CS7640: Advanced Image Processing **Danny Perry**

Non-linear Image Denoising

1 Introduction

One of the classic problems in image analysis is image denoising. We are interested in the specific approaches to image denoising that preserve edges. A classic approach to edge-preserving non-linear denoising was introduced by [6], where edge preservation was achieved by modeling the problem as an anisotropic diffusion problem. Later work in [7] introduced the so called "ROF Model" where edge-preserving denoising could be done my minimizing the total variation of the image. Total variation minimization is based on the principle that noisy images have high total variation, which is the integral of the norm of the gradient. By reducing the total variation subject to being a close match to the original image removes unwanted detail (hopefully noise), while preserving important details like edges [3].

Here we evaluate three popular algorithms in minimization of total variation:

1. TV using Chambolle's Algorithm
2. TV using Primal-Dual Algorithm
3. Anisotropic TV using Split-Bregman Iterations

2 Methods

Here we briefly describe the specific motivation for each algorithm, how they work in theory, and the specific engineering decisions used in the implementations.

2.1 Total Variation Minimization

Here we motivate and describe the method of total variation. As briefly described above varation can be described like so:

$$\int |\nabla u| dxdy$$

Where we want to minimize the varation while maintaining some similarity with the original image, we hope to minimize the following constrained problem:

$$\min \int |\nabla u| dxdy \text{ s.t. } \int (u - u_0)^2 dxdy = \sigma^2 \tag{1}$$

This specific formulation is referred to colloquially as the ROF model. When originally presented in [7], a gradient descent type algorithm was suggested to minimize the Euler-Lagrange equations of the problem. While the results were good, the specific algorithm for solving it was numerically difficult, due to the possibility of $|\nabla u| = 0$. Later work, presented below attempted to improve on that approach.

2.2 Chambolle's Algorithm

In [4] a new formulation was presented introducing a so called "dual problem", which addressed the possible zero gradients by changing the formulation into a constrained maximization problem, ie

$$\begin{aligned}\int |\nabla u| dxxy &= \max_{|p|\leq 1} \int p \nabla u dxdy \\ &= \max_{|p|\leq 1} \int (\nabla \cdot p) u dxdy\end{aligned}$$

The second equation is obtained via integration by parts. In this form, the problem can be solved in a two-step approach:

1. Gradient descent to maximize p:

$$p^{k+1} = p^k - \Delta t \left[\nabla(\nabla \cdot p^k) + \lambda \nabla u_0 \right] \tag{2}$$

2. Solution to the tv minimization:

$$u = f + \lambda \nabla \cdot \hat{p} \tag{3}$$

In practice this two step approach is repeated until desirable results are reached.

Here the implementation was done entirely in C++ using the ITK framework. ITK was used primarily for image and vector data structures, I/O, and some infrastructure for parallelization. Any actual numerics were rewritten for this project, including gradient and divergence operations. A reader interested in following the code would be directed to primarily read the respective "GenerateData()" and "ThreadedGenerateData()" methods of the .hxx files of the classes described below, as that is where the action takes place - the rest is infrastructure. A README.txt file in the code directory describes how to build the executable.

In order to parallelize the algorithm, it was decomposed into a few distinct parallel steps over the image, with a single threaded iteration loop. The structure follows directly the algorithm structure:

1. ChambolleFilter.h,.hxx - the main loop
2. ChambolleDualFilter.h,.hxx - the parallel dual solution
3. ChambollePrimalFilter.h,.hxx - the parallel primal solution
   DivergenceFilter.h,.hxx - the parallel divergence computation
   UnitGradientFilter.h,.hxx - the parallel unit gradient computation

As discussed in class, I found it important to "balance" the one-sided derivatives used, so that the result ended up "on the grid" and not in the "gap". For example, in this solution I use a "right-sided" difference for the gradient and a "left-sided" difference for the divergence.

The implementation was done primarily from in class notes, which some details taken from the original paper.

