Math 563 Final Project

Aidan Gerkis
Student ID: 260827581

April Niu
Student ID: 260763893

Antonios Valkanas
Student ID: 260672034

Cheng Shou
Student ID: 261008491

Linda Hu
Student ID: 260896450

Abstract

In this project we apply convex optimization methods to solve an image deblurring and denoising problem. We implement four different algorithms for deblurring problem and run numerical experiments to understand the dependence of each algorithm on its hyperparameters and select the best hyperparameter values. Further analysis is performed to understand how different blurring and noising parameters affect the best hyperparameter values. Three different error metrics, Root Mean Squared Error, Peak Signal-to-Noise Ratio, and Variation of Information are implemented to compare the performance of each algorithm. Finally, the algorithms are compared to each other both numerically and heuristically to draw conclusions.

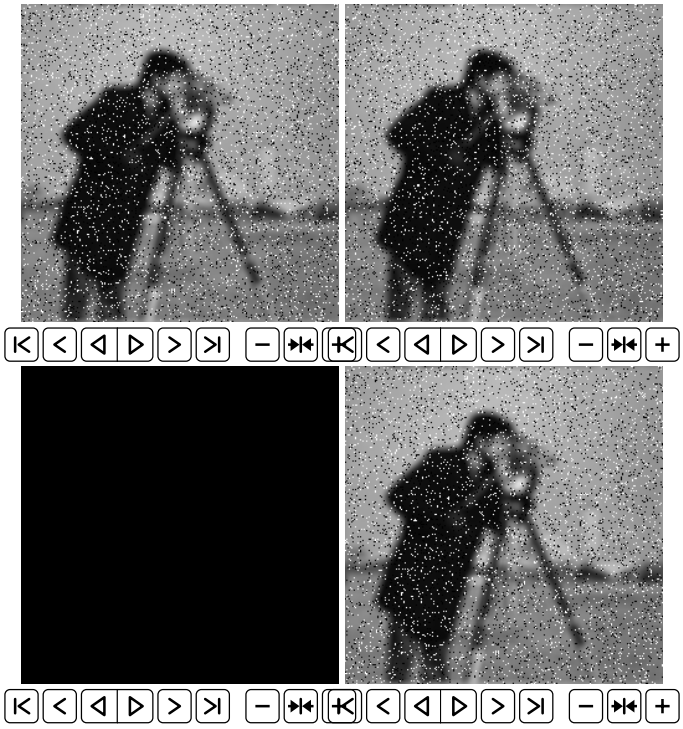


Figure 1: Algorithms in action! Top: Primal Douglas-Rachford (left), Primal-Dual Douglas-Rachford (right). Bottom: ADMM (left), Bottom: Chambolle-Pock (right). **Best viewed in Adobe Acrobat PDF viewer.**

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

1 Introduction

Modern photography is dominated by digital cameras which use CCD or CMOS sensors to capture images. These cameras are light, cheap, and reusable, but suffer from a susceptibility to noise and image blurring. As such fast and efficient deblurring and denoising algorithms are becoming a necessity in image processing. In this final project we address the challenges of image deblurring and denoising. The blurring of an image can be modeled as the convolution of the original image with a kernel plus additive noise: $k * x + \eta = b$. This model is linear, and thus can be represented as

$$Kx + \eta = b \tag{1}$$

where K is a matrix representing the convolution operation and x, η, and b are $n^2 \times 1$ vectors. The deblurring problem can then be formulated as an optimization problem

$$\min_x \phi_f(Kx - b) + \phi_r(x) + \phi_s(x) \tag{2}$$

where $\phi_f(\cdot)$ ensures the fidelity of the recovered image, $\phi_r(\cdot)$ ensures the pixel values stay within some acceptable range, and $\phi_s(\cdot)$ enforces denosing of the image. For the purposes of this report we convert this general problem into two specific formulations:

$$\min_x \quad \|Kx - b\|_1 + \delta_S(x) + \gamma\|Dx\|_{\text{iso}} \tag{3}$$

and

$$\min_x \quad \|Kx - b\|_2^2 + \delta_S(x) + \gamma\|Dx\|_{\text{iso}} \tag{4}$$

In these formulations the characteristic function of S guarantees the image's pixels are in the range of allowed intensities. The discrete gradient operator $D(x)$ is used to denoise the image and smooth the sharp edges. This operator is a concatenation of two discrete derivative operators: $\hat{D}_1$ for horizontal and $\hat{D}_2$ for vertical derivatives, represented as: $D = \begin{bmatrix} \hat{D}_1 \\ \hat{D}_2 \end{bmatrix}$. Two different fidelity terms, the L_1-norm and L_2-norm, are considered. These formulations are convex, and thus may be solved via convex optimization methods. In particular we apply four splitting algorithms to solve problems 1 & 1. We explore the performance difference between these two problems and determine the best hyperparameters for each algorithm, comparing the performance of each algorithm with the selected hyperparameters.

The rest of the report is organized as follows. Section 2 explains in detail the derivations of the proximal operators used in each algorithm. Section 3 gives a summary of the four different algorithms: Primal Douglas-Rachford Splitting, Primal-Dual Douglas-Rachford Splitting, Alternating Direction Method of Multipliers (ADMM), and Chambolle-Pock. Section 4 summarizes the error metrics which are used for hyperparameter tuning and algorithm comparison. Section 5 explains the hyperparameter tuning process and discusses the effects of each hyperparameter on algorithm performance. In section 6, we compare the performance of each algorithms. Finally, section 7 concludes the project.

2 Prox Operator Derivations

2.1 Box Proximal Operator

By definition the proximal operator of $\lambda f(x) = \lambda\delta_S(x)$ is given by

$$(P_\lambda f)(x) = (P_1 \lambda f)(x) = \text{argmin}_u\{\lambda\delta_S(x) + \frac{1}{2\lambda}||x - u||_2^2\} = \text{argmin}_u\{\frac{1}{2\lambda}||x - u||_2^2\} = \tag{5}$$

which is just the component-wise projection of the vector x onto $[0, 1]$, so each component satisfies $x_i \in [0, 1]$. Let p be the result of the projection. Then there are three cases: $x_i < 0 \to p_i = 0$, $x_i \in [0, 1] \to p_i = x_i$, and $x_i > 0 \to p_i = 1$. Note that this is equivalent to $p_i = \min\{\max\{0, x_i\}, 1\}$.

2.2 Proximal Operator of L1-Norm

By definition the proximal operator of $f(x) = ||x||_1$ is given by

$$(P_\lambda f)(x) = (P_1 \lambda f)(x) = \text{argmin}_u\{\lambda||u||_1 + \frac{1}{2}||x - u||_2^2\}$$

Define $h(u) = \lambda||u||_1 + \frac{1}{2}||x-u||_2^2\}$, then (Prop. 7.1.4)

$$(P_1\lambda f)(x) = \{\bar{u} \mid 0 \in \partial h(\bar{u})\}$$

Additionally, define $f(\cdot) = \lambda||\cdot||_1$, $g(\cdot) = \frac{1}{2}||x-\cdot||_2^2$. Then the qualification condition ($0 \in$ int(dom$f$)Ldom f)) clearly holds, and $f, g \in \Gamma$, so

$$\partial h(u) = \partial f(u) + \partial g(u)$$

where because g is differentiable and convex its subdifferential is just its gradient (Theorem 7.4.1), so

$$\partial g(u) = \nabla g(u) = u - x$$

Then because $f(x) = \lambda \sum_i = 1^n |x_i|$ is a separable of $f_i(u_i) = \lambda|u_i|$ we have (by HW 7.10)

$$\partial f(u) = \partial f_1(u_1) \times \partial f_2(u_2) \times ... \times \partial f_n(u_n)$$

where we have computed previously (HW 7.11) that

$$\partial f_i(u_i) = \begin{cases} \lambda & u_i > 0 \\ \{v | v \in [-\lambda, \lambda]\} & u_i = 0 \\ -\lambda & u_i < 0 \end{cases}$$

Then we have the following component-wise expression of the subdifferential of h,

$$\partial h_i(u_i) = \begin{cases} \lambda + u_i - x_i & u_i > 0 \\ \{v + u_i - x_i | v \in [-\lambda, \lambda]\} & u_i = 0 \\ u_i - x_i - \lambda & u_i < 0 \end{cases}$$

which we can solve case-wise. First consider the case $u_i > 0 \iff x_i > \lambda$:

$$0 \in \partial h_i(u_i) \iff 0 \in \lambda + u_i - x_i \iff u_i = x_i - \lambda.$$

