An ADMM Approach for Solving a Graph Laplacian Regularization for the Image Deblurring Problem

Report for GSMMC 2024

Emmanuel Adebayo, Jiayue Chen. Clayton Coonrod, Tyler Fuller, Christopher Koh, Dimitri Lopez, Abiodun Sumonu, Trishala Thakur, and Xiaonan Xu
Faculty Mentor: Malena I. Español, Arizona State University

June 23, 2024

Abstract

Image deblurring is a classic example of an ill-posed inverse problem. It is in the nature of these problems to find a solution by means of minimizing a regularized least squares problem. Several design choices must be made regarding the formulation of the regularization term. In this report, we make two such choices. First, we use the ℓ_1 norm on the regularization term to promote edge preservation in our image. Second, we use the weighted graph Laplacian with kernel pre-processing to determine the regularization matrix. Due to the non-differentiability of the ℓ_1 norm, we use an iterative reweighting scheme to reduce the ℓ_1 problem into a series of ℓ_2 norm approximations. We also explore utilizing an ADMM solver for the ℓ_1 minimization problem.

1 Introduction

We are interested in solving the image deblurring problem. Suppose we have a blurry image $\mathbf{b}$ and want to recover the clean, original image $\mathbf{x}$. This problem can be modeled as

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1}$$

where $\mathbf{A}$ is a blurring operator that acts on $\mathbf{x}$ to produce $\mathbf{b}$. Our objective is to recover $\mathbf{x}$ given $\mathbf{b}$ and $\mathbf{A}$. However, in practice, our data is contaminated with noise. Thus, we have

$$\mathbf{A}\mathbf{x} \approx \mathbf{b}_{true} + \epsilon = \mathbf{b}, \tag{2}$$

where $\mathbf{b}_{true} = \mathbf{A}\mathbf{x}$ and ϵ represents additive Gaussian white noise.

The problem of deblurring an image is an ill-posed inverse problem. Thus, $\mathbf{A}$ is generally ill-conditioned, and simply inverting $\mathbf{A}$ to recover $\mathbf{x}$ results in a poor solution (see [3, 6, 7]). To get around this, we solve the related regularized least squares problem instead. One example of such a formulation is Tikhonov regularization, where we want to find

$$\mathbf{x} = \arg\min_{\mathbf{x}} \{||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2 + \lambda||\mathbf{L}\mathbf{x}||_2^2\}, \tag{3}$$

where λ is a positive constant and $\mathbf{L}$ is an operator designed to extract the desired features from the image. Tikhonov regularization lends itself well to optimization, often resulting in an analytic solution that vastly reduces computational intensity. However, the regularization term penalizing the square of the parameters may cause significant depletion or outright suppression of features present in the data. In image or signal processing tasks such as edge detection or feature extraction, the quadratic penalization of Tikhonov regularization may smooth out sharp transitions and details that constitute edges. These edges are crucial to represent boundaries and discontinuities within the data accurately. Specifically in image-denoising applications where preserving sharp edges is essential, the aggressive nature of Tikhonov regularization might

blur or remove these edges, resulting in perceptually blurred recovered images. For these reasons, researchers often turn to the more computationally intensive ℓ_1 regularization, also known as Lasso regularization, which penalizes the absolute values of the parameters rather than their squares. The ℓ_1 regularization approach encourages sparsity and promotes the preservation of important features such as edges while still achieving the desired regularization effect. It selectively targets and shrinks less critical parameters to zero and offers a more nuanced control over feature preservation, making it particularly suitable for our application. To significantly preserve features, we consider ℓ_1 regularization

$$\mathbf{x} = \arg\min_{\mathbf{x}} \{||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2 + \lambda||\mathbf{L}\mathbf{x}||_1\}. \tag{4}$$

The non-differentiability of the ℓ_1 norm adds additional difficulty to the minimization problem. As a result, we will be using the Alternating Direction Method of Multipliers (ADMM) [4] method to minimize (4). The matrix $\mathbf{L}$ modifies $\mathbf{x}$ to promote desired properties in our reconstruction. Popular choices include discretizations of the first and second derivative operator [6]. For this report, we explore using the graph Laplacian regularizer to define $\mathbf{L}$ at each iteration of ADMM.

The paper is organized as follows. In Section 2, we introduce different regularization matrices, and in particular, we discuss the graph Laplacian regularizer to adaptively determine the regularization matrix $\mathbf{L}$ of our iterative solver in 2.2. In Section 3, we discuss the ADMM method for solving the ℓ_1 regularization problem. We present numerical results in Section 4. Finally, we offer conclusions in Section 5.

2 Regularization Operators

2.1 First Derivative Operator

One possible regularization term is selecting $\mathbf{L}$ as the discrete two-dimensional first derivative [2]. The intuition of such an $\mathbf{L}$ is to reduce high-frequency oscillation by minimizing the difference between two adjacent pixels. Such regularization purely depends on spatial dependencies and promotes continuous/smooth change of pixel intensities. Structurally, such an $\mathbf{L}$ is a sparse matrix with 1 and -1 values between neighboring pixels. In this case, the blurring operator is defined as

$$\mathbf{L} = \begin{bmatrix} \mathbf{I} \otimes \mathbf{L}_{1\mathrm{D}} \\ \mathbf{L}_{1\mathrm{D}} \otimes \mathbf{I} \end{bmatrix},$$

where

$$\mathbf{L}_{1\mathrm{D}} = \begin{bmatrix} -1 & 1 & 0 & \cdots & & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & -1 & 1 & 0 \\ 0 & \cdots & & 0 & -1 & 1 \end{bmatrix}$$

and $\otimes$ is the Kronecker product, which for two matrices $\mathbf{A}$ and $\mathbf{B}$ is defined by

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \dots & a_{1n}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \dots & a_{2n}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \dots & a_{mn}\mathbf{B} \end{bmatrix}.$$

2.2 Graph Laplacian Regularization

Within the context of an image reconstruction problem

$$\min \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 + \lambda\|\mathbf{L}\mathbf{x}\|_p^p,$$

we aim to contribute to current techniques of using the graph Laplacian to define $\mathbf{L}$ within the regularization term. While there are plenty of other choices of $\mathbf{L}$, such as the first derivative operator explored in the previous

section, the graph Laplacian gives us a high amount of flexibility with what our regularizer encourages. This comes from the expressiveness that is possible by encoding the interactions between non-local pairs of pixels as a weighted graph.

There is a rich history of using the graph Laplacian for image processing problems - including the image deblurring problem. One such work is that of Aleotti et al.[1], where they use the graph Laplacian within the ℓ_1 regularizer for the image deblurring problem. This particular regularizer encourages smoothness within regions of high similarity in the reconstructed image.

As a brief overview, this is done by taking our image and creating a graph representation of the pixels. The resulting graph should capture relevant dynamics between pixels, even those that are not adjacent to one another. This is the true strength of the graph Laplacian as it is able to promote arbitrary characteristics within the reconstructed image.

We start with a basic realization of the graph Laplacian, following the work of Aleotti et al.[1]. We will be creating a weighted graph $G = (V, E)$ where V is the set of vertices corresponding to pixels within our original image. E is the set of edges capturing the relationship between pairs of vertices or in this case pixels. Within a weighted graph, we associate a given weight to each of the edges $\omega : E \to \mathbb{R}_+$. Let $\mathbf{x}$ be a given image and the intensity of a pixel to be defined as $\mathbf{x}_{\vec{i}}$ where $\vec{i}$ and $\vec{j}$ are indices of the pixel, meaning that $\vec{i}, \vec{j} \in \mathbb{N}^2$.

Our goal is to create a weighted graph that has edges between pairs of pixels that are visually similar and spatially near one another. To do this, for each pixel, we only consider other pixels within a radius R. Formally, we can write out our radius constraint as pairs of pixels that satisfy $0 < \|\vec{i} - \vec{j}\|_\infty \leq R$. If two pixels are near enough to one another, we describe their similarity as a function of the intensity of the two pixels:

$$\omega_{\vec{i},\vec{j}} = \exp(-(\mathbf{x}_{\vec{i}} - \mathbf{x}_{\vec{j}})^2/\sigma). \tag{5}$$

Our graph representation should ultimately capture how information diffuses through the graph. This is the main reason why the Gaussian is used to determine the relatedness of pixel intensities.

