away with nInner = 5-10
%  epsilon: stopping criterion
%
% Modified from mrics.m, Dec 2015, Kevin Chow
%
    [rows,cols] = size(f);

  % Reserve memory for the auxillary variables
    f0 = f;
    u = zeros(rows,cols);
    x = zeros(rows,cols);
    y = zeros(rows,cols);
    bx = zeros(rows,cols);
    by = zeros(rows,cols);

  % Build Kernels
    scale = sqrt(rows*cols);
    murf = ifft2(mu*(conj(R).*f))*scale;

    uker = zeros(rows,cols);
    uker(1, 1) = 4;  uker(1, 2) = -1;
    uker(2, 1) = -1;   uker(rows, 1) = -1;
    uker(1,cols)=-1;
    uker = mu*(conj(R).*R) + lambda*fft2(uker);

  % Do the reconstruction
    outer = 0; % count while loop its
    while norm(R.*fft2(u)/scale-f0, 2)/norm(f0, 2) > epsilon
        for inner = 1:nInner;
          % update u
            rhs = murf+lambda*Dxt(x-bx)+lambda*Dyt(y-by);
            u = ifft2(fft2(rhs)./uker);

          % update x and y
            dx = Dx(u);
            dy = Dy(u);
            [x,y] = shrink2( dx+bx, dy+by,1/lambda);

          % update bregman parameters
```

```
            bx = bx+dx-x;
            by = by+dy-y;
        end

        f = f+f0-R.*fft2(u)/scale;
        murf = ifft2(mu*R.*f)*scale;
        outer = outer+1;
    end
end

function d = Dx(u)
    d = zeros(size(u));
    d(:, 2:end) = u(:, 2:end) - u(:, 1:end-1);
    d(:, 1) = u(:, 1) - u(:, end);
end

function d = Dxt(u)
    d = zeros(size(u));
    d(:, 1:end-1) = u(:, 1:end-1) - u(:, 2:end);
    d(:, end) = u(:, end) - u(:, 1);
end

function d = Dy(u)
    d = zeros(size(u));
    d(2:end, :) = u(2:end, :) - u(1:end-1, :);
    d(1, :) = u(1, :) - u(end, :);
end

function d = Dyt(u)
    d = zeros(size(u));
    d(1:end-1, :) = u(1:end-1, :) - u(2:end, :);
    d(end, :) = u(end, :) - u(1, :);
end

function [xs,ys] = shrink2(x,y,lambda)
    s = sqrt(x.*conj(x)+y.*conj(y));
    ss = s-lambda;
    ss = ss.*(ss>0);

    s = s+(s<lambda);
    ss = ss./s;

    xs = ss.*x;
    ys = ss.*y;
end
```

```
function[samples] = sample_m2d(m, N, prob)
% sample_m2d.m
%  Draw m samples from frequencies
%      (n,k) \in {-N/2+1,...,N/2}x{-N/2+1,...,N/2}
%  according to the distribution prob.
% ex.
%  N = 128; C = pn2dC(N);
%  abab = sample_m2d(round(N^2/10), N, @(n,k) pn2d(n,k,N,C));
%  figure, imagesc(fftshift(abab)); colormap('gray');
%  axis equal, axis tight, axis off, title('Sampling map')
%
% Inputs:
%  m,N: sampling ratio of m/N, N describes range of frequencies
%  prob: probability distribution; function handle.
% Returns NxN matrix of 0s and 1s.
% Ref: Poon, On the role of TV in compressed sensing (2015)
% Created Dec 2015, Kevin Chow
%
 % Generate m unif random numbers; Create storage.
  m = round(2.6*m);
  mm = rand(1, m); mm = sort(mm);
  samples = zeros(N, N);

  p = prob(0,0);
  idx_at = nnz(mm ≤ p)+1;
  if idx_at > 1, samples(1,1) = 1; end
  if idx_at > m, return; end

 % Sub-blocks in the for loop does the following resp.:
 % Sample frequencies of the type (\pm j,0), (0, \pm j)
 % Sample frequencies of the type (\pm j, \pm j)
 % Sample frequencies of the type (N/2, \pm j), (\pm j, N/2)
 % Sample frequencies of the type (\pm j, \pm ii), (\pm ii, \pm j)
  for j = 1:N/2-1
    row1 = [j+1 1 N-j+1 1 j+1];
    pj = prob(j, 0);
    for jj = 1:4
      p = p+pj;
      if mm(idx_at) < p
        samples(row1(jj), row1(jj+1)) = 1;
        while idx_at ≤ m && mm(idx_at) ≤ p
            idx_at = idx_at + 1;
        end
        if idx_at > m, return; end
      end
    end

    row2 = [j+1 j+1 N-j+1 N-j+1 j+1];
    pj = prob(j, j);
    for jj = 1:4
      p = p+pj;
      if mm(idx_at) < p
```