Now consider the case $u_i < 0 \iff x_i < -\lambda$:

$$0 \in \partial h_i(u_i) \iff 0 \in u_i - x_i - \lambda \iff u_i = x_i + \lambda$$

And finally consider the case $u_i = 0 \iff |x_i| \leqslant \lambda$:

$$0 \in \partial h_i(u_i) \iff 0 \in \{v - x_i | v \in [-\lambda, \lambda]\} \iff u_i = 0$$

which gives the component-wise solution

$$[(P_1\lambda f)(x)]_i = \begin{cases} x_i - \lambda & x_i > \lambda \\ 0 & |x_i| \leqslant \lambda \\ x_i + \lambda & x_i < -\lambda \end{cases} = \max\{0, |x_i| - \lambda\} \cdot \text{sgn}(x_i) \tag{6}$$

2.3 Proximal Operator of L2-Norm Squared

Similarly, we'll also find the proximal operator $f(x) = ||x||_2^2$. This is given by:

$$(P_\lambda f)(x) = (P_1\lambda f)(x) = \text{argmin}_u\{\lambda||u||_2^2 + \frac{1}{2}||x-u||_2^2\}$$

Then (Prop. 7.1.4)

$$(P_1\lambda f)(x) = \{\bar{u} \mid 0 \in \partial h(\bar{u})\}$$

Define

$$h(u) = \lambda||u||_2^2 + \frac{1}{2}||x-u||_2^2,\ f(\cdot) = \lambda||\cdot||_2^2,\ g(\cdot) = \frac{1}{2}||x-\cdot||_2^2$$

with $f, g \in \Gamma$. Since the qualification condition clearly holds, we have that:

$$\partial h(u) = \partial f(u) + \partial g(u)$$

where, because f and g are both differentiable and convex, their subdifferentials are just their gradients (Theorem 7.4.1), so

$$\partial f(u) = \nabla f(u) = 2\lambda u, \text{ and } \partial g(u) = \nabla g(u) = u - x$$

Therefore, we have that:

$$\partial h(u) = 2\lambda u + u - x = (2\lambda + 1)u - x$$

Hence

$$0 \in \partial h(u) \iff 0 \in (2\lambda+1)u - x \iff x = (2\lambda+1)u \iff u = \frac{x}{2\lambda+1}$$

Our proximal operator is therefore:

$$(P_1\lambda f)(x) = \frac{x}{2\lambda+1} \tag{7}$$

2.4 Proximal Operator of Iso-Norm

We are interested in finding the proximal operator of the iso-norm, $f(x) = ||x||_{\text{iso}}$, defined as

$$(P_\lambda f)(x) = (P_1 \lambda f)(x) = \text{argmin}_u\{\gamma||u||_{iso} + \frac{1}{2}||x-u||_2^2\}.$$

Note that this function is a separable sum so (HW 7.10)

$$\partial h(u) = \partial h_1(z_1) \times \partial h_2(z_2) \times ... \times \partial h_n(z_n).$$

For $h(x_i, y_i) = \sqrt{x_i^2 + y_i^2} = ||z||_2$ consider $h^* = \delta_{||z||\leq 1}(z)$.

$$\text{prox}_{h^*}(z) = \text{argmin}\left\{\delta_{||u||\leq 1}(u) + \frac{1}{2}||z-u||_2^2\right\} = \min_{||u||\leq 1} \frac{1}{2}||z-u||_2^2 \tag{8}$$

$$= \text{proj}_{||u||_2\leq 1}(z) = \begin{cases} z & ||z||_2 < 1 \\ \frac{z}{||z||_2}, & ||z||_2 \geq 1. \end{cases} \tag{9}$$

Now we use the Moreau decomposition theorem [7][1]: $z = \text{prox}_{\gamma h}(z) + \gamma\,\text{prox}_{h^*/\gamma}(z/\gamma)$. Applying the decomposition and solving for $\text{prox}_{\gamma h}$ yields:

$$\begin{aligned}
\text{prox}_{\gamma h}(z) &= z - \gamma\,\text{prox}_{h^*/\gamma}(z/\gamma) \\
&= z - \gamma \text{proj}_{||u||_2\leq 1}(z/\gamma) \qquad \text{(this step follows from 9)} \\
&= z - \gamma \begin{cases} z/\gamma, & ||z/\gamma||_2 < 1 \\ \frac{z/\gamma}{||z/\gamma||_2}, & ||z/\gamma||_2 \geq 1 \end{cases} \\
&= z - \gamma \begin{cases} z/\gamma, & ||z||_2 < \gamma \\ \frac{z}{||z||_2}, & ||z||_2 \geq \gamma \end{cases} \\
&= \begin{cases} 0, & ||z||_2 < \gamma \\ z - \frac{z}{||z||_2}, & ||z||_2 \geq \gamma \end{cases} \\
&= \alpha\,(x_i, y_i)
\end{aligned}$$

where α

$$\alpha = \begin{cases} 0 & ||x_i^2 + y_i^2||_2 \leqslant \gamma \\ 1 - \frac{\gamma}{||x_i^2+y_i^2||_2} & else \end{cases}$$

which is equivalent to

$$\alpha = \max\left\{0, 1 - \frac{\gamma}{||x_i^2 + y_i^2||_2}\right\}. \tag{10}$$

3 Summary of Algorithms

In order to solve the problem mentioned above, we applied four different algorithms to recover the blurred black-and-white image: Primal Douglas-Rachford, Primal-Dual Douglas Rachford, ADMM, and Chambolle-Pock. These four algorithms solve this convex optimization problem in different ways and will be explained in detail below.

3.1 Primal Douglas-Rachford Splitting Algorithm

To solve the minimization problem mentioned in equation (1) and (2), the Primal Douglas-Rachford algorithm [3] splits this problem into 2 parts: $\min_x f(x) + g(x)$, where

$$f(x) = \delta_S(x) \tag{11}$$

[1]The general form of Moreau decomposition is a well-known identity, see: https://en.wikipedia.org/wiki/Proximal_gradient_methods_for_learning#Moreau_decomposition

$$g(x) = \|Kx - b\|_2^2 + \delta_S(x) + \gamma\|Dx\|_{\text{iso}} \text{ or } g(x) = \|Kx - b\|_1 + \delta_S(x) + \gamma\|Dx\|_{\text{iso}} \quad (12)$$

both functions f and g are convex, closed functions with proximal operators that are easy to calculate. Therefore, we may now consider the problem as finding an x such that $0 \in \partial(f + g(A))(x)$, where the sum rule can then simplify the problem into a straightforward addition [3]. Such a vector x can then be found by fixed point iteration.

3.2 Primal-Dual Douglas-Rachford Splitting Algorithm:

The Primal-Dual Douglas-Rachford algorithm [3] employs a similar approach to the Primal Douglas-Rachford. It also splits the original minimization problem into two functions. However, the Primal-Dual considers the maximization problem of dual functions $\max_y\{-f^*(-A^T y) - g^*(y)\}$ instead. Since f and g are convex, closed functions, Prop. 7.1.5 tells us that $\partial(f^*) = (\partial(f))^{-1}$ and $\partial(g^*) = (\partial(g))^{-1}$. Moreover, the generalized Moreau Decomposition (shown in section 2.4) provides us with a relationship between the primal and dual proximal operators. Implementing those techniques and applying fixed point iteration hence gives us the desired result.

3.3 Alternating Direction Method of Multipliers (ADMM):

The idea behind ADMM [5] [1] is also analogous to the Douglas-Rachford algorithms in that it splits our original minimization problem into two functions. It is different from the previous two algorithms in that it aims to minimize the dual problem: $f^*(-A^T(z)) + g^*(z)$. Using a change of variables and interchanging the order of the updates the minimization problem can then be transformed to a separable problem of the form: $\min f(u) + g(y)$ s.t. $Au + By = c$. Straightforward solution of this problem lead us to the desired outcome.

3.4 Chambolle-Pock:

The Chambolle-Pock algorithm [2], on the other hand, solves the problem by finding the solution of the saddle point problem: $\min_x \max_y \langle Ax, y\rangle + f(x) - g^*(y)$ Unlike the previous three algorithms, it does not rely on any solutions of linear equations, and instead, it only requires multiplication by A and A^T. Another unique feature of Chambolle-Pock is that the step sizes depend on the size of the operator, A, of the problem. Besides these differences differences, similarities exist between ADMM and Chambolle-Pock as well. In fact, Chambolle-Pock can be interpreted as a preconditioned version of ADMM [6].