It's also worth noting that because of the double loop needed to obtain quality results, this algorithm is the slowest of the three algorithms investigated. Specific results will be discussed below.

2.3 Primal-Dual Algorithm

A more efficient solution to the primal and dual problems called the Primal-Dual algorithm was described in [8]. Instead of solving the dual problem to completion (2) and then using that solution in the primal (3), the Primal-Dual algorithm solves both problems simultaneously in a sort of combined gradient descent.

In a single main iteration two steps are used to find the solution to the primal and dual equations:

1. Dual step:
$$p^{k+1} = P_p(p^k + \tau_k \lambda A^T u^k) \tag{4}$$
Where $P_p(z)$ is the projection back onto p.

2. Primal step:
$$u^{k+1} = u^k - \theta_k(1/\lambda A p^{k+1} + u^k - u_0) \tag{5}$$

Again, this was implemented in a parallel fashion using C++ and ITK. A single thread loop controlled the main iteration, while separate parallel processes compute each step. Here is the specific break down:

1. PrimalDualFilter.h,.hxx - the main loop
2. DualFilter.h,.hxx - the parallel dual step
3. PrimalFilter.h,.hxx - the parallel primal step
   UnitGradientFilter.h,.hxx - parallel unit gradient computation

Again, the "balance" of the derivatives was very important here. Similar to above the gradient used a "right-sided" difference, while the divergence was computed using a "left-sided" difference.

While I found it necessary to compute the initial p unit gradient field, all other gradient and divergence calculations are done in the dual and primal step filters themselves.

I primarily used the original paper to do the implementation.

This algorithm was much faster than Chambolle's, but specific results are shown below.

2.4 Split-Bregman Algorithm

The most recent of those approaches discussed here, is the Split-Bregman approach described in [5]. While the two methods discussed above are quite similar, this approach takes a slightly different route. Also, while the above algorithms minimize isotropic total variation, the algorithm implemented here minimizes an anisotropic total variation. The same paper [5] describes an isotropic version of Split-Bregman, but is not discussed in this report.

Instead of solving the problem described in 1, the anisotropic Split-Bregman addresses this similar anisotropic problem:
$$\arg\min_u \left[|du/dx|_1 + |du/dy|_1 + \frac{\mu}{2}||u - u_o||_2^2 \right] \tag{6}$$

The basic algorithm is as follows:

1. A single Gauss-Seidel (or similar) step to solve the L_2 problem.

$$u^{k+1} = G(u^k, d^k, b^k)$$

2. A shrinkage step

$$d_x^{k+1} = \text{shrink}(\nabla_x u^{k+1} + b_x^k, 1/\lambda)$$
$$d_y^{k+1} = \text{shrink}(\nabla_y u^{k+1} + b_y^k, 1/\lambda)$$

3. A Bregman update

$$b_x^{k+1} = b_x^k + (\nabla_x u^{k+1} - d_x^{k+1})$$
$$b_y^{k+1} = b_y^k + (\nabla_y u^{k+1} - d_y^{k+1})$$

This was also implemented in C++ using ITK. This algorithm was surprisingly simple to implement. It's the only algorithm where I simply coded up the algorithm, compiled, and it worked. This may have been due to it being the last one implemented, but I'd like to think it's somewhat due to the simplicity and robustness of the approach. It was also surprising how fast it is. When I first completed the code I thought something must be wrong, but it just converges very quickly.

To parallelize this code, I broke the algorithm into two distinct parallel steps, the Gauss-Seidel step and a combined shrinkage and Bregman update step, with a single iteration loop:

1. SplitBregman.h,.hxx - the main loop
2. GaussSeidelFilter.h,.hxx - the parallel Gauss-Seidel step
3. ShrinkBregmanFilter.h,.hxx - the parallel Shrink and Bregman update step GradientFilter.h,.hxx - a parallel gradient filter

This code was mainly implemented using the original paper [5], however a few details like edge conditions, etc. I obtained from Goldsteins implementation available for download at his website [2].