```
                    samples(row2(jj), row2(jj+1)) = 1;
                    while idx_at ≤ m && mm(idx_at) ≤ p
                            idx_at = idx_at + 1;
                    end
                    if idx_at > m, return; end
                end
            end

            row3 = [j+1 N/2+1 N-j+1 N/2+1 j+1];
            pj = prob(j, N/2);
            for jj = 1:4
                p = p+pj;
                if mm(idx_at) < p
                    samples(row3(jj), row3(jj+1)) = 1;
                    while idx_at ≤ m && mm(idx_at) ≤ p
                            idx_at = idx_at + 1;
                    end
                    if idx_at > m, return; end
                end
            end

            for ii = j+1:N/2-1
                row4 = [j+1 ii+1 N-j+1 N-ii+1 j+1 N-ii+1 N-j+1 ii+1 j+1];
                pj = prob(j, ii);
                for jj = 1:8
                    p = p+pj;
                    if mm(idx_at) < p
                        samples(row4(jj), row4(jj+1)) = 1;
                        while idx_at ≤ m && mm(idx_at) ≤ p
                            idx_at = idx_at + 1;
                        end
                        if idx_at > m, return; end
                    end
                end
            end
        end
        samples(N/2+1, N/2+1) = 1;
end
```

```
function[pnk] = pn2d(n, k, N, C)
% pn2d.m
%   We have a 2d probability distribution.
%   p(n, k) = C*(log(N)*max(1, n^2+k^2))^{-1}, for k,n = -N/2+1, ... , N/2.
%   The constant C = C(N) is chosen so that p is a probability distribution.
%   The statement p(n, k) is the probability that frequency (n, k) is to be
%   sampled.
%
% This function takes inputs (n, k), a frequency, N, the number of
% frequencies under consideration and constant C, the normalizing constant,
% and returns p(n), the probability that we sample the frequency n.
%
% Ref1: Poon, On the role of TV in compressed sensing (2015)
% Ref2: Krahmer and Ward, Stable and robust sampling stragies for
%   compressive sensing (2014)
%
if nargin < 3
    error('Require minimum 3 inputs');
elseif nargin < 4
    C = 1+10/N^2;
    for ell = 1:N/2-1
        C = C+16./(N^2+4*ell^2) + 6./ell^2 + ...
            8*sum(1./((ell+1:N/2-1).^2+ell^2));
    end
    C = C/log(N); C = 1/C;
end
if max(n) > N/2 || min(n) < -N/2
    error('(n,k) outside of expected frequencies');
elseif max(k) > N/2 || min(k) < -N/2
    error('(n,k) outside of expected frequencies');
end
    pnk = C./(log(N).*max(1, n.^2+k.^2));
end
```

```
function[C] = pn2dC(N)
% pn2dC.m
%   Returns the normalizing constant needed for pn2d.m
%
if nargin < 1, error('Supply N'); end
    C = 1+10/N^2;
    for ell = 1:N/2-1
        C = C+16./(N^2+4*ell^2) + 6./ell^2 + ...
            8*sum(1./((ell+1:N/2-1).^2+ell^2));
    end
    C = C/log(N); C = 1/C;
end
```

References

[1] E.J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, Feb 2006.

[2] Emmanuel J. Candes, Justin K. Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006.

[3] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*. Birkhäuser, 2013.

[4] Tom Goldstein. Split Bregman. `http://www.ece.rice.edu/~tag7/Tom_Goldstein/Split_Bregman.html`. Accessed: 2015-10-14.

[5] Tom Goldstein and Stanley Osher. The split Bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[6] F. Krahmer and R. Ward. Stable and robust sampling strategies for compressive imaging. *Image Processing, IEEE Transactions on*, 23(2):612–622, Feb 2014.

[7] Deanna Needell and Rachel Ward. Near-optimal compressed sensing guarantees for total variation minimization. *IEEE Transactions on Image Processing*, 22(10):3941–3949, 2013.

[8] Deanna Needell and Rachel Ward. Stable image reconstruction using total variation minimization. *SIAM Journal on Imaging Sciences*, 6(2):1035–1058, 2013.

[9] Clarice Poon. On the role of total variation in compressed sensing. *SIAM Journal on Imaging Sciences*, 8(1):682–720, 2015.