4 Error Metrics

In our experiments we were given reference ground truth images for algorithm testing. To automate the parameter tuning process we derived metrics in the form of loss functions. These functions include: Root Mean Squared Error (RMSE) averaged over pixel intensity values, Peak Signal-to-Noise Ratio (PSNR) and the Variation of Information (VI) metric. These three metrics are used to quantify the difference between two signals or images, often used in the context of image processing or computer vision. RMSE measures the average of the squares differences between predicted and actual values. Advantages of RMSE include striaghforward computations and intuitive interpretation; lower values indicate better performance. Some limitations of RMSE are that it can be sensitive to outliers and that it doesn't account for perceptual differences; all errors are treated equally no matter if they are in the foreground or the background of the image. PSNR measures the ratio between the maximum power of a signal and the magnitude of corrupting noise. An advantage of PSNR is that it can reflect human perception better than RMSE, especially for visual data which is why it is a popular metric. VI measures information-theoretic distance between two distributions. We choose to employ a probabilistic metric to add robustness to our metrics and complement deterministic metrics with probabilistic one. While this improves metric diversity it is not as interpretable or as straightforward as RMSE or PSNR. The RMSE and PSNR errors are computed simply as

$$\text{MSE}(X, Y) = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [X(i,j) - Y(i,j)]^2. \quad (13)$$

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{14}$$

To compute the VI error consider forming a histogram over pixel intensity values of image X. Depending on the size and number of the bins of the histogram we may note the number of pixels that belong to each bucket. Normalizing this leads to a discrete probability distribution over the pixel intensity values of X, where for bin i we have probability p_i. In this sense, we can view any image as a sample from a (potentially very high dimensional) discrete probability space. To define the VI metric, we recall the definitions of entropy $H(X)$ and mutual information $I(X, Y)$ between discrete distributions X, Y.

$$H(X) = -\sum_i p_i \log p_i, \tag{15}$$

$$I(X;Y) = D_{\mathrm{KL}}(P_{(X,Y)} \| P_X \otimes P_Y) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} P_{(X,Y)}(x,y) \log\left(\frac{P_{(X,Y)}(x,y)}{P_X(x)\, P_Y(y)}\right), \tag{16}$$

$$\mathrm{VI}(X;Y) = H(X) + H(Y) - 2I(X,Y) \tag{17}$$

RMSE and PSNR are primarily used for comparing the fidelity of images or signals. They provide a direct measure of the difference between the original and reconstructed data. However, they may not capture semantic differences well, especially in image processing tasks where small changes might not be perceptible. VI, on the other hand, is more suitable for comparing segmentations or clusterings. It can capture differences in structure and organization, which may not be reflected in pixel-wise error metrics like RMSE or PSNR. Together, these metrics provide a comprehensive evaluation of both fidelity and structural differences in data. While RMSE and PSNR focus on fidelity, VI provides insights into structural differences, thus complementing each other in evaluating various aspects of the data from a deterministic as well as probabilistic point of view. As we can see in Fig. 2, our metrics correlate strongly and saturate after a similar number of iterations. This is not surprising as RMSE and PSNR can be related by a direct transformation. The VI metric, which is totally independent to RMSE and PSNR, follows the same trend, which is indicating that a "good" image and/or converged algorithm can be detected automatically regardless of the specific error metric choice. For this report we chose to use RMSE due to its simple and direct interpretability.

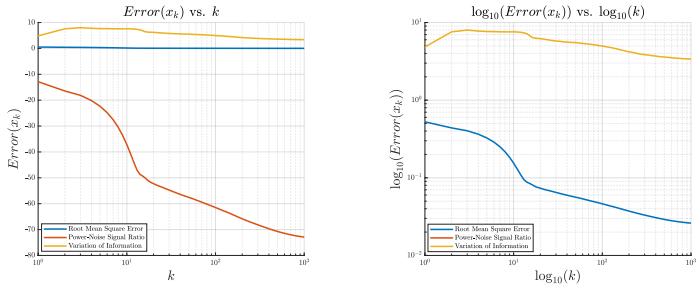


Figure 2: Comparison of different metrics computed on the same problem (deblurring the cameraman image blurred with 9×9 gaussian blur and standard deviation 4). Left: all metricsversus log(iterations), Right: zoomed in view of loglog plot of RMSE and VI.

5 Hyperparameter Selection

The algorithms developed and implemented each have several parameters which must be set by the user. These parameters, hereafter referred to as hyperparameters, are t, ρ, & γ for the Primal Douglas-Rachford, Primal-Dual Douglas-Rachford, and ADMM algorithms and t, s, & γ for Chambolle-Pock. Each of these hyperparameters controls different aspects of the algorithms performance and will be discussed in detail here. The parameter γ modifies the iso-norm term in the loss function. Since the iso-norm attempts to enforce a minimal difference between two adjacent pixels in the image, thus reducing noise, the γ term effectively controls the level of denoising that the algorithm performs. Larger values of γ correspond to more noise reduction while smaller values correspond to less noise reduction. As a consequence of this we expect that larger values of γ will result in a smoothing effect on the image, where sharp edges become poorly defined, while smaller values of γ will result in better detail recovery at the cost of reduced noise reduction. In the Primal Douglas-Rachford

and Primal-Dual Douglas-Rachford algorithms the parameter t scales the value of the iso-norm and deblurring term in the computation of the proximal operator of g,

$$P_{tg}(y_1, y_2, y_3) = \min_x \{t||x_1 - b||_1 + t\gamma||(y_2, y_3)||_{iso} + \frac{1}{2}||x - y||^2\} \tag{18}$$

In this (and following) equations the L_1-norm is used, although this may be replaced by the L_2-norm and an equivalent interpretation arises. If the value of t is less than 1 the loss function will be scaled down, meaning that at the point u^* minimizing (18) the norm values can be larger, thus resulting in smaller changes in the image guess at each iteration. Similarly, if the value of t is greater than 1 the loss function will be scaled up, so at the point u^* minimizing (18) the norm values will need to be small, thus resulting in larger changes in the image guess at each iteration. The same interpretation of the t and s hyperparameters can be applied to the Chambolle-Pock algorithm, as they also scale the functions on which the proximal operators are being computed. Thus, the hyperparameters t and s should act as step-sizes, affecting the convergence rate of the algorithms. The hyperparameter ρ also controls step-size, as it explicitly controls the amount by which the deblurred image is changed at each iteration

$$x^{(k+1)} = x^{(k)} + \rho\Delta^{(k)} \tag{19}$$

where $\Delta^{(k)}$ is the computed change in the image guess at iteration k. Equation 19 tells us that a larger value of ρ will result in bigger steps at each iteration, while smaller values will result in smaller steps. Based on this analysis we hypothesize that the parameters t and $\rho \backslash s$ will affect the convergence rate of the iterates, while the parameter γ will affect the noise reduction. In particular we expect small values of γ to be desirable when low amounts of noise are present and large values of γ to be required when noise density is high.

To choose the values of the hyperparameters for each algorithm we ran a 3 dimensional sweep of the hyperparameter space consisting of 10 values of each parameter, for a total of 1000 candidate points. A small discretization of each hyperparameter was chosen due to computational limitations; larger sweeps were computationally infeasible as the number of algorithm iterations required grew with the cube of the discretization size. To surmount this limitation we implemented an iterative search method that first performed a coarse search on the hyperparameter space, then shrank and refined the search space at each iteration based on the results of the previous sweep. To implement this method a large range of hyperparameter values is first swept to provide an initial sample of the loss function at each point in the hyperparameter space. Algorithm performance is quantified by considering the loss function value, defined as

$$f(x^{(k)}) = ||Kx^{(k)} - b||_1 + \gamma||Dx^{(k)}||_{\text{iso}} + \delta_S(x^{(k)})$$

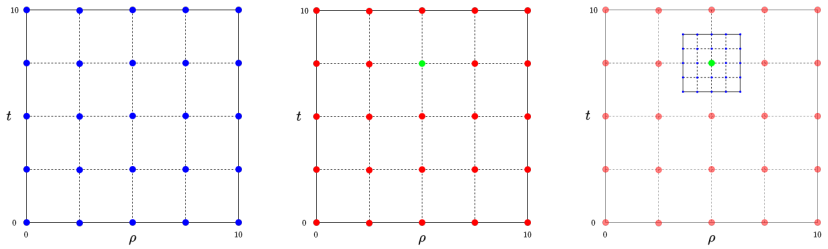
at the final iterate of each algorithm run. The best set of hyperparameters (meaning those hyperparameters resulting in the lowest value of $f(x_{final})$) from this initial sweep is chosen and a smaller sweep space is constructed to be centered on this point. This process is repeated until the difference between the best hyperparameters found in two consecutive iterations is below some threshold value. A visualization of this sweeping algorithm is shown in Figure 3. Computation time was further reduced by implementing an early stop criteria, which stopped the algorithm iterations when the difference in loss between two iterates was less than some threshold

$$f(x^{(k+1)}) - f(x^{(k)}) < \sigma_T$$

where σ_T was chosen to be 0.1 for the purposes of this sweep. Implementing this method allowed us to capture the global behaviour of the algorithms with respect to the hyperparameter values while ensuring that close to optimal hyperparameter values were selected. The sweep was performed on the cameraman image, blurred with a 9×9 Gaussian blur of standard deviation 4 and with salt & pepper noise of density 0.1 applied.