By combining our radius constraint with the similarity with pairs of pixels, we get the weighted graph described by:

$$\mathbf{\Omega}_{\vec{i},\vec{j}} = \begin{cases} \exp(-(\mathbf{x}_{\vec{i}} - \mathbf{x}_{\vec{j}})^2/\sigma), & \text{if } 0 < \|\vec{i} - \vec{j}\|_\infty \leq R, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

We note that $\mathbf{\Omega}$ comes from Aleotti et al.[1]. With this graph in place, we can consider the graph Laplacian. While there are many forms of the graph Laplacian we have chosen to use:

$$\mathbf{L}_\omega = \frac{\mathbf{D} - \mathbf{\Omega}}{\|\mathbf{\Omega}\|_2}, \tag{7}$$

where $\mathbf{D}$ is the diagonal degree matrix defined by $\mathbf{D}_{j,j} = \sum_{i=1}^{n} \mathbf{\Omega}_{i,j}$. This graph Laplacian promotes regions of high similarity in the reconstructed image by encoding the similarity between potentially non-local pixels. While comparing pairs of pixels is interesting, encouraging similar non-local features in the image would be even better. With this in mind, we aim to apply a convolution upon $\mathbf{x}$, allowing us to create a graph that encodes similarities between similar areas around pixels.

2.3 Convolution

We utilize convolution techniques for deblurring, enhancing both sharpening and identity effects. We applied convolution to our initial approximation $\hat{\mathbf{x}} = \mathbf{A}^\top \mathbf{b}$ with a predefined kernel, and we can get two resulting Ω matrices through graph Laplacian regularization: $\mathbf{\Omega}_c$ obtained through the convolved estimation and $\mathbf{\Omega}_p$ without convolution. By normalizing both matrices, we combine them into the final $\mathbf{\Omega}$ matrix using specific weights determined through optimization, aiming to minimize the error between the deblurred and true images. The resulting deblurred image with the addition of convolution is shown in Section 4.3.

3 ADMM Solver

The Alternate Direction Method of Multipliers (ADMM) leverages the multipliers in the dual problem to facilitate the convergence of the original non-smooth optimization problem. Its advantage lies in the

decoupled primal and dual iterates, which can be decentralized and parallelized to speed up the algorithm. Moreover, it is a general umbrella type of algorithm that can be applied to a class of nonsmooth regularizers, and the iterations may only involve linear operations [4, 5]. In this framework, we seek $\mathbf{x}$ to minimize the regularized least squares problem as follows.

$$\frac{1}{2}||\mathbf{A}\mathbf{x}-\mathbf{b}||_2^2+\lambda||\mathbf{L}\mathbf{x}||_1, \tag{8}$$

where λ is a positive parameter, and $\mathbf{L}$ is defined as in the previous section. Let $\mathbf{L}\mathbf{x}=\mathbf{z}$. Then we have the constrained minimization problem

$$\min_{\mathbf{x},\mathbf{z}} \quad \frac{1}{2}||\mathbf{A}\mathbf{x}-\mathbf{b}||_2^2+\lambda||\mathbf{z}||_1 \tag{9}$$

$$\text{s.t.} \quad \mathbf{L}\mathbf{x}=\mathbf{z}. \tag{10}$$

We solve (10) using the Alternating Direction Method of Multipliers (ADMM). Consider the augmented Lagrangian with an additional quadratic term regulated by the parameter $\rho\in\mathbb{R}>0$

$$\mathcal{L}_\rho(\mathbf{x},\mathbf{y},\mathbf{z})=\frac{1}{2}||\mathbf{A}\mathbf{x}-\mathbf{b}||_2^2+\lambda||\mathbf{z}||_1+\langle\mathbf{y},\mathbf{L}\mathbf{x}-\mathbf{z}\rangle+\frac{\rho}{2}||\mathbf{L}\mathbf{x}-\mathbf{z}||_2^2. \tag{11}$$

Let $\mathbf{u}=\mathbf{y}/\rho$. Then, we rewrite (11) as

$$\mathcal{L}_\rho(\mathbf{x},\mathbf{u},\mathbf{z})=\frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\frac{\rho}{2}\left(\|\mathbf{L}\mathbf{x}-\mathbf{z}+\mathbf{u}\|_2^2-\|\mathbf{u}\|_2^2\right) \tag{12}$$

(see Appendix A for details). Define the corresponding Lagrangian dual function as

$$g(\mathbf{u})=\min_{\mathbf{x},\mathbf{z}}\mathcal{L}_\rho(\mathbf{x},\mathbf{z},\mathbf{u}).$$

Therefore, the dual problem to our original optimization problem becomes

$$\max_{\mathbf{u}} \quad g(\mathbf{u}).$$

Since $\mathcal{L}_\rho$ is convex with respect to $\mathbf{x}$ and $\mathbf{z}$, then by Fermat's rule, a necessary condition of optimality is $0\in\partial_{(\mathbf{x},\mathbf{z})}\mathcal{L}_\rho$, that is,

$$0\in\partial_{\mathbf{x},\mathbf{z}}\left\{\frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\frac{\rho}{2}\left(\|\mathbf{L}\mathbf{x}-\mathbf{z}+\mathbf{u}\|_2^2-\|\mathbf{u}\|_2^2\right)\right\},$$

where $\partial_{(\mathbf{x},\mathbf{z})}$ denotes the subdifferential with respect to $(\mathbf{x},\mathbf{z})$. Since the functions are proper, convex, and lower semicontinuous, the subgradient can be computed separately. That is,

$$\partial_{\mathbf{x}}\{\mathcal{L}_\rho\}=\mathbf{A}^\top\mathbf{A}\mathbf{x}-\mathbf{A}^\top\mathbf{b}+\rho\mathbf{L}^\top\mathbf{L}\mathbf{x}+\rho\mathbf{L}^\top\mathbf{u}-\rho\mathbf{L}\mathbf{z}$$

$$\partial_{\mathbf{z}}\{\mathcal{L}_\rho\}=\rho(\mathbf{z}-(\mathbf{L}\mathbf{x}+\mathbf{u}))+\lambda\partial\|\mathbf{z}\|_1=\rho(\mathbf{z}-(\mathbf{L}\mathbf{x}+\mathbf{u}))+\lambda\begin{cases}\mathbf{z}/|\mathbf{z}|, & \text{if } \mathbf{z}\neq\mathbf{0},\\ \mathcal{B}(0;1), & \text{if } \mathbf{z}=\mathbf{0},\end{cases}$$

where $\mathcal{B}(0;1)$ is the unit ball centered at 0. When both subgradients contain zero, we arrive at the ADMM updates, which take the form

$$\begin{aligned}\mathbf{x}^{(k+1)}&=\arg\min_{\mathbf{x}}\left|\left|\begin{pmatrix}\mathbf{A}\\ \sqrt{\rho}\mathbf{L}\end{pmatrix}\mathbf{x}-\begin{pmatrix}\mathbf{b}\\ \sqrt{p}\left(\mathbf{z}^{(k)}-\mathbf{u}^{(k)}\right)\end{pmatrix}\right|\right|_2^2\\ \mathbf{z}^{(k+1)}&=S_{\frac{\lambda}{\rho}}(\mathbf{u}^{(k)}+\mathbf{L}\mathbf{x}^{(k+1)})\\ \mathbf{u}^{(k+1)}&=\mathbf{u}^{(k)}+(\mathbf{L}\mathbf{x}^{(k+1)}-\mathbf{z}^{(k+1)}).\end{aligned} \tag{13}$$

See Appendix B for the derivation of $\mathbf{x}^{(k+1)}$ and Appendix C for the derivations of $\mathbf{z}^{(k+1)}$ and $\mathbf{u}^{(k+1)}$. Note that $\mathbf{x}$ updates only involve a typical least squares optimization procedure, and the updates on $\mathbf{z}$ and $\mathbf{u}$ are pointwise operations on vectors. The least squares problem can be solved using standard numerical linear algebra, but it is often the case that the matrices $\mathbf{A}$ and $\mathbf{L}$ have special structures due to the nature of image deblurring, which can be exploited. None of these techniques were utilized in this project, but they can be part of our future work.