3 Results

Each algorithm was run against several images. The results of three of those images are show below: the cameara man image (Figures 3,4), a sailboat image (Figures 1,2), and a slice of a seismic image (Figure 5).

For the cameraman and sailboat images Gaussian noise with $\sigma^2 = 100$ was added to the image. This was done to make the denoising effect more apparent.

For the slice of seismic volume, the image was equalized to bring out the horizons. Denoising was applied with the goal of making the horizons appear more continuous. The seismic image is a slice taken from the Netherlands data set found on the OpendTect data repository [1].

3.1 Chambolle's Algorithm

Chambolle's algorithm was the slowest of the algorithms by far - mostly due to it's use of two loops. The bar graph in Figure 6 details the speed comparison. While Primal-Dual and Split-Bregman were comparable, both performed much faster than Chambolle.

As mentioned before, Chambolle's main cost are the double loops. The inner loop would normally take about 200-300 iterations to converge for the first few outer loop iterations, then it would take only a few iterations (under 10) to converge for the remaining outer loop iterations. Convergence criteria of the inner loop was computed using the difference of result with the previous - when the result started converging less than an epsilon, the loop was terminated.

The outer loop would converge in about 10-20 iterations. Visual correctness was used to determine convergence of the outer loop. The resulting total iterations required usually came to about 1000 iterations, which is the big reason for Chambolle's poor speed.

The quality of Chambolle's algorithm is good. It would be difficult to argue whether the results are much better or worse than from the other algorithms. For example Figures 1b,2b show the result and difference from noisy image of Chambolle's method. As you can see the result looks good and the difference image isn't obviously better than the other difference images.

Numerically, Chambolle was the trickiest to get right. With certain step sizes, it is quite easy for Chambolle's algorithm to blow up and not converge.

For these experiments I chose a $\lambda = 0.248$ as recommended in the Primal-Dual paper [8]. I also used an initial dual step size of $\Delta t = 0.0001$, but then updated that step size each iteration as described below. The dual step size was chosen proportional to the maximum gradient of the divergence of p, as indicated in class. In other words, take the maximum:

$$\max q = |\nabla(\nabla \cdot p) + \lambda \nabla u_0|$$

Then the step size was:

