Variational Methods as Convex Optimization Techniques for Image Denoising

Schuyler Tilney-Volk
Department of Electrical Engineering
Stanford University
stilneyv@stanford.edu

1. Background

Images and other 2D signals frequent our everyday experiences with technology, but interactions with image-based systems can easily by harmed by noise such as additive white Gaussian noise (AWGN). Many ideas in computer science and statistics, such as deep learning, maximum a posteriori (MAP), and convex optimization, have focused on image denoising techniques with varying amounts of success.

As presented in Mattingley et. al. [4], convex optimization, when applied to the field of signal processing, is normally applied in one of two ways—either to solve for the weights of a system, as when training a machine learning or deep learning model, or to the signal itself either in post-processing or real-time. Advancements in computational power have enabled the second method to grow rapidly in application, including for smaller images. Therefore, this project aims to explain various convex optimization image denoising techniques that fall into the second category and develop interactive application support to demonstrate these methods.[1]

2. Algorithmic Approach

As reviewed in Mattingley et. al. [4], the main application to image denoising in convex optimization is the total variation (TV) algorithm. Several works outline in detail the application of this algorithm, including Ocher et. al. [5], and others [1], [6]. For a noisy image Y, we can find the denoised image $\hat{X}$ using the TV algorithm:

$$\hat{X} = \arg\min_X ||Y - X||_2^2 + \lambda||\nabla X||$$

Where we can see the first term focuses on minimizing the difference between the original image and the image X, while the second term corresponds to a Tikhonov regularization term that applies to the gradients of X. This second term effectively acts as a first order smoothing operation. The algorithm is balanced by the regularization hyperparameter λ. It should be noted that these norms must actually be treated as vector norms, not matrix norms.

It is also worth knowing that as presented in Fan et. al. [2] that ℓ_2 may over-smooth in application. The original norm as in Equation 1 is isotropic but non-differentiable, and therefore we also use a formulation that is easier to minimize, the ℓ_1 method as in Equation 2, although it is anisotropic:

$$||\nabla X||_2 = \sum_{i,j} \sqrt{|X_{i+1,j} - X_{i,j}|^2 + |X_{i,j+1} - X_{i,j}|^2} \tag{1}$$

$$||\nabla X||_1 = \sum_{i,j} |X_{i+1,j} - X_{i,j}| + |X_{i,j+1} - X_{i,j}| \tag{2}$$

Both Equations 1 and 2 are available as options in the interactive tool. Note that the most common choice, the ℓ_1 regularization term, implies a guess that the prior on the gradients (spatial derivatives) of x is Laplacian—since the noise used in this project is AWGN, we also implement the Gaussian prior, which results in the ℓ_2 regularization.

Another addition is a novel type of second-order TV algorithm. This algorithm would not only aim to smooth the gradients of the image, but would also operate on the normal (spatial second derivative) vectors of the image:

$$\hat{X} = \arg\min_X ||Y - X||_2^2 + \lambda||\nabla X||_1 + \mu\ell_{\text{normal}}(X)$$

Where we define the normal vectors as:

$$n_i^X = [-\nabla_x X_i, -\nabla_y X_i, 1]$$
$$n_i^Y = [-\nabla_x Y_i, -\nabla_y Y_i, 1]$$

Where, X_i, Y_i are the ith pixel of X and Y, respectively.

And the normal loss as in Hu et. al. [3]:

$$\ell_{\text{normal}}(X) = \frac{1}{n}\sum_{i=1}^{nd} 1 - \frac{\langle n_i^X, n_i^Y \rangle}{\sqrt{\langle n_i^X, n_i^X \rangle}\sqrt{\langle n_i^Y, n_i^Y \rangle}}$$

[1] https://github.com/stilneyv/EE364B_Project

This formulation can be rewritten as:

$$\ell_{\text{normal}}(X) = 1 - \frac{1}{n}\sum_{i=1}^{nd} \frac{\langle n_i^X, n_i^Y \rangle}{||n_i^X||_2 ||n_i^Y||_2}$$

Using rules of composition, we can see that the term $\frac{1}{-||n_i^X||_2||n_i^Y||_2}$ is convex in n_X. However, if we consider the necessary and sufficient condition for proving convexity in cases of multiplication of two convex functions, where $f(x) = \frac{1}{-||n_i^X||_2||n_i^Y||_2}$ and $g(x) = (n_i^X)^T n_i^Y$:

$$\begin{aligned} h(x) &= f(x) \cdot g(x) \\ h'(x) &= f' \cdot g + f \cdot g' \\ h''(x) &= f'' \cdot g + f' \cdot g' + f \cdot g'' \succeq 0 \end{aligned}$$