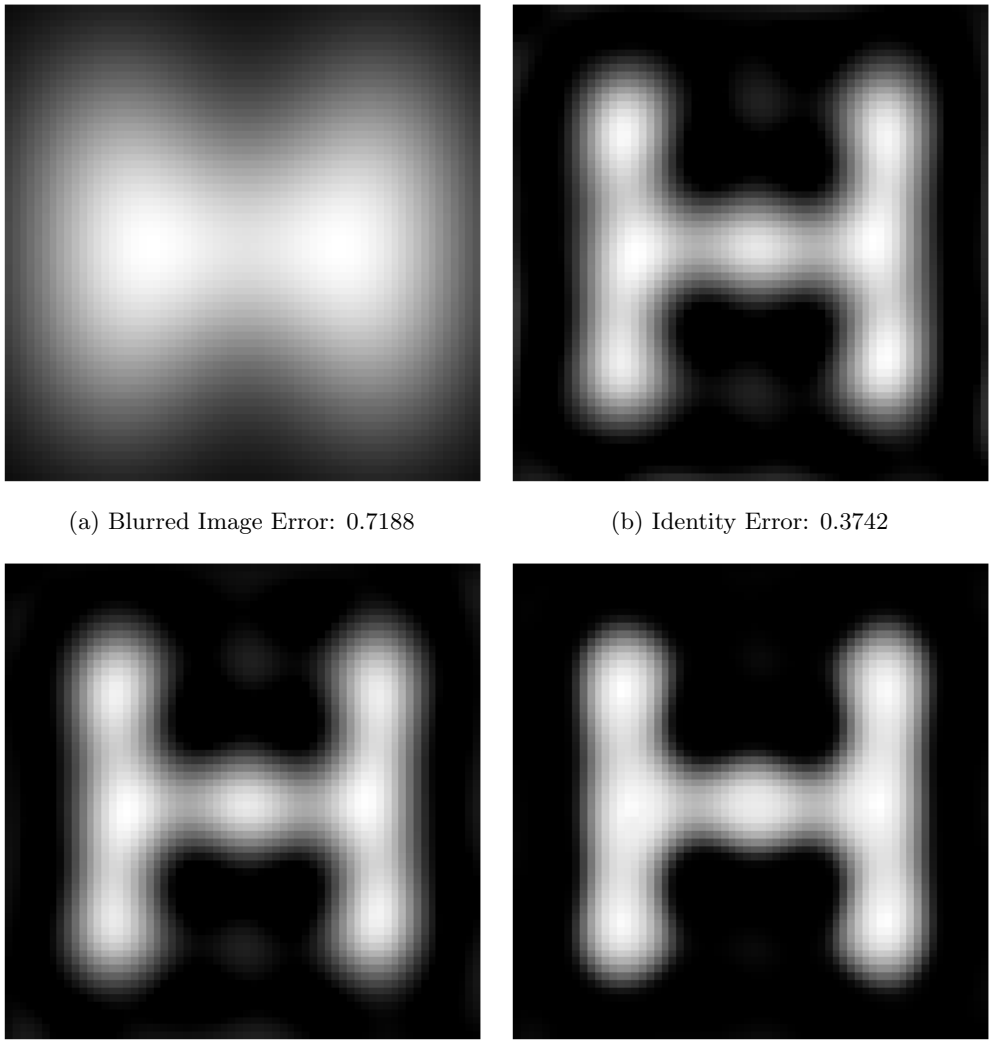
4 Numerical Results

In our example, we assume Gaussian blur with zero boundary conditions and contaminate the blurred images with 1% additive noise. Example code notebook with noised and denoised result images can be found at `https://github.com/Cfckoh/GSMMC-Deblurring`. To measure accuracy, we use relative error, define as

$$\frac{\|\mathbf{x} - \mathbf{x}_{\text{true}}\|_2}{\|\mathbf{x}_{\text{true}}\|_2}.$$

4.1 ℓ_2 norm results

Results for a binary image containing only purely white or black pixels with ℓ_2 regularization can be seen in Figure 1. The graph Laplacian regularizer performs best in the case of ℓ_2 regularization. Another example is using the Einstein image to perform the same task, and the results can be seen in Figure 2. The identity regularizer performs the best in the case of ℓ_2 regularization.



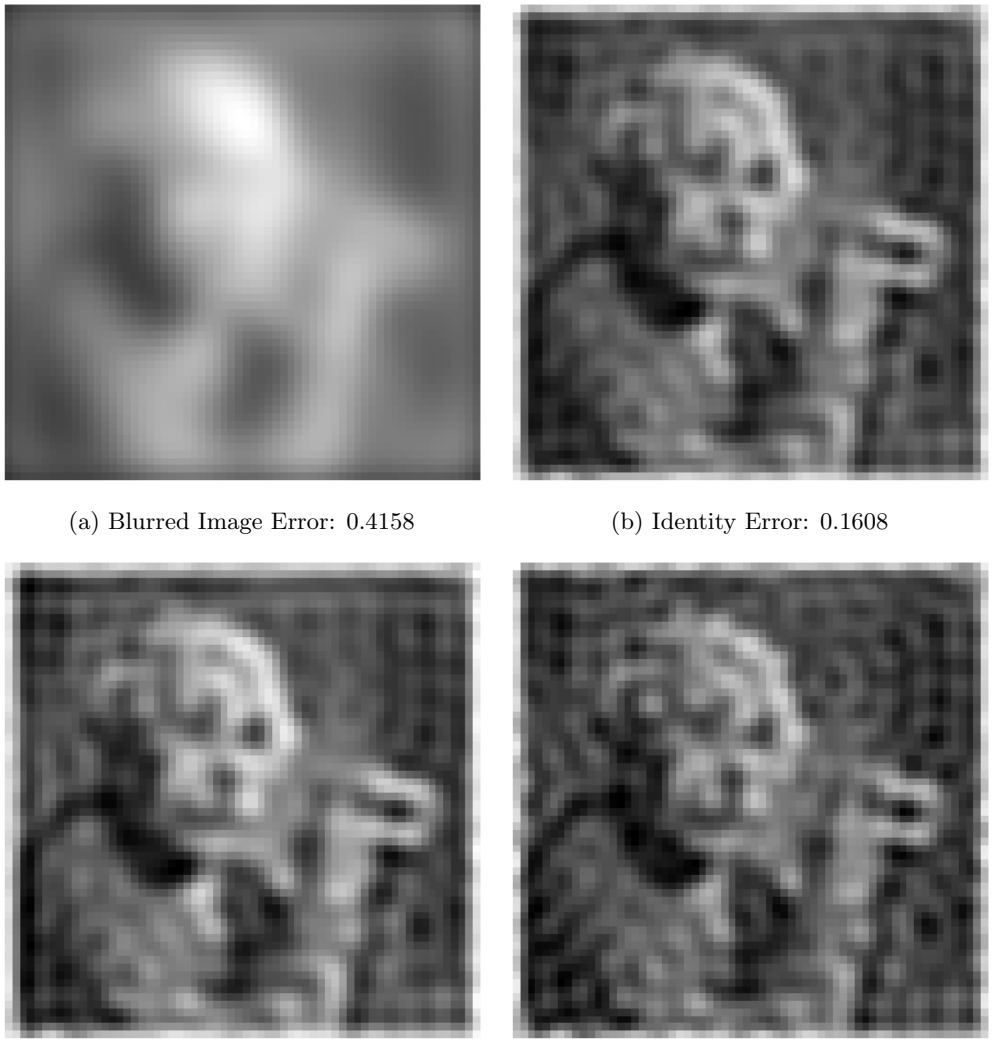
(a) Blurred Image Error: 0.7188

(b) Identity Error: 0.3742

(c) First Derivative Error: 0.3671

(d) Graph Laplacian Error: 0.3119

Figure 1: Reconstructions utilizing different regularization matrices under ℓ_2 regularization. In this case, the graph Laplacian performs best visually and by our quantitative error measurement. The graph Laplacian has higher contrast, enabled by the selective weighting of like neighbors (light with light and dark with dark).



