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 [4], where edge preservation was achieved by modeling the problem as an anisotropic diffusion problem. Later work in [5] 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 [1].

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 [5], 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 [2] 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 [6]. 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 [3]. 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 [3] 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

3 Results

3.1 Chambolle's Algorithm

3.2 Primal-Dual Algorithm

3.3 Split-Bregman Algorithm

4 Discussion

5 Conclusion

References

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

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

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

[4] 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.

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

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