To maintain brevity of the paper, we leave it as an exercise to the reader to show that this formulation yields a proof that $h(x)$, and therefore the resulting normal loss function, is non-convex. For next steps, we could opt to either form a convex relaxation or redefine the spatial second derivative. Because this project is a pedagogical project and not one that focuses on relaxations, we chose a simple definition. This definition simply reapplies the definition of the spatial gradient to the original gradient in Equation 2:

$$||\nabla^2 X||_1 = \sum_{i,j} |\nabla X_{i+1,j} - \nabla X_{i,j}| + |\nabla X_{i,j+1} - \nabla X_{i,j}|$$

Which as we can see merely affects the width of the "filter" we use to regularize, as we calculate the gradients in a discrete space by shifting a two copies of the image along the x and y axis, and then subtracting from the original. While this new formation is anisotropic, it is also convex and therefore easier and quicker to minimize.

The only related work for second order image processing algorithms was Hu et. al. [3], in which they used a second-order smoothing term in one of their loss functions for their study on DNN objectives for single image depth estimation. They assert that natural scenes and images can roughly be modeled by a limited number of smooth surfaces and step edges in between them. For instance, depth will often be discontinuous at the boundary of an object, and errors around such strong edges are well penalized by an ℓ_{grad} term as outlined above. However, since depth differences at such occluding boundaries of objects can sometimes be very large, we often must choose a modest weight $\lambda > 0$ on ℓ_{grad}, as in the case of the jaguar test image (note that ℓ_{grad} is not upper bounded). Then, the term $\lambda \ell_{\text{grad}}$ cannot penalize small structural errors such as those of high-frequency undulation of a surface, which hence necessitates the use of a second order term. Hence, a second order term is extremely useful in high-contrast images.

3. Algorithm Implementation

3.1. Interactive Application

In order to most succinctly demonstrate the effects of the algorithms implemented, we developed an interactive application in Matlab. This allows us to show the stages of the noise and denoise process. The GUI can be seen in Figure 1.

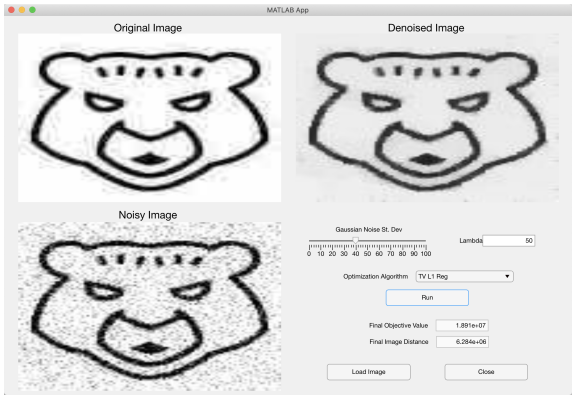

Figure 1: Application Setup

The top right figure of the GUI is the original image uploaded, which the user selects via pressing the "Load Image" button at the bottom right of the screen. Once the image is accepted, it is cropped if it is too large (see Section 3.2), and then AWGN is applied to the image using the standard deviation (ϵ) supplied by the slider. The standard deviation is measured in grayscale shades from 0-255, and defaults to 40 pixels. Adjusting the value on the slider also changes the noisy image once the slider settles on a value. The noisy image is shown on the bottom left.

The second stage of the process is the denoising. The user can select one of two algorithms (currently, but to be expanded. See Section 5) as well as a regularization constant λ. The default value is currently 50, as it has demonstrated promising results in initial trials. The user then must press "Run" to run the process, which initializes an algorithm which then runs a CVX routine. The result, when finished, is then shown in the upper right quadrant of the GUI.

3.2. Algorithms Implemented

To start the project, we implemented the two types of total variation as discussed in Section 2. These algorithms use an ℓ_2 distance metric for the noisy and denoised image and utilize a regularization term applied to the gradients. The gradients in this case are merely spatial derivatives, which are calculated by taking the difference between a pixel of focus $X_{i,j}$ and its neighbors to the right and below, $X_{i+1,j}$ and $X_{i,j+1}$.

However, when working with 2D signals, it should be

noted that this also means the image gradient is twice the size of the image—and while improvements to the CVX implementation of the algorithms did help runtime, all of the images tested and shown so far were 128x128 or smaller, as the 128x128 took roughly 30-40 minutes to run. For a tool with the purpose of being a pedagogical example as opposed to industry standard, we have thus confined the image selection to be 100x100 or below. In addition, color images would require gradients across all 3 RGB channels, making computation even more infeasible. As a result, the app auto-shifts images to grayscale, and larger images will automatically have the center 100x100 section selected and used instead of the original input, though for the second phase of the project we plan to examine alternatives so we can work with larger images. To combat this runtime, we attempted various ways of parallelism and multithreading in Matlab, which unfortunately CVX does not support.