Observing the deblurred images from these sweeps we observed that the final images obtained with the hyperparameters resulting in the best loss function value displayed significant levels of noise, a phenomenon caused by the sweeps tendency to set γ to the lowest possible value. This can be explained by considering the iso-norm term in the loss function, which operates on the quantity

$$Dx^{(k)} = \begin{bmatrix} \hat{D}_1 \\ \hat{D}_2 \end{bmatrix} x^{(k)}$$



(a) First select a large range of parameters to test and discretize the range. Here a discretization into 5 points is performed. Blue points indicate pairs of hyperparameter values to test.

(b) Run the hyperparameter sweep on each pair of hyperparameters identified in the previous step, selecting the pair resulting in the lowest loss function value, highlighted in green.

(c) Select a smaller range of hyperparameters, centered on the best point identified in step b) and run the hyperparameter sweep on the resulting set of hyperparameter values.

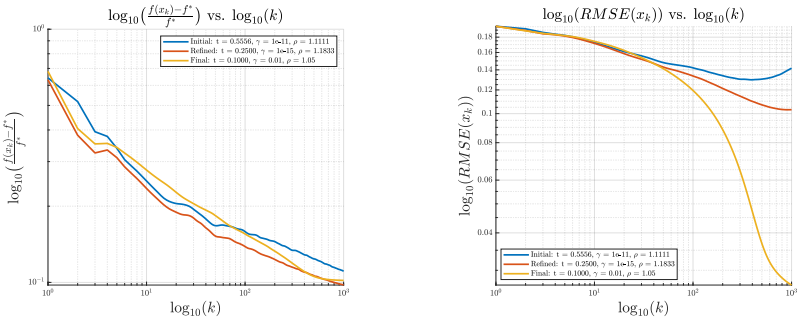
Figure 3: Visualization of the iterative hyperparameter sweep method for a 2D hyperparameter space.

Table 1: Hyperparameter values resulting in the lowest loss function value, $f(x_{final})$, at each step of the hyperparameter sweep.

Primal Douglas-Rachford				**Primal-Dual Douglas-Rachford**			
Sweep Step	t	ρ	γ	Sweep Step	t	ρ	γ
Initial	0.5556	1.1111	$1e-15$	Initial	1.0000	1.0000	0.5000
Refined	0.2500	1.1833	$1e-15$	Refined	1.6667	1.7778	$1e-11$
Final	**0.1000**	**1.0500**	**0.0100**	Final	**1.7500**	**1.7500**	**0.0500**
ADMM				**Chambolle-Pock**			
Sweep Step	t	ρ	γ	Sweep Step	t	s	γ
Initial	1.0000	1.0000	0.0500	Initial	1.0000	1.0000	0.5000
Refined	0.4444	1.3333	$1e-11$	Refined	0.2222	0.5556	$1e-11$
Final	**0.7407**	**1.4815**	$\mathbf{1e-15}$	Final	**0.1000**	**0.4072**	$\mathbf{1e-15}$

where $\hat{D}_1$ & $\hat{D}_2$ represent the horizontal and vertical discrete derivative operators, respectively. The iso-norm term in the loss function acts to reduce this quantity, making the difference between adjacent pixels as small as possible. When minimizing the loss function the algorithms must then find a balance between reducing noise (by minimizing the iso-norm term and recreating detail in the image) and deblurring the image (by minimizing the $L1$-norm or $L2$-norm term). Higher values of γ place a greater importance on noise reduction, forcing adjacent pixels to be close together in value. This results in poor recreation of fine details and edges in images, thus resulting in a high value of the $L1$-norm or $L2$-norm term and producing a larger value of $f(x_{final})$. The opposite effect is observed if γ is small: the iso-norm term is small, meaning that the values of adjacent pixels may differ significantly without having a large effect on the value of the loss function, thus the final image can better capture fine details and edges. Therefore, small γ values result in small values of $f(x_{final})$, causing the hyperparameter sweep to prefer small values of γ. However, the images resulting from these sets of hyperparameters were quite noisy, due to the low importance that was placed on denoising by the selection of a small γ value. To obtain better denoising in the final image, the values of γ obtained from the hyperparameter sweeps were manually tuned until a visually appealling image was found. This manual tuning also resulted in the final hyperparameter values obtaining a lower finishing RMSE value, indicating that the manual tuning was able to better recover the true image.

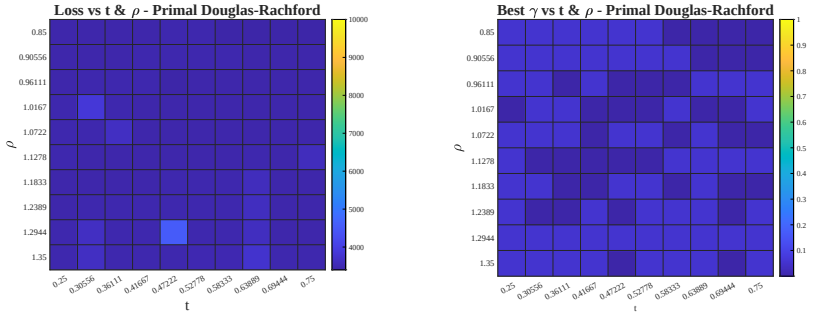
The sweep results at each iteration are summarized in Table 1, with the final hyperparameter values implemented in our software package highlighted in bold. We can see that there is very little agreement in the best hyperparameter values between each algorithm, indicating that each algorithm depends differently on the hyperparameter values.



(a) Log-Log plots of relative loss versus iterations.

(b) Log-Log plots of RMSE versus iterations.

Figure 4: Convergence plots of relative loss and RMSE for the Primal Douglas-Rachford algorithm with hyperparameter sets found via the hyperparameter sweeping method.



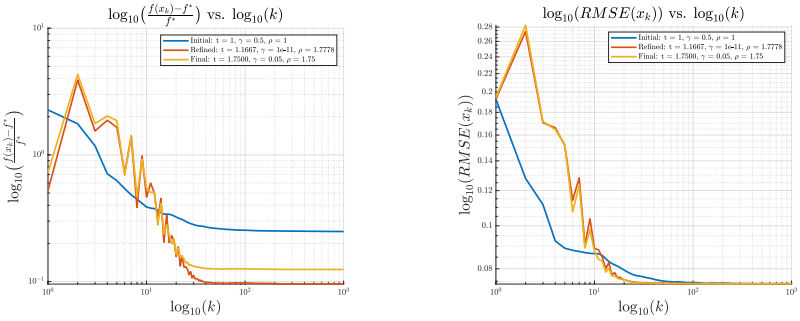
(a) Heatmap of $f(x_{final})$ versus ρ & t with γ fixed to the value resulting in the lowest value of $f(x_{final})$.

(b) Heatmap of γ resulting in the lowest $f(x_{final})$ for each combination of ρ & t.

Figure 5: Heatmaps of loss and γ value obtained from the initial hyperparameter sweep of the Primal Douglas-Rachford algorithm.