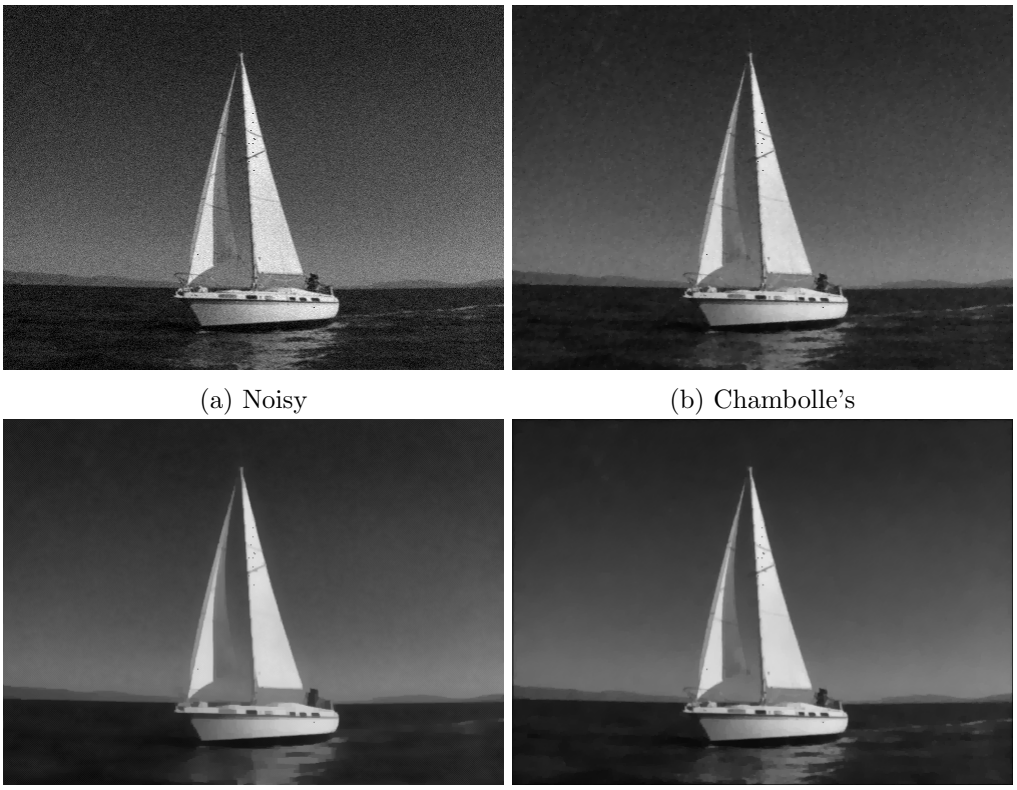
$$\Delta t = \frac{1}{2q}$$

3.2 Primal-Dual Algorithm

The Primal-Dual algorithm is very fast. As Figure 6 shows, its averaged runtime was under a second. It is very comparable to Split-Bregman and they only differed by a few tenths of a second. It would be interesting to see how they compare on some larger images, unfortunately time limitations don't allow for investigating that further right now.

Primal-Dual also had very good quality results. As mentioned before, it is somewhat difficulat to choose a "best" quality image from those shown. For example Figures 3c,4c show the result of running Primal-Dual on the camera man image. As you can see the denoised images looks good. The difference images shows that we probably coud have stopped the iterations a little sooner, as the edges seem to be smoothed more than the other algorithms. Perhaps using a quantitative stopping criteria could have helped in that regard.

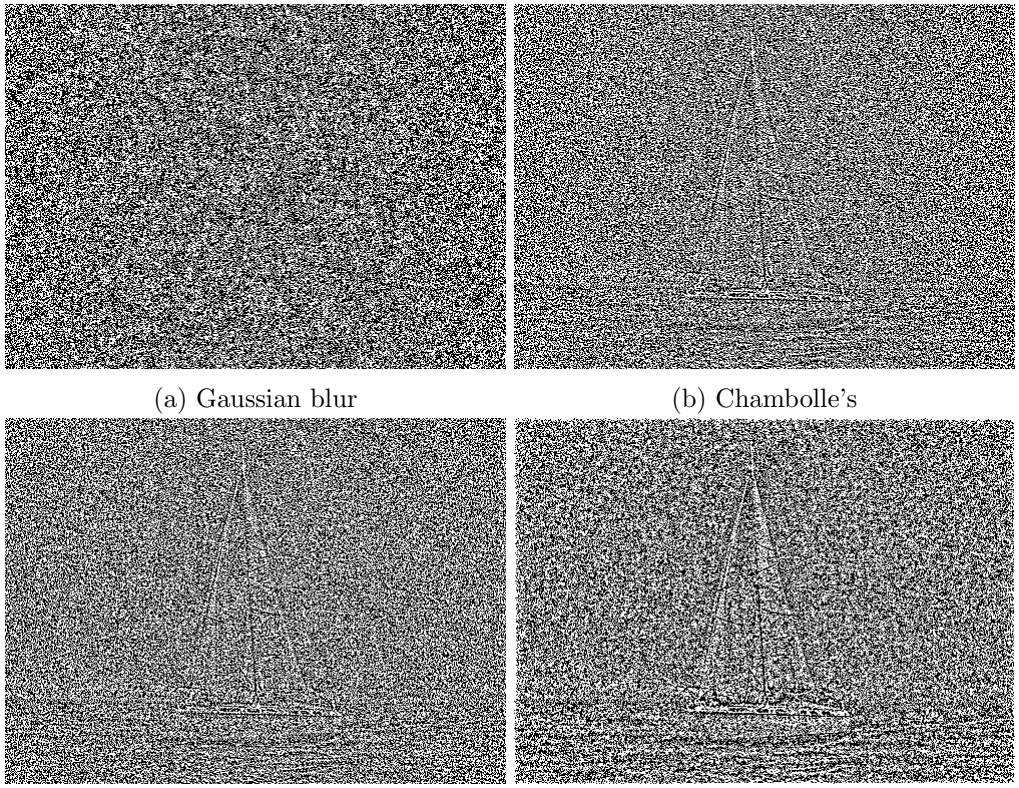
Numerically Primal-Dual was better behaved than Chambolles - it wasn't quite as hard to choose some parameters to make the algorithm work.