(a) Blurred Image Error: 0.4158

(b) Identity Error: 0.1608

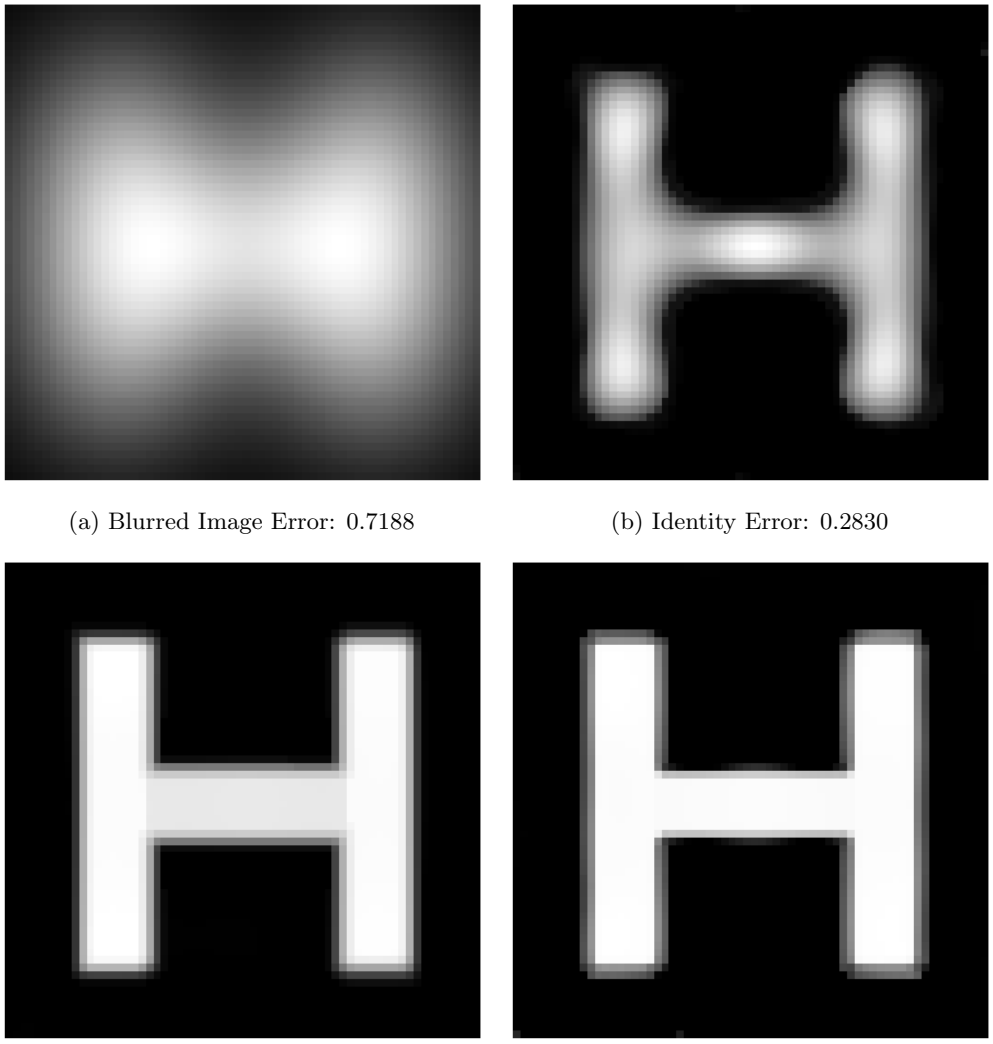
(c) First Derivative Error: 0.1804

(d) Graph Laplacian Error: 0.1857

Figure 2: Reconstructions utilizing different regularization matrices with ℓ_2 regularization. Based on our error estimation, the identity regularizer performs the best, even though the differences are subtle and indistinguishable to the naked eye. This blurred image of Einstein may preserve inherent structures effectively. Consequently, the first derivative and graph Laplacian regularizers, designed to emphasize edges and transitions, might inadvertently introduce noise and excessive smoothing, resulting in a higher error rate.

4.2 ℓ_1 norm results

Results for a binary image containing only purely white or black pixels with ℓ_1 regularization can be seen in Figure 3. The first derivative regularizer performs best in the case of ℓ_1 regularization. However, the graph Laplacian appears to have more visual contrast. Similar to the ℓ_2 norm result, we apply the same procedure to Einstein's image, as illustrated in Figure 4. The first derivative regularizer performs best for ℓ_1 regularization.



(a) Blurred Image Error: 0.7188

(b) Identity Error: 0.2830

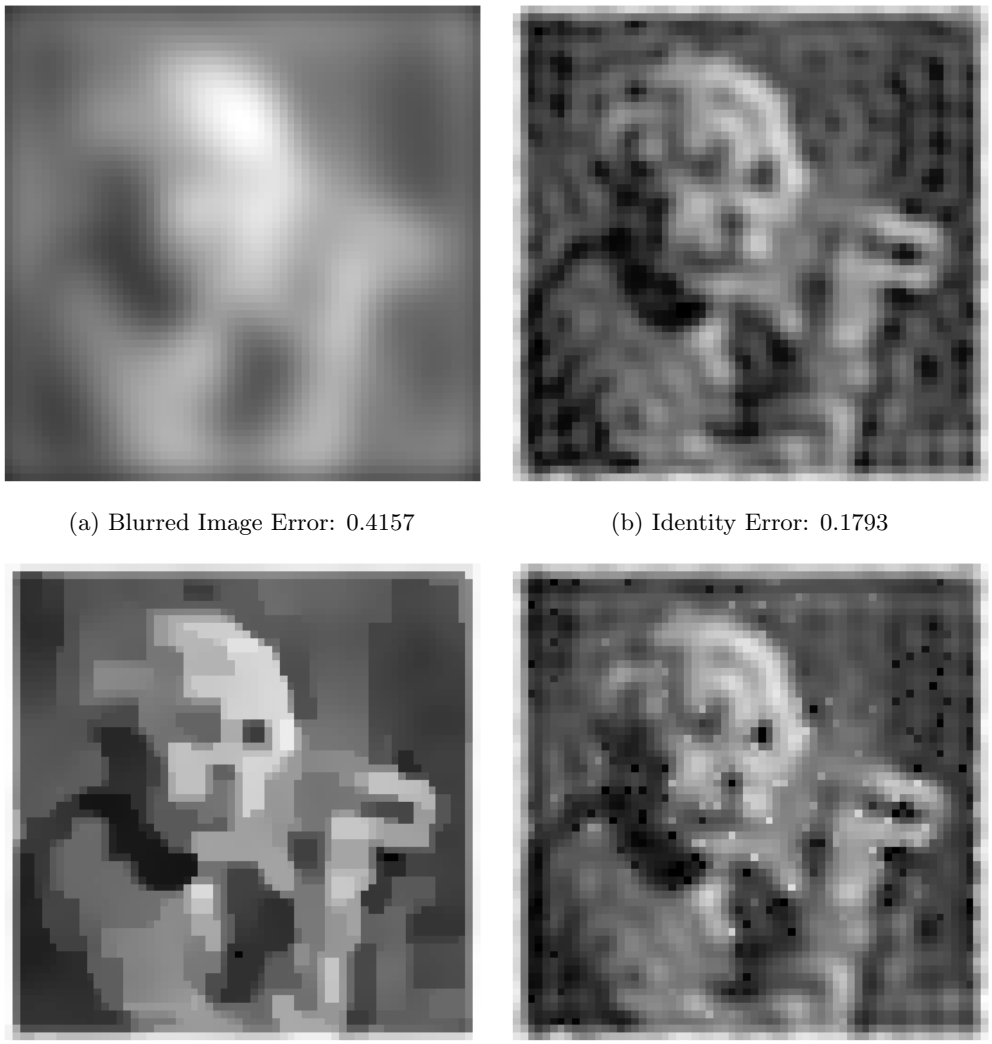
(c) First Derivative Error: 0.2008

(d) Graph Laplacian Error: 0.2411

Figure 3: Reconstructions utilizing different regularization matrices under ℓ_1 regularization. The ℓ_1 regularization term enables sharper edges in the graph reconstruction as the high contrast along the edges is penalized less heavily. Under ℓ_1 the first derivative performs best by our quantitatively measured error. Interestingly, the graph Laplacian appears to create a visually more defined and uniformly colored H, though the error is higher.

4.3 Preprocessing Convolutions

Figures 5 and 6 illustrate the deblurred image obtained by first applying the sharpening convolution kernel 1 and kernel 2, then reconstructing the images of 'H' and Einstein respectively using ℓ_2 graph Laplacian regularization as described in the previous sections. The sharpening convolution kernels used were:



(a) Blurred Image Error: 0.4157

(b) Identity Error: 0.1793

(c) First Derivative Error: 0.1626

(d) Graph Laplacian Error: 0.1965

Figure 4: Reconstructions utilizing different regularization matrices and ℓ_1 regularization. Based on our error metrics, the first derivative regularizer performs best in the ℓ_1 situation, while visually, the identity regularizer appears more effective. This discrepancy may be due to the lower noise levels in the first derivative regularizer, as evidenced by fewer dark spots and more consistent color strips. Different regularizers perform variably depending on the image, with the identity regularizer excelling in images where the inherent structure is well-preserved.