To further analyze the algorithm performance with varying hyperparameter values we plotted the convergence of the relative loss, $\frac{f(x_k)-f^*}{f^*}$, where f^* denotes the loss function evaluated at the true image, and RMSE versus iterations for the best hyperparameter values found in each step of the hyperparameter sweep described above. For the Primal Douglas-Rachford algorithm, Figure 4, the parameters at each step displayed similar convergence rates in the relative loss, with the final loss decreasing between the initial and refined sweeps. The manual tuning of γ resulted in the finishing loss with the final hyperparameter values to be slightly higher than the finishing loss obtained in the refined sweep. However, the final parameters obtained significantly faster convergence in RMSE than the initial and refined hyperparameters, with the finishing RMSE value being significantly lower than that obtained with these hyperparameter sets. The finishing loss function value was also plotted versus t & ρ, with γ fixed to the value resulting in the lowest finishing loss (over all values of t, ρ, γ). These results, shown in Figure 5a, show that the Primal Douglas-Rachford algorithm is not very sensitive to the values of t & ρ, as we observe a relatively constant value of $f(x_{final})$ across the majority of the range of t & ρ. The value of γ obtaining the lowest value of $f(x_{final})$ was also plotted versus t & ρ in Figure 5b. This plot showcases the aforementioned phenomenon where very small γ (as close to 0 as the hyperparameter range allows) is preferred almost uniformly.

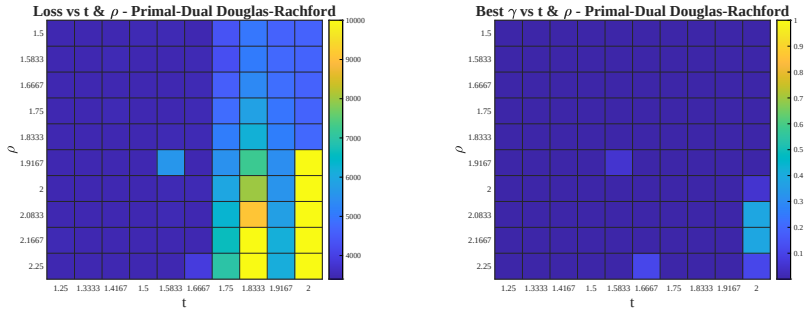
A more significant difference in the convergence rate of the initial hyperparameter sets and the refined and final hyperparameters was noticed for the Primal-Dual Douglas-Rachford method, the plots of which are shown in Figures 6. Similarly to the results of the Primal Douglas-Rachford algorithm analysis, the best finishing loss found with the refined sweep is lower than the finishing loss of the final hyperparameter selection, which is expected due to the manual tuning of the γ parameter. This



(a) Log-Log plots of relative loss versus iterations.

(b) Log-Log plots of RMSE versus iterations.

Figure 6: Convergence plots of relative loss and RMSE for the Primal-Dual Douglas-Rachford algorithm with hyperparameter sets found via the hyperparameter sweeping method.



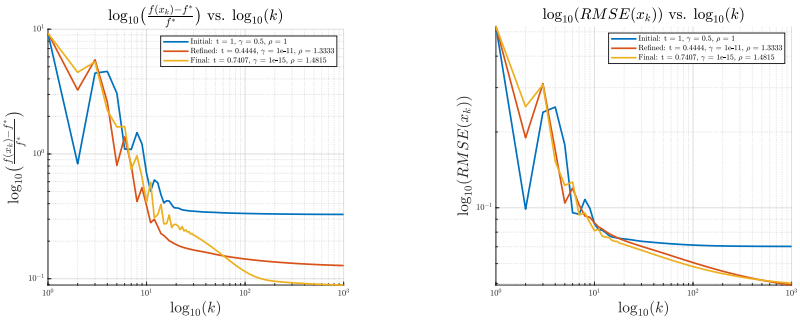
(a) Heatmap of $f(x_{final})$ versus ρ & t with γ fixed to the value resulting in the lowest value of $f(x_{final})$.

(b) Heatmap of γ resulting in the lowest $f(x_{final})$ for each combination of ρ & t.

Figure 7: Heatmaps of loss and γ value obtained from the initial hyperparameter sweep of the Primal-Dual Douglas-Rachford algorithm.

manual tuning is justified by the RMSE convergence, which is observed to be slightly faster with the final hyperparameters than with the refined hyperparameters. The difference in algorithm performance is less clear when observing the plots of RMSE convergence, as while the hyperparameters obtained in the refined and final sweeps result in slightly faster convergence, all algorithms obtain a similar finishing RMSE. Because of the similar performance in RMSE convergence between the refined and final sweeps we opted to choose the hyperparameters found in the final sweep as the best ones, since they attained a slightly faster convergence rate. The heatmaps of $f(x_{final})$ versus t & ρ with fixed γ and of γ obtaining the lowest value of $f(x_{final})$ versus t & ρ were also plotted in Figure 7. The results show that the performance of the Primal-Dual Douglas-Rachford algorithm has a slightly higher dependence on the hyperparameter values, with the best attainable $f(x_{final})$ differing noticeably over the range of hyperparameters. However, the same phenomenon is observed where the lowest value of γ is preferred almost uniformly for all values of t & ρ.

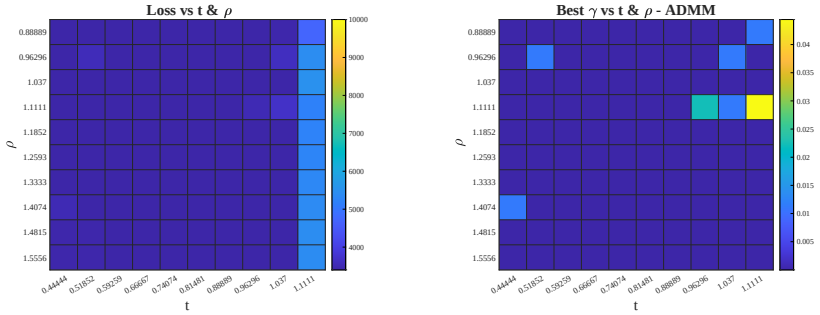
The results of the ADMM hyperparameter sweep, Figure 8, show a similar convergence rate for each hyperparameter set. However, there is a significant difference between the finishing values of relative loss with each of these sets, with the finishing loss decreasing from the initial sweep results to the final sweep results. The final selected hyperparameters obtain distinctly better performance (in loss). It is interesting to note that the finishing RMSE values obtained with the refined and final hyperparameter sweep results are very similar, despite a significant difference in the finishing loss values for these two algorithms. This tells us that there is some discrepancy between the loss used in the optimization formulation of the problem and the accuracy of the resulting image to the true image. The heatmaps of the best value of $f(x_{final})$ and the value of γ obtaining the lowest $f(x_{final})$, plotted in Figure 9, show that the performance of ADMM has a small dependence on hyperparameter



(a) Log-Log plots of relative loss versus iterations.

(b) Log-Log plots of RMSE versus iterations.

Figure 8: Convergence plots of relative loss and RMSE for the ADMM algorithm with hyperparameter sets found via the hyperparameter sweeping method.



(a) Heatmap of $f(x_{final})$ versus ρ & t with γ fixed to the value resulting in the lowest value of $f(x_{final})$.

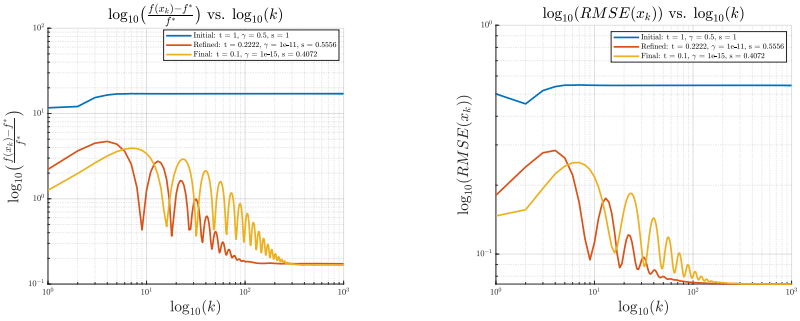
(b) Heatmap of γ resulting in the lowest $f(x_{final})$ for each combination of ρ & t.

Figure 9: Heatmaps of loss and γ value obtained from the initial hyperparameter sweep of ADMM.

values, with the finishing loss function value varying only slightly with t & ρ. Figure 9b shows the same phenomenon observed in the other algorithms, with low γ being preferred almost uniformly for all values of t & ρ.