(a) Noisy (b) Chambolle's

(c) Primal-Dual (d) Split-Bregman

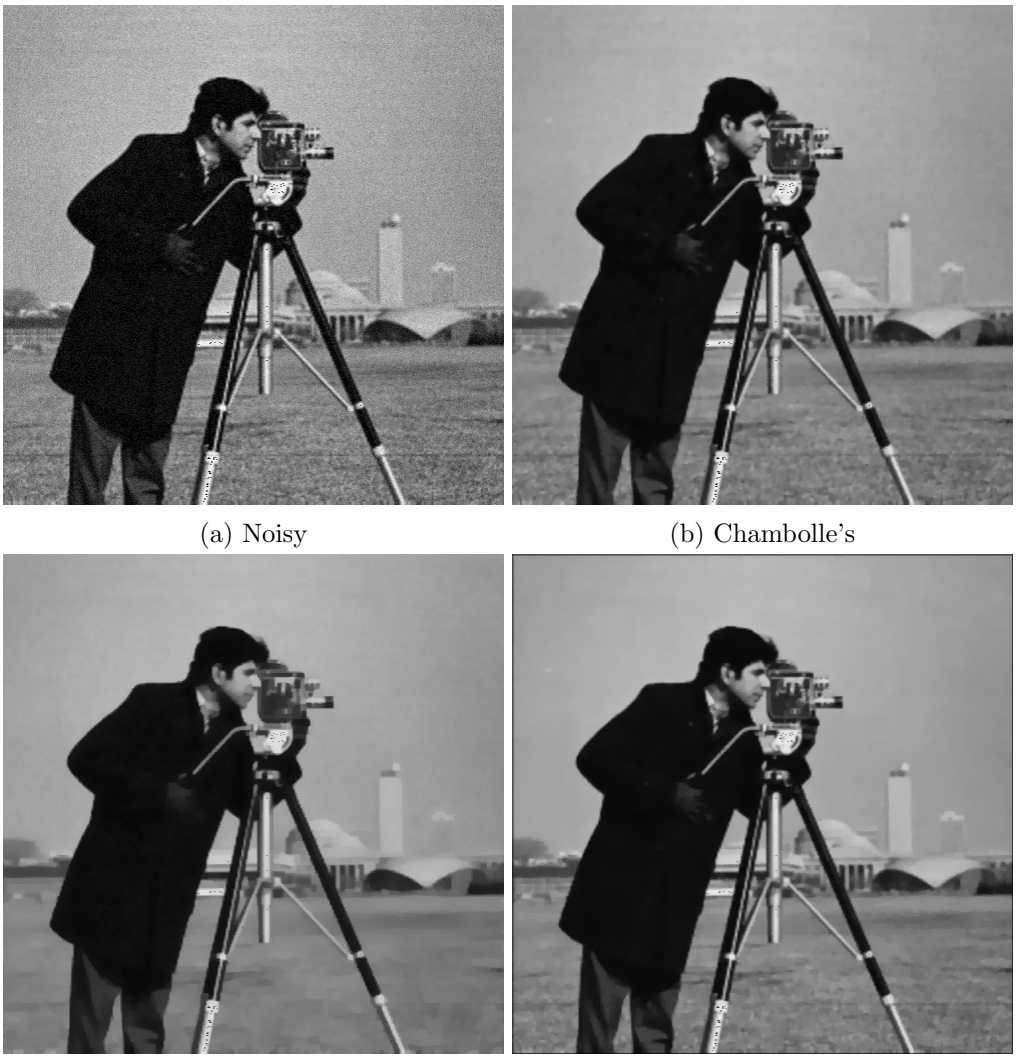
Figure 1: Results of each algorithm on a noisy image of a sailboat.