Kernel 1:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Kernel 2:

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Both kernels aim to enhance the edges in the blurred image by highlighting the differences in intensity between adjacent pixels. After applying these kernels, ℓ_2 graph Laplacian regularization was used to reconstruct the deblurred images. The results show that both kernels successfully reduced the error compared to the blurred image.

4.4 ADMM Results

Figure 7 compares the result of using a native implementation of ADMM to reconstruct a blurred image with two types of regularizers: (i) the graph Laplacian, and (ii) ℓ_{1D}. We did not use any preconditioning, nor did we take advantage of the structures of $\mathbf{A}$ and $\mathbf{L}$, and were still able to deblur the images to some extent. From these images, we can see that the ADMM construction with the graph Laplacian yields better results than the original deblurred image. The ℓ_2 construction is about to recover the underlying image; however, it remains slightly more blurred than the graph Laplacian. When running the ADMM method on the Einstein image, however, the graph Laplacian was not able to recover the image. With the L_{1D} matrix, we were able to recover Einstein, even though the image remains slightly blurred. The results for the Einstein image with ADMM and ℓ_2 are shown in Figure 8.

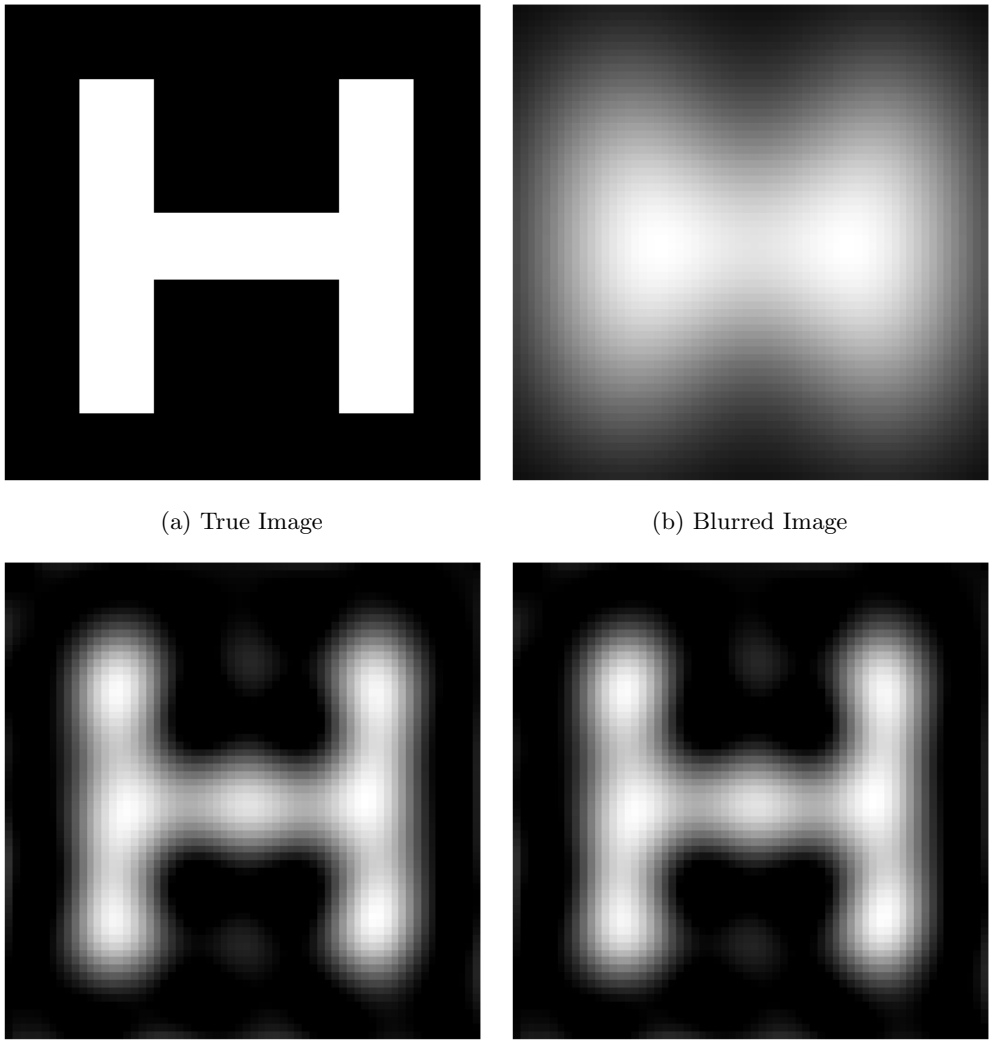
5 Conclusion

In this project, we worked on the ill-posed image deblurring problem. Within the problem, we explored a set of regularization terms, in particular, using the graph Laplacian and convolutions to encourage features within the reconstructed image. Furthermore, we formulated an optimization procedure by applying the ADMM method to ℓ_1 regularization.

Our original goal in using the graph Laplacian was to leverage the expressivity of graphs in order to promote characteristics in the reconstructed image. We followed the work of Aleotti et al. [1] and applied the graph Laplacian $\mathbf{L}_\omega$ with various metrics. When using ℓ_2, the graph Laplacian performed best on a binary image. It performed better than all other methods we used when using ℓ_2 as the metric for the regularization term. Intuitively, it makes sense that it worked best on this binary image as the graph Laplacian encourages sharp edges by promoting similarity between pixels that are close and similar in intensity to one another. When applied to a non-binary image, as seen with the one of Einstein, we found that several artifacts showed up in the reconstructed image. The image was sharpened a fair amount but not necessarily better compared to using the more naive identity matrix for $\mathbf{L}_\omega$.

To better the results by using the graph Laplacian, we used a set of convolutions as pre-processing on the images. By using convolutions, we were able to incorporate the context of the area surrounding a pixel into the graph representation. The end goal of which was to hopefully promote regions of high similarity throughout the image. Unfortunately, this pre-processing step using a convolution did not make a significant difference. We tried several sharpening kernels but found results that were on par or worse than just using the graph Laplacian. Perhaps different or larger kernels would produce better results. The idea of including the context of pixels when creating a graph Laplacian is one that might have merit, but we were unable to show that it does.

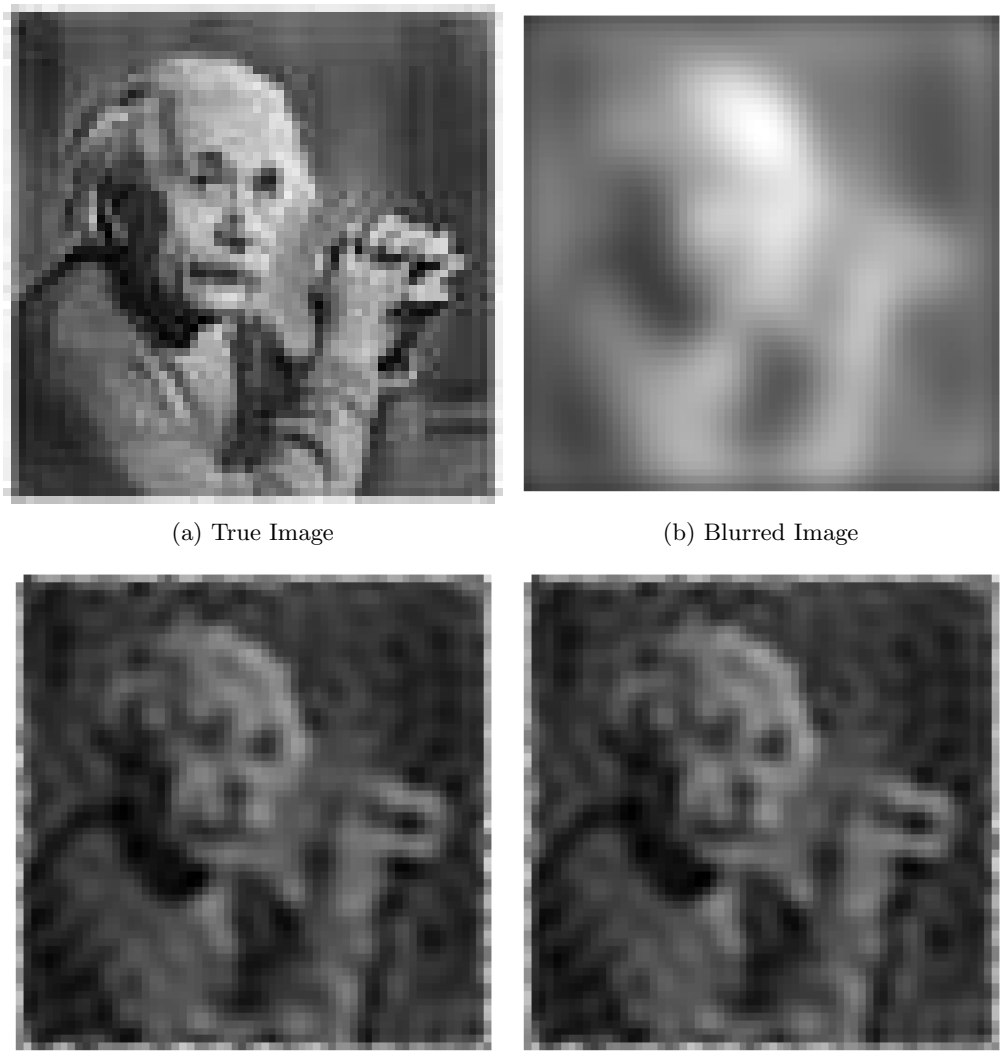
We found that the ADMM method was able to reconstruct the H image using both the first derivative and graph Laplacian regularizers, but neither reconstruction was able to surpass the reconstructions of the methods that did not utilize ADMM. Further, we were able to reconstruct the Einstein image using ADMM both with first derivative regularization and graph Laplacian regularization. In both cases, the reconstructions were much poorer than with the other methods considered. However, we were unable to recover the Einstein image with ADMM and graph Laplacian regularization.