Sweeping hyperparameter values for the Chambolle-Pock algorithm revealed that this method's performance, plotted in Figure 10, is very dependent on the hyperparameter values. We observed that the best hyperparameter values found in the initial sweep result in a loss value that stays roughly constant with iterations, and is significantly higher than the loss obtained with the results of the refined and final sweeps. Additionally, a significant difference in convergence rates is noted between the refined and final hyperparameter values, with the final values taking longer to converge while obtaining a marginally lower finishing loss. The same behaviour is noticed in the RMSE convergence as in the loss convergence. While the final hyperparameters are able to obtain the lowest finishing loss, the performance gain is marginal, and is not noticeable in the finishing RMSE values (when compared to the refined hyperparameter values). This performance gain also comes at the cost of significantly slower convergence, requiring more than 100 additional iterations to obtain the lower loss value. The heatmap of $f(x_{final})$ versus t & ρ with fixed γ, Figure 11a, validate the observation that performance of the Chambolle-Pock algorithm has a high dependence on the hyperparameter values. Observing the heatmap of γ obtaining the lowest $f(x_{final})$, Figure 11b, we see the same phenomenon observed with the other algorithms: the lowest value of γ is preferred almost uniformly across the tested range of hyperparameters.

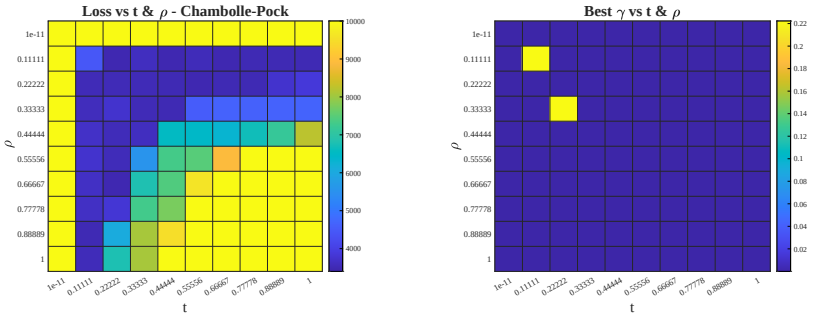
We observed that the dependence of algorithm performance on the hyperparameter values varied for each algorithm. The Chambolle-Pock algorithm showed the strongest dependence on the hyperparameter values, with hyperparameters that differed significantly from the final values resulting in



(a) Log-Log plots of relative loss versus iterations.

(b) Log-Log plots of RMSE versus iterations.

Figure 10: Convergence plots of relative loss and RMSE for the Chambolle-Pock algorithm with hyperparameter sets found via the hyperparameter sweeping method.



(a) Heatmap of $f(x_{final})$ versus ρ & t with γ fixed to the value resulting in the lowest value of $f(x_{final})$.

(b) Heatmap of γ resulting in the lowest $f(x_{final})$ for each combination of ρ & t.

Figure 11: Heatmaps of loss and γ value obtained from the initial hyperparameter sweep of the Chambolle-Pock algorithm.

significantly worse algorithm performance. Even small changes in the hyperparameter values significantly affected the convergence rate of the Chambolle-Pock algorithm. On the other hand, the Primal Douglas-Rachford & ADMM algorithms showed the weakest dependence on the hyperparameter values, with different hyperparameter values resulting in only small changes to convergence rate and finishing values of the loss and RMSE.

5.1 Effects on Hyperparameters

In this section, we describe how different kernels, noise types, noise-densities, and deblurring terms influence the choice of hyperparameters (i.e., the values that minimizes the loss). The results are summarized in table 3, 4, 5 and 6, respectively, in the Appendix. For all the experiments, we vary one of the aforementioned parameters and fix the others. Figure 12 shows some example images before and after deblurring using various kernels and noise types.

The default configuration is: Gaussian filter with hsize = $[9, 9]$ and standard deviation $\sigma = 4$, ADMM, $L1$ prox, and salt & pepper with noise density = 0.1. This default configuration is used in all tests in this section. For example, when studying the effects of kernel choice on hyperparameters, we fix everything else in the configuration and vary only the kernels. Additionally, we also use the default parameters from MATLAB for all the kernels and noise types except Gaussian filter with hsize = $[9, 9]$ and standard deviation $\sigma = 4$.

We observed that the best value of γ is always $1e - 15$, independent of the experiment. Similarly, the best ρ is always 1, except when for the Laplacian kernel, where ρ is almost 0. However, the

(a) Motion (blurred) (b) Poisson (blurred) (c) Noise density 0.5 (blurred) (d) $L2$ prox operator (blurred) (e) Motion (deblurred) (f) Poisson (deblurred) (g) Noise density 0.5 (deblurred) (h) $L2$ prox operator (deblurred)

Figure 12: The blurred cameraman images using various kernel and noise types (densities) and resulting deblurred images obtained with the best ADMM hyperparameters after 1000 iterations.

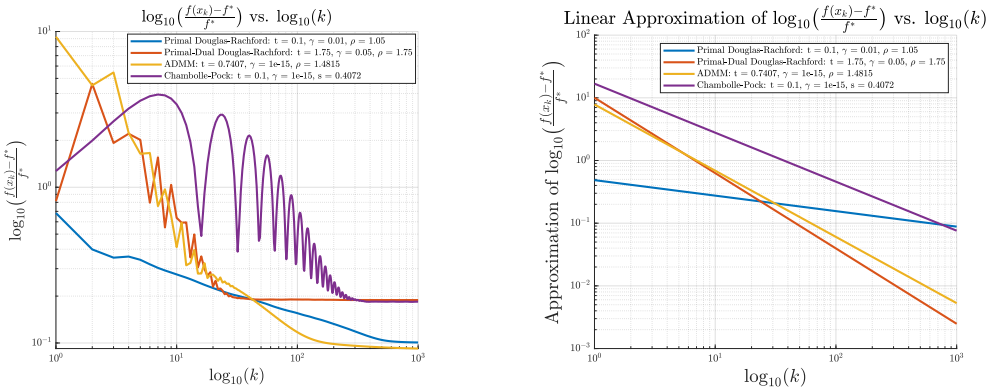
optimal step size changes significantly as we vary among different kernels, noises (densities), and prox-operators. This might be attributed to the choice of the ADMM algorithm, whose step-size is hard to select [6]. This fact is also reflected in our experiment, where the optimal step sizes change significantly as we choose different kernels, noises and prox-operators. In general we observed that the best hyperparameter values differ significantly depending on the blur and noise applied to the image. The quality of the output image is also affected, as the ADMM algorithm can be seen to produce worse images for higher noise densities. Additionally, we observed that the $L2$-norm deblurring term results in higher relative loss and a worse output image than the $L1$-norm term. Therefore we recommend the use of the $L1$-norm for image deblurring.

6 Algorithm Comparison

To compare the algorithms against each other, each algorithm was run on the cameraman image, blurred and noised with a 9×9 gaussian blur with standard deviation 4 and salt & pepper noise of density 0.1. For each algorithm the best hyperparameters found with the hyperparameter sweep performed in Section 5 were used and the algorithm was run for 1000 iterations. To compare the four algorithms we considered four performance metrics:

1. Relative loss, computed as $(f(x_k) - f^*)/f^*$
2. RMSE, computed according to Equation 13
3. Convergence rate, κ, computed as the slope of the best-fit line to the relative loss vs iterations.
4. Computation time, computed as the time for one complete algorithm iteration to complete.

We also compared the algorithms based on the heuristic quality of the image. The relative loss versus iteration was plotted in Figure 13a for each algorithm, with the resulting best fit lines, whose slope represents the convergence rate, κ, plotted in Figure 13b. The RMSE was also plotted versus iteration for each algorithm in Figure 14. The values of the final relative loss, final RMSE, computation time, and κ, which is computed by fitting a line to the linear segments of the log-log plots in Figure 13a, are summarized in Table 2. The computation time was measured on a system with an Intel Core i7-12700H, 2.3GHZ processor with 14 cores and 16 GB of RAM and was computed as the average computation time of 1000 iterations of the algorithm. From these analyses it is clear that the Primal Douglas-Rachford and ADMM algorithms obtain the lowest relative loss values, while Primal-Dual Douglas-Rachford and Chambolle-Pock attain similar values of relative loss. Analyzing the RMSE we can see a similar behaviour in the performance of the Primal-Dual Douglas-Rachford and Chambolle-Pock algorithms. However, we observed that the Primal Douglas-Rachford algorithm is able to attain significantly lower RMSE values than the ADMM value, meaning that while the two algorithms behave similarly with respect to the relative loss, the Primal Douglas-Rachford algorithm is able to deblur the image with higher accuracy. This improved performance comes with the cost of slower convergence; the Primal Douglas-Rachford algorithm converges to its final RMSE value much slower than ADMM, and the RMSE value is significantly higher than that of the other algorithms until more than 500 iterations have been performed. The two algorithms with the fastest convergence rate are ADMM and Primal-Dual Douglas-Rachford, which both converge with a rate $\kappa > 1$. Chambolle-Pock converges slightly slower than the ADMM and Primal-Dual Douglas-Rachford algorithms while Primal Douglas-Rachford converges significantly slower than all of the other algorithms. We can conclude that all of the algorithms converge linearly with the number of iterations, k, as the log-log plot of relative loss versus k may be easily approximated by a linear function for all algorithms. Based



(a) Log-Log plots of relative loss versus iteration for each algorithm.