(a) Gaussian blur (b) Chambolle's

(c) Primal-Dual (d) Split-Bregman

Figure 2: Diff of each result with the noisy image, (a) shows the result of a standard Gaussian blur denoising using a width of 4.



(a) Noisy (b) Chambolle's

(c) Primal-Dual (d) Split-Bregman

Figure 3: Results of each algorithm on a noisy image of a cameraman.

I chose $\lambda = 0.0415$. The Primal-Dual paper suggested a way of estimating λ, but instead I've just used one of the values they reported, which seems to have performed fine. I chose the initial $\tau = 0.2$, where τ and θ were updated each time step using the the following formulas:

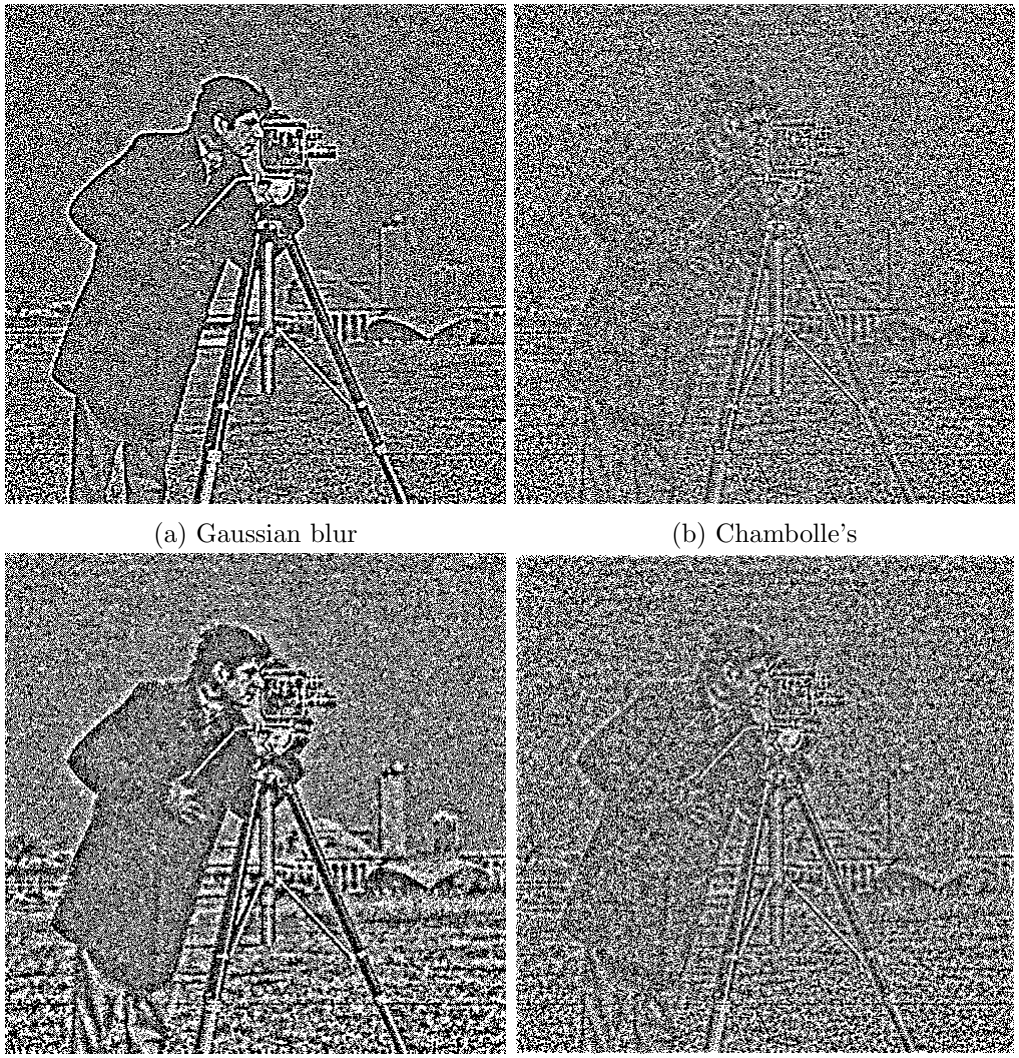
$$\tau = 0.2 + k * 0.08\theta \qquad = (0.5 + 5/(15 + k))/\tau$$