This prompted us to examine further methods that do not require the use of CVX—in particular, we attempted to implement alternating direction method of multipliers (ADMM) for the ℓ_1 method of regularization on the gradients. For this method, we have an image to optimize $X \in \mathbb{R}^{n\times d}$, slack variable $Z \in \mathbb{R}^{n\times d\times 2}$, and augmented Lagrangian variable $U \in \mathbb{R}^{n\times d\times 2}$. The noisy image is defined as Y. We have the following update rules, and leave the extended proofs to the reader to find in Yang et. al. [7]:

$$X^{k+1} = \frac{Y + \sum_{i=1}^{2} \rho Z_i^k + U_i^k}{1 + 2\rho}$$
$$Z_i^{k+1} = \arg\min_{Z_i} ||Z_i + \frac{1}{\rho} U_i^k - X^{k+1}||_F + \rho F_i(Z_i)$$
$$U_i^{k+1} = U_i^k + \rho(Z_i^{k+1} - X^{k+1})$$

where,

$$F_i(Z) = \sum_{/\{j_i\}} \sum_{j_i=1}^{I_i-1} |x_{j_1,\ldots,j_{i-1},j_i+1,j_{i+1},\ldots,j_m} - x_{j_1,\ldots,j_{i-1},j_i,j_{i+1},\ldots,j_m}|$$

The notation $/\{j_i\}$ indicates the set of j's with the j_ith element removed. For example, $F_1(Z)$ for a 2D tensor (eg. an image) is $F_1(Z) = \sum_{i=1}^{n-1} \sum_{j=1}^{d} |x_{i,j} - x_{i+1,j}|$.

It is clear that one of the update rules is not shown in closed form. As a results, when implementing ADMM, we used CVX to find the value of the argmin. This significantly slowed the convergence process, as optimizing the parameter Z requires optimization of $n-1 \times d-1 \approx 5,000$ parameters twice per iteration, using CVX SDPT3 package. This took about 7 seconds $\times$ 2 dimensions = 14 seconds per iteration. Forming a faster update method for Z is a strong future step for this project.

In addition, we implemented the second-order spatial derivative method, using CVX to optimize the objective. Upon re-evaluation of the algorithm, the original proposed normal loss was not convex, as highlighted above, and aimed to use a simplified redefinition of a second-order loss that took the spatial derivative of the gradients again. As mentioned above, the advantage of this loss is it encourages smoothness across higher frequency undulations that the gradient-based loss term may not be able to capture without a large regularization constant.

However, it quickly became clear that this method, especially when just implemented with CVX's out-of-the-box algorithms, faced tremendous downsides not only in terms of runtime, but also in terms of memory. When using new definition of the spatial second derivative, we can see that the second order terms need to run twice on each of the two gradient matrices—which generates a total of 4 more $n-2 \times d-2$ matrices. This effectively more than doubles the number of variable based terms. While we did implement the algorithm with CVX in Matlab, the algorithm couldn't complete denoising for a reasonably-sized image in a reasonable amount of time. This is likely why Hu et. al. [3] used Pytorch and a GPU setup.

4. Results

Given a grayscale image of a bunch of banana leaves, we get the following results using $\epsilon = 40$ pixels and $\lambda = 50$, as seen in Figures 2 and 3. Quantitative results are shown in the tables below.

It should be noted that the leaf image is 70x70 pixels and the jaguar image is 100x100. In addition, the images were chosen to be nearly opposite types—the jaguar image is far more binary than the multi-toned and somewhat blurred banana leaf image. Finally, the distance to the original image is measured using SNR in decibels (as a reminder, the higher the SNR, the better, contrary to most metrics used in optimization).

Table 1: Leaf Results for $\epsilon = 40$p, $\lambda = 50$			
Statistic	ℓ_2 Reg	ℓ_1 Reg	ADMM ℓ_1
Optval	$1.075 \cdot 10^7$	$1.169 \cdot 10^7$	$2.464 \cdot 10^6$
SNR (dB)	36.66	33.19	33.33
Time (min:sec)	2:31	1:56	13:12

Table 2: Jaguar Results for $\epsilon = 40$p, $\lambda = 50$			
Statistic	ℓ_2 Reg	ℓ_1 Reg	ADMM ℓ_1
Optval	$1.693 \cdot 10^7$	$1.902 \cdot 10^7$	N/A
SNR (dB)	88.43	78.36	N/A
Time (min:sec)	4:26	10:43	Really High

We also experimented with $\epsilon = 70$ pixels and $\lambda = 50$, though we do not show the images for brevity.