(a) True Image (b) Blurred Image

(c) Sharpen Kernel 1 + ℓ_2 Error: 0.36816 (d) Sharpen Kernel 2 + ℓ_2 Error: 0.36789

Figure 5: In the deblurring process with convolutions, the blurred image was first convolved with two different sharpening kernels, followed by reconstruction using ℓ_2 graph Laplacian regularization. The true image (a) represents the ground truth. The blurred image (b) serves as the input for the deblurring process. The deblurred image obtained by applying sharpening convolution kernel 1 and then using ℓ_2 graph Laplacian regularization (c) has an error of 0.36816. Similarly, the deblurred image obtained by applying sharpening convolution kernel 2 followed by ℓ_2 graph Laplacian regularization (d) has an error of 0.36789. Both kernels aim to enhance the edges in the blurred image by highlighting the differences in intensity between adjacent pixels. The results show that both sharpening kernels deblurred the image (reduced the error), with kernel 2 performing slightly better in this context.



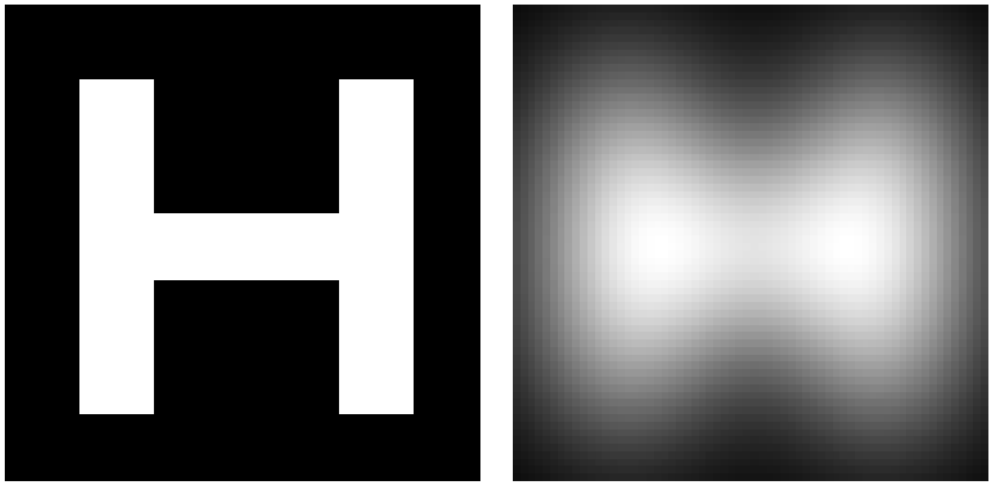
(a) True Image

(b) Blurred Image

(c) Sharpen Kernel 1 + ℓ_2 Error: 0.18180

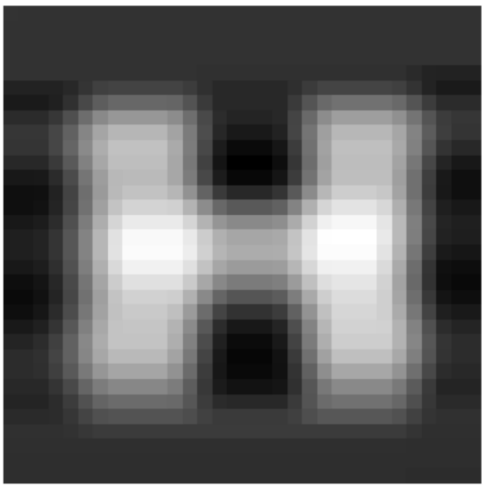
(d) Sharpen Kernel 2 + ℓ_2 Error: 0.17617

Figure 6: In the deblurring process involving convolutions, the blurred image was initially processed with two distinct sharpening kernels, followed by reconstruction using ℓ_2 graph Laplacian regularization. The true image (a) serves as the ground truth. The blurred image (b) is the starting point for the deblurring procedure. The deblurred image created by applying sharpening convolution kernel 1 and subsequently using ℓ_2 graph Laplacian regularization (c) has an error of 0.18180. Similarly, the deblurred image produced by applying sharpening convolution kernel 2 and then performing ℓ_2 graph Laplacian regularization (d) has an error of 0.17617. The results indicate that both sharpening kernels successfully deblurred the image, with kernel 2 achieving a slightly lower error. One interesting thing to note is the artifacts created at the image boundaries, which may be remedied by exploring different boundary conditions when convolving the blurred image.

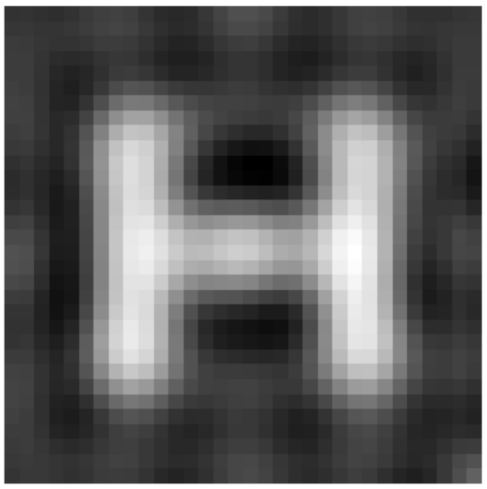


(a) Original H Image

(b) Blurred H Image



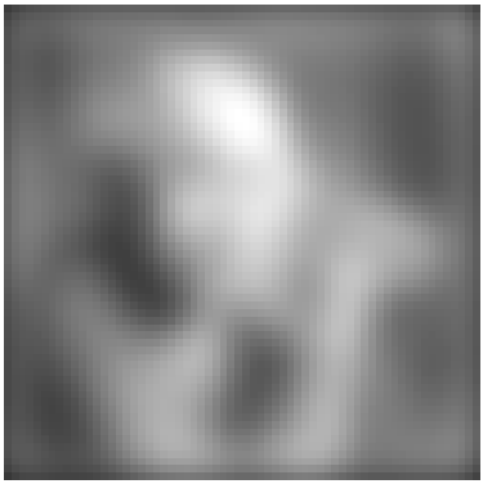
(c) ADMM Reconstruction using L_{1D}



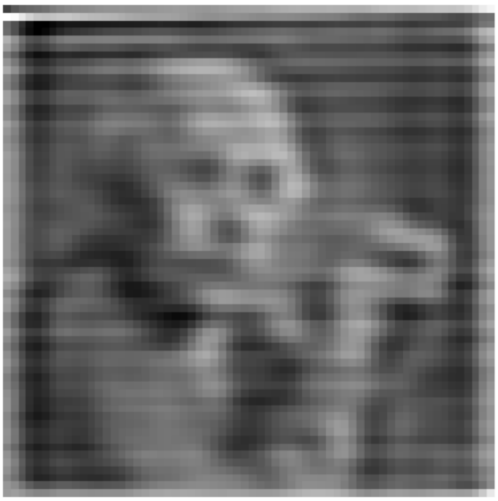
(d) ADMM Reconstruction with GL

Figure 7: ADMM Reconstructions of a binary image utilizing different regularization matrices in ℓ_1 regularization. The true image (a) represents the ground truth. The blurred image (b) serves as the input for the deblurring process. The image (c) is the deblurred image using the first derivative as the regularizer. The image (d) is the deblurred image using the graph Laplacian as the regularizer. Neither of the results is very satisfactory, but the rough shape of the original image was recovered. Using the graph Laplacian yielded a sharper image, which is consistent with previous numerical experimentation using other algorithms.