where k is the iteration. These choices seem to perform fine on the test images used.

3.3 Split-Bregman Algorithm

Split-Bregman was the fastest algorithm, again shown in Figure [**?**]. As mentioned previously, this algorithm slightly outperformed Primal-Dual by a few tenths of a second.

The quality of Split-Bregman is very good, which is very exciting for such a fast algorithm!



(a) Gaussian blur (b) Chambolle's

(c) Primal-Dual (d) Split-Bregman

Figure 4: Diff of each result with the noisy image, (a) shows the result of a standard Gaussian blur denoising using a width of 4.

Another very appealing feature Split-Bregman is it's simplicity and Numerical tolerance. I chose $\mu = 0.1$ without any prior information and the algorithm converged very quickly. It was surprising how forgiving and quick the algorithm is.

It's important to note that for this project I only implemented the Anisotropic Split-Bregman. In the future it would be of interested ot compare the isotropic version as well.

Even though this is the anistropic version, the results don't appear very different from the Primal-Dual and Chambolle's isotropic solutions.

Looking at Figure 5d, you can see the result of Split-Bregman on a slice of seismic sensor data. Looking at the data, one can see that it might be of interested to a geologist or geophysicist to see the specific horizons in the data. To that end the image was first histogram equalized to bring out the contrast. However with the constrast also emphasizes some of the discontinuities in the horizons.

I was curious to see how the total variation denoising performed on the data. The results appear encouraging. There doesn't appear to be any bad blurring across edges, and the roughness of the original equalized image appears to be smoothed out. A more detailed analysis shows that there are some blurring of the data, especially in regions where the lines are more unclear. It would be interesting to see how the algorithm performs on the volume data directly.

4 Conclusion

This was a very interesting project. It was very enlightening to learn first hand some of the problems and strengths of the different algorithms. From class discussions and reading the papers, I knew that Primal-Dual was faster than Chambolle's but I didn't realize how drastic the improvement was. Additionally it's interesting to see how robust and fast Split-Bregman is. It's a shame I don't have time to pursue the topics deeper, it would have been interesting to continue comparing across different measurements, and especially to incorporate other non-linear methods like Perona-Malik and the Isotropic Split-Bregman.

References

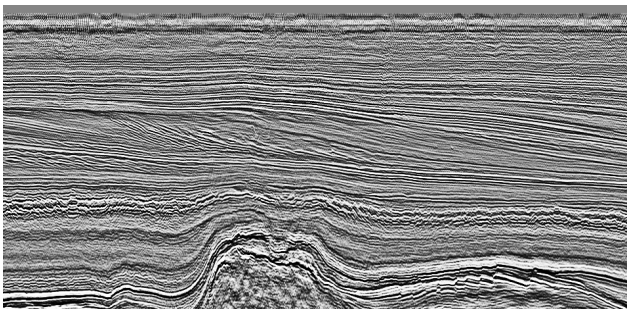
[1] Opendtect seismic repository. `http://www.opendtect.org/index.php/share-seismic-data/osr.html`. Accessed: 02/25/2013.

[2] Split bregman: project page. `http://tag7.web.rice.edu/Split_Bregman.html`. Accessed: 02/25/2013.