Table 3: Leaf Results for $\epsilon = 70$p, $\lambda = 50$			
Statistic	ℓ_2 Reg	ℓ_1 Reg	ADMM ℓ_1
Optval	$1.696 \cdot 10^7$	$4.217 \cdot 10^6$	N/A
SNR (dB)	17.69	19.46	19.97
Time (min:sec)	2:28	2:07	10:54

Table 4: Jaguar Results for $\epsilon = 70$p, $\lambda = 50$			
Statistic	ℓ_2 Reg	ℓ_1 Reg	ADMM ℓ_1
Optval	$2.483 \cdot 10^7$	$2.737 \cdot 10^7$	N/A
SNR (dB)	37.57	37.29	N/A
Time (min:sec)	7:59	10:24	Really High

Based on the quantitative results for $\epsilon = 40$ pixels and $\lambda = 50$, we can see that performance was better across both images using the ℓ_1 gradient regularization, as well as being less costly on time.

But looking at the image results themselves, we can see two interesting things. First, for the jaguar, the ℓ_1 regularization seems to have more pure white pixels in the final image than not only the ℓ_2, but than the original image—it's effectively smoothed the few gray pixels that even existed in the original, for example on the nose and cheeks of the jaguar. In addition, when looking at the leaf results, we can see that the image appears to be oversmoothed. This can be explained by the fact that the ℓ_1 norm encourages sparsity, and sparsity in the gradient term forces nearby pixels to share the same tones. As a result, the image looks a bit like a conglomeration of Voronoi cells from farther away, each of which contains a region of pixels with (roughly) the exact same color tone. For example, on the leaf in the middle, we see much less variation in the white-toned part of the leaf compared to the ℓ_2 version. We've also lost a lot of the leaf's definition there.

That said, the ℓ_1 also creates more defined lines and edges where sparsity isn't as enforceable, particularly in contrast to the ℓ_2. For example, on the bottom right edge middle leaf in the banana leaf image, we can see the white part on the edge in the ℓ_1 image aren't as plagued with bits of dark noise as the ℓ_2 image. The intuition behind this can also be explained by the behavior of the ℓ_2 norm: in contrast to the ℓ_1, the Euclidean norm attempts to accept small errors but penalize large differences due to the quadratic relation.

Another interesting result lies in the ADMM version—obviously, due to the convexity of the problem, we expect to see the exact same result as in the ℓ_1 regularized version. And we roughly do. Due to the time issues of running CVX twice for the Z update step, we set the convergence criterion to an $\epsilon = 10^{-4}$. As such, we can see the images are quite close (as are the SNR values), but we prematurely terminate the operation such that we can still get results in a more reasonable timeframe. Due to these same runtime constraints, we decided not to run ADMM on the jaguar due to the jaguar's size (see Table 1 for ADMM runtime results on the 70x70 leaf image).

5. Future Work

Because this is a pedagogical project primarily focused on optimization in application, future work for this project will primarily include aspects that make this tool a better educational reference. Ideally, it would allow students to experiment with different methods on different types of images (high/low noise, high/low contrast) and would give students a framework to implement their own experimental methods, as all they would need to do is write their procedure and add a line in the .mlapp file.

As such, other application-based improvements speeding up the ADMM method by deriving a closed-form update rule or even just a faster process than calling CVX twice per iteration. In addition, we would like to experiment with a wider range of ϵ and λ values, implementing more algorithms outside of variational denoising—TV is a variational denoising method that strictly operates on a local level. Additional non-local algorithms could be added, as local algorithms are time-efficient but performance-constrained for higher amounts of AWGN. Two more algorithms that could be implemented are non-local regularization, as again mentioned by Fan et. al. [2], as well as B3MP which builds on the non-local means (NLM) foundation of of NLR. Or we could also implement different priors in the variational denoising method (ie. sparse priors) as above.

References

[1] A. Beck and M. Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE Transactions on Image Processing*, 18(11):2419–2434, 2009.

[2] L. Fan, F. Zhang, H. Fan, and C. Zhang. Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine, and Art*, 2, 12 2019.

[3] J. Hu, M. Ozay, Y. Zhang, and T. Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. 2019.

[4] J. Mattingley and S. Boyd. Real-time convex optimization in signal processing. *IEEE Signal Processing Magazine*, 27(3):50–61, 2010.

[5] S. Osher, A. E, and L. VESE. Image decomposition and restoration using total variation minimization and the h-1 norm. 08 2003.

[6] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin. Deep learning on image denoising: An overview, 2020.

[7] S. Yang, J. Wang, W. Fan, X. Zhang, P. Wonka, and J. Ye. An efficient admm algorithm for multidimensional anisotropic total variation regularization problems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge*

(a) Original Image

(b) Noisy Image

(c) L1 Regularized

(d) ADMM L1 Regularized

(e) L2 Regularized

Figure 2: L1, L2 TV Denoising on Banana Leaf Image

Discovery and Data Mining, KDD '13, page 641–649, New York, NY, USA, 2013. Association for Computing Machinery.

(a) Original Image

(b) L1 Regularized

(c) Noisy Image

(d) L2 Regularization

Figure 3: L1, L2 TV Denoising on Jaguar Icon