(b) Best fit linear model of relative loss versus iteration for each algorithm.

Figure 13: Convergence plots of relative loss for each algorithm, evaluated over 1000 iterations on the cameraman image, blurred with a 9×9 gaussian blur with standard deviation 4.

Table 2: The values of the chosen performance metrics for each algorithm, evaluated on 1000 iterations of the algorithm on the cameraman image, blurred with a 9×9 gaussian blur with standard deviation 4.

Performance Metrics				
Algorithm	$\frac{f(x_{final})-f^*}{f^*}$	RMSE	κ	Computation Time
Primal Douglas-Rachford	0.1008	0.0274	0.2472	0.0972s
Primal-Dual Douglas-Rachford	0.1887	0.0740	1.2021	0.0997s
ADMM	0.0921	0.0501	1.0568	0.1546s
Chambolle-Pock	0.1841	0.0740	0.7819	0.0884s

on these results we can say that the Primal Douglas-Rachford algorithm is the best algorithm in terms of the relative loss function and RMSE and is able to obtain the best approximation of the original image out of the tested algorithms. However, this comes with a slow convergence rate; the Primal Douglas-Rachford algorithm can obtain strong results only if the algorithm is run for many iterations. As a trade-off between accuracy and speed the ADMM algorithm is a good choice. It is able to obtain better relative loss and RMSE than the Primal Douglas-Rachford and Chambolle-Pock algorithms while converging significantly faster than Primal Douglas-Rachford. These characteristics make this algorithm a strong choice for real-world applications, where computational constraints play a strong role in algorithm selection. One major downside of the ADMM algorithm is its computation time, which is significantly higher than any of the other algorithms. Therefore, if the computational capabilities of your system are quite weak the Primal-Dual Douglas-Rachford algorithm may be a better choice as it provides a similar convergence rate to the ADMM algorithm but with lower computation time.

Each algorithm was then tested on three test images and the resulting images were compared heuristically. In each test case the image was blurred with a 9×9 Gaussian blur with standard deviation 4 and salt & pepper noise was applied with density 0.1. To deblur the images each algorithm was run for 1000 iterations using the best hyperparameters found in section 5. The resulting deblurred images are displayed in Figure 15 for the cameraman image with their original and blurred counterparts. Figures 20, & 21 in the appendix provide the same comparison for the man with hat and McGill images. In each test case we observed that the reproduced images computed via the Primal Douglas-Rachford & ADMM algorithms ((c) & (e) in each figure) were the most accurate to the original image; both algorithms were able to recover most of the detail in the image and remove all of the noise. This observation aligns with the conclusions drawn from analysis of the relative loss & RMSE, where the Primal Douglas-Rachford algorithm was expected to have the best performance as it obtained the lowest RMSE on the test image. The ADMM algorithm displays a rippling behaviour

(a) The cameraman image, without blur.

(b) The cameraman image with a 9×9 gaussian blur with standard deviation 4.

(c) The cameraman image deblurred with Primal Douglas-Rachford.

(d) The cameraman image deblurred with Primal-Dual Douglas-Rachford.

(e) The cameraman image deblurred with ADMM.

(f) The cameraman image deblurred with Chambolle-Pock.

Figure 15: The cameraman image before blurring (a), after blurring (b) and after deblurring with each algorithm (c)-(f). The image was blurred with a 9×9 gaussian blur of standard deviation 4 and salt & pepper noise with density 0.1 was applied. To deblur the image each algorithm was run for 1000 iterations using the best hyperparameters found in Section REFERENCE.

around the edges of objects, see Figure 22 in the appendix, which may indicate that the algorithm is putting a larger importance on the denoising term, as features in the image appear to be replicated in adjacent pixels. The Primal-Dual Douglas-Rachford & Chambolle-Pock algorithms performed significantly worse on each test image. While they were able to remove noise from the image, they struggled to recover finer details, like the facial features of the cameraman or the feathers in the man with hat image. The conclusions from this heuristic analysis align with those obtained by analysis of the relative loss & RMSE: the Primal Douglas-Rachford algorithm results in the best reconstruction of the original image, with the Primal-Dual Douglas-Rachford & Chambolle-Pock algorithms performing significantly worse.

7 Conclusion

In this project we solved the image deblurring and image denoising problem by applying four different algorithms based on the theory of convex optimization. Three error metrics, RMSE, PSNR, and VI, were proposed to measure the accuracy of the recovered images and compared to show their equivalency. The behaviour of each algorithm with respect to its hyperparameters was analyzed numerically with a 3d sweep of the hyperparameter space. Using the results of this sweep the hyperparameter values for each algorithm were chosen to obtain the smallest relative loss. The relationship between the best hyperparameter values and the noise and blur type was analyzed numerically for each algorithm, showing that there is a strong dependence on the type of blur and noise and the resulting best hyperparameter values. Additionally, we found that the $L1$-norm deblurring term produces much better results thant the $L2$-norm deblurring term. We concluded that the Primal Douglas-Rachford algorithm best recovers the true image from the blurred image but converges slowly. ADMM performs strongly as well, but has a higher computation time per iteration. The Chambolle-Pock algorithm was the easiest to implement, but has a strong dependence on its hyperparameters, making it difficult to tune to obtain good results. The Primal Douglas-Rachford algorithm is the best algorithm overall due to its simple implementation, strong performance and, weak dependence on hyperparameter values.

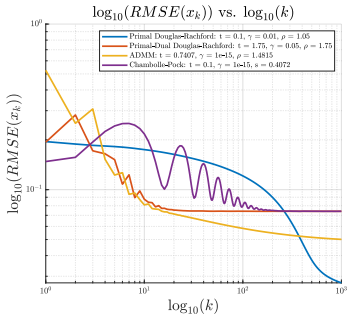


Figure 14: Log-Log plots of RMSE versus iteration for each algorithm, evaluated over 1000 iterations on the cameraman image, blurred with a 9×9 gaussian blur with standard deviation 4.

Acknowledgments

Aidan Gerkis (Student ID: 260827581) Derived box proximal operator. Implemented proximal operators in MATLAB and helped to debug Primal Douglas-Rachford, Primal-Dual Douglas-Rachford, and ADMM algorithms. Worked with Antonios to implement Chambolle-Pock. Wrote imopt package to implement deblurring algorithms. Performed hyperparameter sweeps to select best hyperparameter values and compare algorithm performance. Contributed to the final project write-up and final edits.

Antonios Valkanas (Student ID: 260672034) Derived iso-prox. Worked with Aidan for Chambolle-Pock implementation as well as generating the animation in Figure 1 of this paper. Decided which metrics to use for the algorithm and implemented them in matlab. Contributed in the final project write-up and final edit round.

April Niu (Student ID: 260763893) Worked with Cheng and Linda on drafting the first three algorithms, testing the effect of noises and some other parameters has on the 4 algorithms. Contributed in the final project write-up.

Cheng Shou (Student ID: 261008491) Worked with Linda and April on drafting the first three algorithms, testing the effect of noises and some other parameters has on the 4 algorithms. Contributed in the final project write-up.

Linda Hu (Student ID: 260896450) Derived L2-squared proximal operator. Worked with Cheng and April on drafting the first three algorithms, testing the effect of noises and some other parameters has on the 4 algorithms. Contributed in the final project write-up.

References

[1] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3:1–122, jan 2011.

[2] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Imaging Vision*, 40(1):120–145, 2011. ISSN 0924-9907. doi: 10.1007/s10851-010-0251-1. URL https://doi-org.proxy3.library.mcgill.ca/10.1007/s10851-010-0251-1.

[3] Jonathan Eckstein and Dimitri Bertsekas. On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(3): 293–318, 1992.