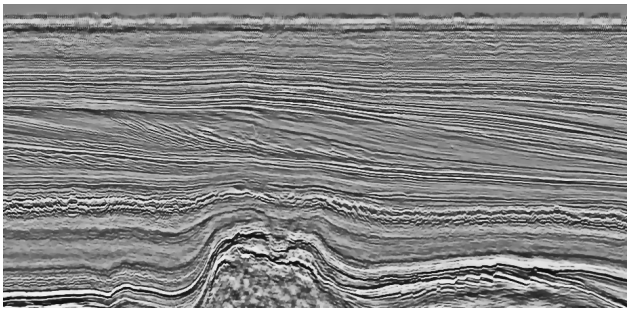
[3] Total variation denoising. `http://en.wikipedia.org/wiki/Total_variation_denoising`. Accessed: 02/25/2013.

[4] Antonin Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1):89–97, 2004.

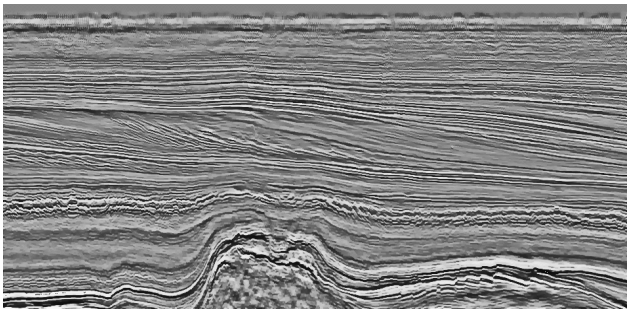
[5] Tom Goldstein and Stanley Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.



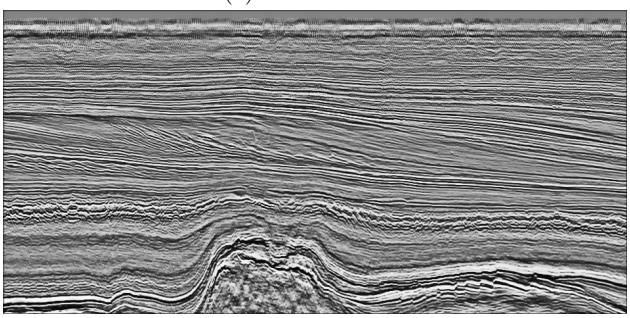
(a) Equalized no denoising



(b) Chambolle's



(c) Primal-Dual



(d) Split-Bregman

Figure 5: Results of each algorithm on a slice of a seismic volume from the OpendTect project [1]. The slice has been equalized prior to smoothing to enhance appearance of horizons.

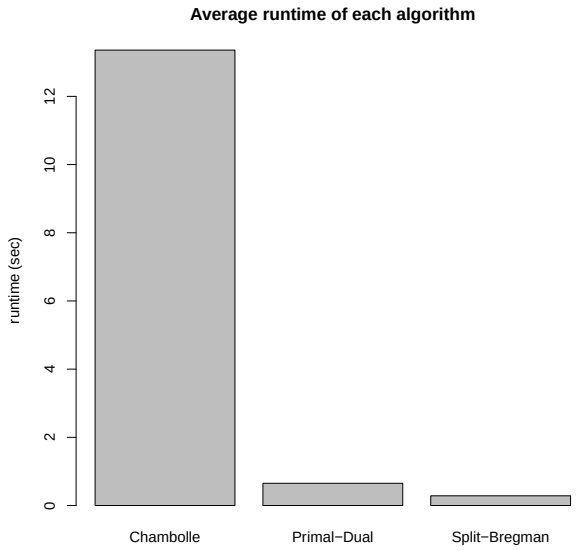


Figure 6: Average time spent accross all images by algorithm. Used visual stopping criteria.

[6] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639, 1990.

[7] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[8] Mingqiang Zhu and Tony Chan. An efficient primal-dual hybrid gradient algorithm for total variation image restoration. *UCLA CAM Report*, pages 08–34, 2008.