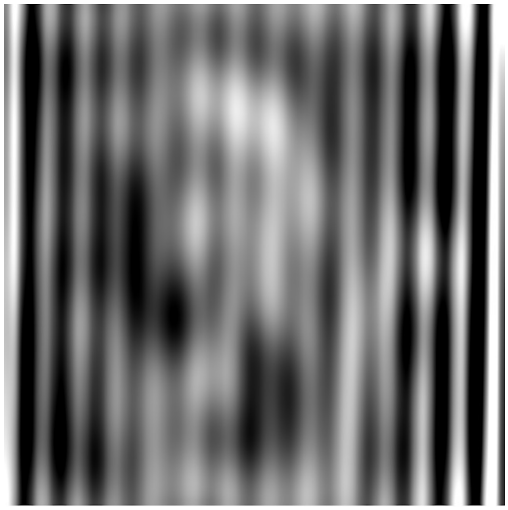
(a) True Image



(b) Blurred Image



(c) ADMM with $\mathbf{L}_{1D}$ Regularizer



(d) GL

Figure 8: ADMM Reconstructions of a grayscale image using $\mathbf{L}_{1D}$ regularizer and with graph Laplacian regularizer. ADMM Reconstructions of a binary image utilizing different regularization matrices in ℓ_1 regularization. The true image (a) represents the ground truth. The blurred image (b) serves as the input for the deblurring process. The image (c) is the deblurred image using the first derivative as the regularizer. The image (d) is the deblurred image using the graph Laplacian as the regularizer. Both images contain some artifacts not present in the original image, but an outline can still be roughly made out. We emphasize that we did not fine-tune the parameters or apply any numerical linear algebra techniques to improve convergence.

Future work can be encompassed in two categories. First, we can revisit the ADMM solver as a more rigorous study of its conversion rates, scalability, and parallelizability to understand its performance in different contexts. Second, we can refine our regularization approach. This would include better heuristics for determining regularization parameters and further exploration of edge weight functions.

References

[1] Stefano Aleotti, Alessandro Buccini, and Marco Donatelli. Fractional graph laplacian for image reconstruction. *Applied Numerical Mathematics*, 200:43–57, 2024.

[2] David Austin, Malena I Español, and Mirjeta Pasha. The image deblurring problem: Matrices, wavelets, and multilevel methods. *Notices of the American Mathematical Society*, 69(8), 2022.

[3] Jonathan M. Bardsley. *Computational Uncertainty Quantification for Inverse Problems.* Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 2018.

[4] Stephen Boyd. Alternating direction method of multipliers.

[5] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[6] Per Christian Hansen. *Discrete Inverse Problems: Insight and Algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 2010.

[7] James G. Nagy Per Christian Hansen and Dianne P. O'Leary. *Discrete Inverse Problems: Insight and Algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 2006.

A Lagrangian Reformulation

We rewrite the augmented Lagrangian by completing the square,

$$\begin{aligned}
\mathcal{L}_\rho(\mathbf{x},\mathbf{y},\mathbf{z}) &= \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\langle\mathbf{y},\mathbf{L}\mathbf{x}-\mathbf{z}\rangle+\frac{\rho}{2}\|\mathbf{L}\mathbf{x}-\mathbf{z}\|_2^2\\
&= \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\langle\mathbf{y},\mathbf{L}\mathbf{x}-\mathbf{z}\rangle+\frac{\rho}{2}\langle\left(\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho}\right)-\frac{\mathbf{y}}{\rho},\left(\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho}\right)-\frac{\mathbf{y}}{\rho}\rangle\\
&= \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\langle\mathbf{y},\mathbf{L}\mathbf{x}-\mathbf{z}\rangle+\frac{\rho}{2}\left(\langle\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho},\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho}\rangle-2\langle\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho},\frac{\mathbf{y}}{\rho}\rangle+\langle\frac{\mathbf{y}}{\rho},\frac{\mathbf{y}}{\rho}\rangle\right)\\
&= \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\langle\mathbf{y},\mathbf{L}\mathbf{x}-\mathbf{z}\rangle+\frac{\rho}{2}\left(\left\|\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho}\right\|_2^2-2\langle\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho},\frac{\mathbf{y}}{\rho}\rangle+\left\|\frac{\mathbf{y}}{\rho}\right\|_2^2\right)\\
&= \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\langle\mathbf{y},\mathbf{L}\mathbf{x}-\mathbf{z}\rangle+\frac{\rho}{2}\left(\left\|\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho}\right\|_2^2+\left\|\frac{\mathbf{y}}{\rho}\right\|_2^2\right)+\frac{\rho}{2}\left(-2\langle\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho},\frac{\mathbf{y}}{\rho}\rangle\right)\\
&= \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\langle\mathbf{y},\mathbf{L}\mathbf{x}-\mathbf{z}\rangle+\frac{\rho}{2}\left(\left\|\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho}\right\|_2^2+\left\|\frac{\mathbf{y}}{\rho}\right\|_2^2\right)+\left(-\langle\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho},\mathbf{y}\rangle\right)\\
&= \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\langle\mathbf{y},\mathbf{L}\mathbf{x}-\mathbf{z}\rangle+\frac{\rho}{2}\left(\left\|\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho}\right\|_2^2+\left\|\frac{\mathbf{y}}{\rho}\right\|_2^2\right)+\left(-\langle\mathbf{L}\mathbf{x}-\mathbf{z},\mathbf{y}\rangle-\langle\frac{\mathbf{y}}{\rho},\mathbf{y}\rangle\right)\\
&= \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\frac{\rho}{2}\left(\left\|\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho}\right\|_2^2+\left\|\frac{\mathbf{y}}{\rho}\right\|_2^2\right)-\frac{1}{\rho}\|\mathbf{y}\|_2^2\\
&= \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\frac{\rho}{2}\left\|\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho}\right\|_2^2+\frac{\rho}{2}\frac{1}{\rho^2}\|\mathbf{y}\|_2^2-\frac{1}{\rho}\|\mathbf{y}\|_2^2\\
&= \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\frac{\rho}{2}\left\|\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho}\right\|_2^2-\frac{1}{2\rho}\|\mathbf{y}\|_2^2\\
&= \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\frac{\rho}{2}\left\|\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho}\right\|_2^2-\frac{\rho}{2}\left\|\frac{\mathbf{y}}{\rho}\right\|_2^2\\
&= \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\frac{\rho}{2}\left(\left\|\mathbf{L}\mathbf{x}-\mathbf{z}+\frac{\mathbf{y}}{\rho}\right\|_2^2-\left\|\frac{\mathbf{y}}{\rho}\right\|_2^2\right).
\end{aligned}$$

Define the auxiliary vector $\mathbf{u}=\mathbf{y}/\rho$, then we have

$$\mathcal{L}_\rho(\mathbf{x},\mathbf{u},\mathbf{z})=\frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{z}\|_1+\frac{\rho}{2}\left(\|\mathbf{L}\mathbf{x}-\mathbf{z}+\mathbf{u}\|_2^2-\|\mathbf{u}\|_2^2\right). \tag{14}$$

B ADMM Updates: x

Consider $\mathcal{L}$ as defined in Appendix A. Define L_x as a function whose terms are all of the terms of $\mathcal{L}_\rho$ that depend on $\mathbf{x}$. Then, the ADMM iterates for $\mathbf{x}$ is given as

$$\mathbf{x}^{(k+1)}=\arg\min_{\mathbf{x}}\{L_x\}=\arg\min_{\mathbf{x}}\left\{\frac{1}{2}||\mathbf{A}\mathbf{x}-\mathbf{b}||_2^2+\frac{\rho}{2}||\mathbf{L}\mathbf{x}-\mathbf{z}^{(k)}+\mathbf{u}^{(k)}||_2^2\right\}. \tag{15}$$

