Compressed Sensing

- The idea is that we have an algorithm to perfectly(?) reconstruct a sparse signal by sampling well under the Nyquist rate.

- Say we have a signal represented by the vector $\vec{u} \in \mathbb{R}^n$. We sample the signal by projecting it onto some basis, but we do not take n samples. Instead, we only take m samples where $m < n$, represented the matrix $A_{m \times n}$. Each row in A corresponds to a projection of u onto some basis, and each column details a different sample. In taking the samples, we obtain the vector $\vec{b} \in \mathbb{R}^m$.

- With specific reference to our robotics project, we have an n pixel terrain and the robot needs to reconstruct the terrain by taking m samples. Each sample is obtained as follows: the robot roams over some pixels, integrating the reflectance values it obtains to give the total reflectance sensor reading obtained by traveling the ENTIRE path. The vector $\vec{u}$ is the the 'solution' that we're trying to find; it contains the actual reflectance sensor values(?) of the terrain. A will be the matrix where the ith entry in a row gives the magnitude of projection of the ith pixel in $\vec{u}$ and each column will be a different sample. $\vec{b}$ will be the vector containing the integrated data that the robot obtained. It is an $m \times 1$ matrix, where the ith entry is the total reflectance sensor value obtained from the ith sample.

- So essentially, the problem we're trying to solve is

  $$A\vec{u} = \vec{b}$$

  But this is an underdetermined system, so we need to add a constraint. Initially we wanted to solve by minimizing the L_0 norm, which essentially intends to find the solution u by minimizing the number of non-zero entries.

  $$\min ||\vec{u}||_{l_0} \text{ s.t } A\vec{u} = \vec{b}$$

  But this problem is too hard. So instead we solve by minimizing the L_1 norm, which consists of minimizing the sum of the non-zero entries.

  $$\boxed{\min_u ||\vec{u}||_{l_1} \text{ s.t } A\vec{u} = \vec{b}} \tag{1}$$

- The way to achieve this is to convert it into an unconstrained problem. Split Bregman iteration solves the problem

  $$\min_{(u,d)} |d| + H(u) + \frac{\lambda}{2} \text{ s.t } d = \Phi(u)$$

where $H(\cdot)$ and $|\Phi(\cdot)|$ are convex functionals, by making it unconstrained as follows:

$$\min_{(u,d)} |d| + H(u) + \frac{\lambda}{2}||d - \Phi(u)||_2^2$$

Refer to page 329-331 of "A Fast Method For L1-Regularized Problems" by Tom Goldstein and Stanley Osher to see the exact method to solve this. $\Phi(u)$ is L1-norm we wish to minimize, while $H(u)$ is the data fitting term. **In our case we set $\Phi(u) = u$ and $H(u) = \lambda_1||A\vec{u} - \vec{b}||_2^2$ and convert (1) to an unconstrained problem**. Then implementing the split bregman iteration by following the algorithm on Goldstein 331, we have

1. Step 1

   $$\min_u \lambda_1||A\vec{u} - \vec{b}||_2^2 + \lambda_2||d_k - \vec{u} - b_k||_2^2 \tag{2}$$

   which is a least-squared problem. Note b_k and d_k are just some vectors in $\mathbb{R}^n$ we are using in the iteration. We solve this problem by taking the gradient and setting it to zero. In doing so, we obtain the linear equation

   $$(\lambda_1 A^T A + \lambda_2 I)\vec{u} = \lambda_1 A^T\vec{b} + \lambda_2(d_k - b_k)$$

   Solving this for u gives the minimum $\vec{u}$ sought after by the problem, in this iteration. An alternative is to use the MATLAB function

   ```
   fminsearch(fun, x0)
   ```

   to minimize (2).

2. Step 2

   Next we look to minimize d, so we use the shrink function, as given in Goldstein 330,

   $$\min_d shrink\left(\vec{u} + b_k, \frac{1}{\lambda_2}\right)$$

   Note: The $\vec{u}$ we use here is the updated $\vec{u}$ from step 1.

3. Step 3

   Now we increment b, simply as

   $$b_{k+1} = b_k + \vec{u} - d_k$$

Implementing this in MATLAB we find that we are able to reconstruct $\vec{u}$ successfully.