To solve it, we first rewrite L_x to get

$$\begin{aligned}
L_x &= \frac{1}{2}(\mathbf{A}\mathbf{x}-\mathbf{b})^\top(\mathbf{A}\mathbf{x}-\mathbf{b})+\frac{\rho}{2}(\mathbf{x}^\top\mathbf{L}^\top-\mathbf{z}^{(k)\top}+\mathbf{u}^{(k)\top}(\mathbf{L}\mathbf{x}-\mathbf{z}^{(k)}+\mathbf{u}^{(k)})\\
&= \frac{1}{2}\mathbf{x}^\top\mathbf{A}^\top\mathbf{A}\mathbf{x}-\mathbf{x}^\top\mathbf{A}^\top\mathbf{b}+\mathbf{b}^\top\mathbf{b}+\frac{\rho}{2}(\mathbf{x}^\top\mathbf{L}^\top\mathbf{L}\mathbf{x}-2\mathbf{x}^\top\mathbf{L}^\top\mathbf{z}^{(k)}+\mathbf{z}^{(k)\top}\mathbf{z}^{(k)})+\mathbf{y}^\top\mathbf{L}\mathbf{x}-\mathbf{y}^{(k)\top}\mathbf{z}^{(k)}.
\end{aligned} \tag{16}$$

Taking the partial derivative with respect to $\mathbf{x}$,

$$\frac{\partial}{\partial\mathbf{x}}\{(L_x)\}=\mathbf{A}^\top\mathbf{A}\mathbf{x}-\mathbf{A}^\top\mathbf{b}+\rho\mathbf{L}^\top\mathbf{L}\mathbf{x}+\rho\mathbf{L}^\top\mathbf{u}^{(k)}-\rho\mathbf{L}\mathbf{z}^{(k)}. \tag{17}$$

If we set the partial derivative with respect to $\mathbf{x}$ equal to zero, we can rearrange (17) to get the normal equation

$$(\mathbf{A}^\top\mathbf{A} + \rho\mathbf{L}^\top\mathbf{L})\mathbf{x} = \mathbf{A}^\top\mathbf{b} + \rho\mathbf{L}^\top\left(\mathbf{z}^{(k)} - \mathbf{u}^{(k)}\right). \tag{18}$$

Solving the system in (18) is equivalent to solving the least squares problem

$$\min_{\mathbf{x}} \left|\left| \begin{pmatrix} \mathbf{A} \\ \sqrt{\rho}\mathbf{L} \end{pmatrix} \mathbf{x} - \begin{pmatrix} \mathbf{b} \\ \sqrt{p}\left(\mathbf{z}^{(k)} - \mathbf{u}^{(k)}\right) \end{pmatrix} \right|\right|_2^2, \tag{19}$$

which can be solved using a standard iterative scheme (for example, LSQR).

C ADMM Updates: z and u

The ADMM iterates for $\mathbf{x}$ and $\mathbf{z}$ are given by

$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}}\{L_x\} = \arg\min_{\mathbf{x}} \left\{ \frac{1}{2}||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2 + \frac{\rho}{2}||\mathbf{L}\mathbf{x} - \mathbf{z}^{(k)} + \mathbf{u}^{(k)}||_2^2 \right\} \tag{20}$$

$$\mathbf{z}^{(k+1)} = \arg\min_{\mathbf{z}}\{L_z\} = \arg\min_{\mathbf{z}} \left\{ \lambda||\mathbf{z}||_1 + \frac{\rho}{2}||\mathbf{L}\mathbf{x}^{(k+1)} - \mathbf{z} + \mathbf{u}^{(k)}||_2^2 \right\}, \tag{21}$$

and $\mathbf{u}^{k+1}$ is updated according to the dual feasibility condition, since for each update, $(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1})$ needs to satisfy the necessary conditions of optimality. First note that the Lagrangian in Equation (14) is *strongly* convex with respect to $\mathbf{x}$, and it is also smooth with respect to $\mathbf{x}$, so the minimizer of $\mathcal{L}_\rho$ is the point $\mathbf{x}^*$ such that the gradient of $\mathcal{L}_\rho(\mathbf{x}, \mathbf{u}^k, \mathbf{z}^k)$ with respect to $\mathbf{x}$ vanishes. That is,

$$\begin{aligned} 0 &= \nabla_{\mathbf{x}}\mathcal{L}_\rho(\mathbf{x}, \mathbf{u}, \mathbf{z}) \\ &= \nabla_{\mathbf{x}} \left(\frac{1}{2}||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2 + \frac{\rho}{2}||\mathbf{L}\mathbf{x} - \mathbf{z}^{(k)} + \mathbf{u}^{(k)}||_2^2 \right) \\ &= \mathbf{A}^\top(\mathbf{A}\mathbf{x} - \mathbf{b}) + \rho\mathbf{L}^\top(\mathbf{L}\mathbf{x} - \mathbf{z}^{(k)} + \mathbf{u}^{(k)}). \end{aligned}$$

Similarly, $\mathcal{L}_\rho$ is also convex with respect to $\mathbf{z}$, as a sum of two convex functions of a linear transformation of $\mathbf{z}$, albeit not smooth. However, notice that in Equation (21), the minimizer is exactly the proximal operator

$$\mathbf{prox}_{f,\rho} = \arg\min_{\mathbf{z}}(f(\mathbf{z}) + \frac{\rho/\lambda}{2}\|\mathbf{z} - \mathbf{v}\|_2^2)$$

with

$$f = \|\cdot\|_1, \text{ and } \mathbf{v} = \mathbf{L}\mathbf{x} + \mathbf{u}.$$

In the repository ProxRepo, we see that this corresponds to exactly the shrinkage function, i.e.,

$$\mathbf{z}^{(k+1)} = S_{\rho/\lambda}(\mathbf{v}).$$

Moreover, as $\mathbf{z}^{k+1}$ minimizes $\|\mathbf{z}\|_1 + \frac{\rho/\lambda}{2}\|\mathbf{z} - (\mathbf{L}\mathbf{x}^{(k+1)} + \mathbf{u}^{(k)})\|_2^2$, by Fermat's rule, we must have that

$$0 \in \partial\left(\|\mathbf{z}\|_1 + \frac{\rho/\lambda}{2}\|\mathbf{z} - (\mathbf{L}\mathbf{x}^{(k+1)} + \mathbf{u}^{(k)})\|_2^2 \right).$$

Since both $\|\cdot\|_1$ and $\frac{\rho/\lambda}{2}\|\mathbf{z} - \mathbf{v}\|_2^2$ are convex and (lower semi)continuous, the subdifferential of their sum is exactly the sum of their subdifferentials. That is, when $\mathbf{z}^{(k+1)}$ is a minimizer of (21), then

$$0 = \partial\|\mathbf{z}^{(k+1)}\|_1 + \partial\left(\frac{\rho/\lambda}{2}\|\mathbf{z}^{(k+1)} - (\mathbf{L}\mathbf{x}^{(k+1)} + \mathbf{u}^{(k)})\|_2^2 \right) = \partial\|\mathbf{z}^{(k+1)}\|_1 + \frac{\rho}{\lambda}(\mathbf{z}^{(k+1)} - (\mathbf{L}\mathbf{x}^{(k+1)} + \mathbf{u}^{(k)})). \tag{22}$$

Therefore, if we define $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + (\mathbf{L}\mathbf{x}^{(k+1)} - \mathbf{z}^{(k+1)})$, then (22) becomes

$$0 = \partial\|\mathbf{z}^{(k+1)}\|_1 + \frac{\rho}{\lambda}(\mathbf{z}^{(k+1)} - (\mathbf{L}\mathbf{x}^{(k+1)} + (\mathbf{u}^{(k+1)} - (\mathbf{L}\mathbf{x}^{(k+1)} - \mathbf{z}^{(k+1)})))) = \partial\|\mathbf{z}^{(k+1)}\|_1 - \frac{\rho}{\lambda}\mathbf{u}^{(k+1)}.$$

That is,

$$0 = \partial\lambda\|\mathbf{z}^{(k+1)}\|_1 - \rho\mathbf{u}^{(k+1)}.$$

Hence, with this particular updating procedure on $\mathbf{u}$, the dual feasibility conditions at each iteration are automatically satisfied.