[4] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2nd ed. edition, 2004.

[5] R. Glowinski and A. Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(R2):41–76, 1975.

[6] Daniel O'Connor and Lieven Vandenberghe. Primal Dual Decomposition by Operator Splitting and Applications to Image Deblurring. *Siam Journal Imaging Sciences*, 7(3), 2014.

[7] R.T. Rockafellar and R.J.-B. Wets. *Variational Analysis*. Springer, Berlin, 1998.

A Appendix

A.1 Hyperparameter Dependence on Blur & Noise Type

The tables in this section summarize the best hyperparameter values found for the ADMM algorithm when using different blurs and noise types, supplementing the analysis in subsection 5.1. A heuristic comparison of the different deblurring terms ($L1$-norm and $L2$-norm) is also provided through comparison of the cameraman image blurred with a 9×9 gaussian blur of standard deviation 4 and with salt & pepper noise of density 0.1 applied and deblurred with ADMM using the $L1$-norm deblurring term and ADMM using the $L2$-norm deblurring term. This analysis is provided in Figure 16. The full-scale images of the comparison in Figure 12 is also shown in Figure 17. Finally, a close-up comparison of the ADMM & Primal Douglas-Rachford algorithms is shown in Figure 22

Table 3: Effects of kernel choice on hyperparameters

Kernel	t	ρ	γ
Gaussian	0.5	1	$1e-15$
Motion	0.25	1	$1e-15$
Laplacian	$1e-11$	$1e-11$	$1e-15$
Disk	0.5	1	$1e-15$

Table 4: Effects of noise types on hyperparameters

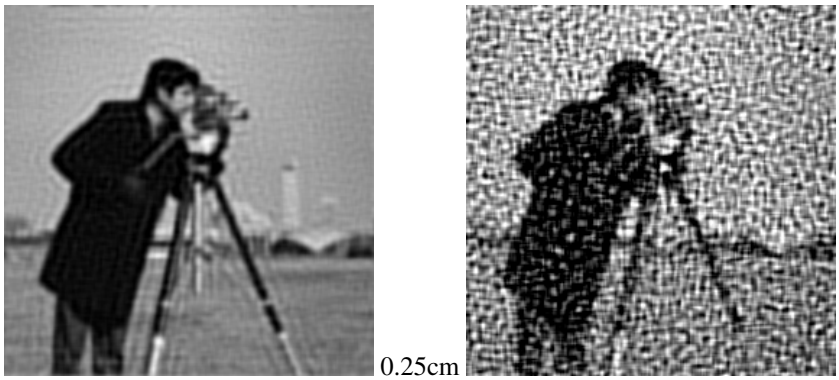
Noise Type	t	ρ	γ
Salt and Pepper	0.5	1	$1e-15$
Gaussian	$1e-11$	1	$1e-15$
Poisson	0.5	1	$1e-15$
Speckle	$1e-11$	1	$1e-15$

Table 5: Effects of noise density on hyperparameters

Noise Density	t	ρ	γ
0.001	0.5	1	$1e-15$
0.005	0.5	1	$1e-15$
0.01	0.25	1	$1e-15$
0.02	0.75	1	$1e-15$
0.03	0.75	1	$1e-15$
0.05	0.75	1	$1e-15$
0.07	0.5	1	$1e-15$
0.1	0.5	1	$1e-15$
0.5	0.5	1	$1e-15$

Table 6: Effects of noise types on hyperparameters

Prox-operator	t	ρ	γ
L1	0.5	1	$1e-15$
L2 squared	$1e-11$	1	$1e-15$



(a) $L1$-norm deblurring. (b) $L2$-norm deblurring.

Figure 16: Comparison of $L1$ versus $L2$ squared operator

As we can see in Fig. 16, the $L1$-norm encourages robustness to outliers whereas the $L2$-norm is much more prone to them. This is not surprising. It can be shown that minimizing the $L2$-norm of the error is equivalent to predicting the conditional expectation of the true image given the noisy input. However, $L1$-norm are typically resistant to outliers as they predict the posterior mode. Here is a sample (Bayesian) probabilistic justification for this [4].

(a) Motion (blurred) (b) Poisson (blurred) (c) Noise density 0.5 (blurred) (d) $L2$ prox operator (blurred)

(e) Motion (deblurred) (f) Poisson (deblurred) (g) Noise density 0.5 (deblurred) (h) $L2$ prox operator (deblurred)

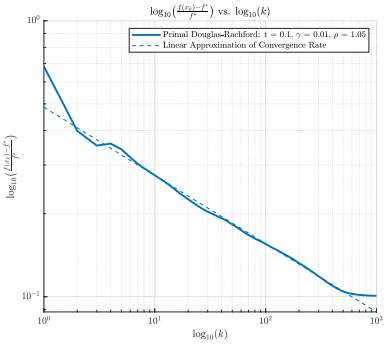
Figure 17: The blurred cameraman images using various kernel and noise types (densities) and resulting deblurred images obtained with the best ADMM hyperparameters after 1000 iterations.

A.2 Convergence Plots

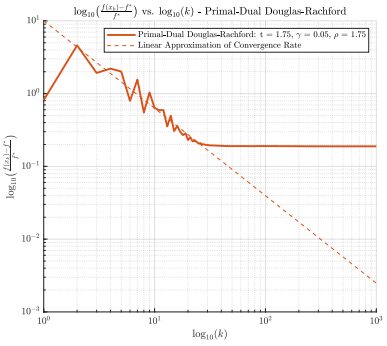
To compare the relative loss versus iteration plots with their linear approximations (used to determine the rate of convergence, κ, of each algorithm) each convergence plot was overlayed with their linear approximation in Figure 18. The resulting plots show the accuracy of the linear approximations.

A.3 Heuristic Algorithm Comparison

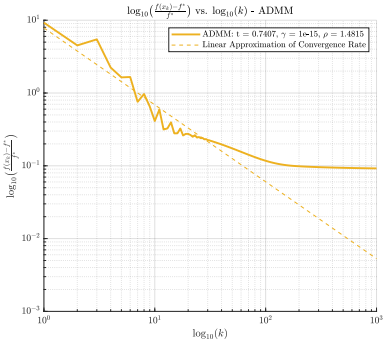
Here the algorithm performance on different images with the same blur and noise (9×9 gaussian blur of standard deviation 4) is shown. The full scale images from Figure 15 are also shown in Figure 19.



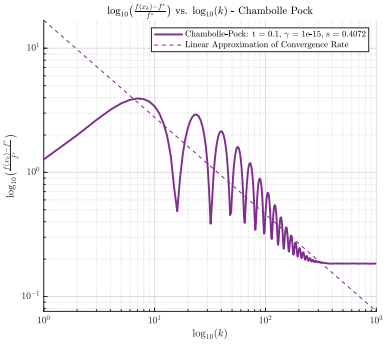
(a) The log-log plot of relative loss versus iterations (solid line) for the Primal Douglas-Rachfordalgorithm overlayed with its linear approximation (dashed line).



(b) The log-log plot of relative loss versus iterations (solid line) for the Primal-Dual Douglas-Rachfordalgorithm overlayed with its linear approximation (dashed line).



(c) The log-log plot of relative loss versus iterations (solid line) for the ADMM algorithm overlayed with its linear approximation (dashed line).



(d) The log-log plot of relative loss versus iterations (solid line) for the Chambolle-Pockalgorithm overlayed with its linear approximation (dashed line).

Figure 18: Comparison of log-log plots of relative loss versus iterations with their best-fit linear approximation for each algorithm.

(a) The cameraman image, without blur.

(b) The cameraman image with a 9×9 gaussian blur with standard deviation 4.

(c) The cameraman image deblurred with Primal Douglas-Rachford.

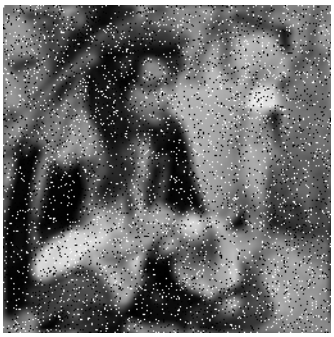
(d) The cameraman image deblurred with Primal-Dual Douglas-Rachford.

(e) The cameraman image deblurred with ADMM.

(f) The cameraman image deblurred with Chambolle-Pock.

Figure 19: The cameraman image before blurring (a), after blurring (b) and after deblurring with each algorithm (c)-(f). The image was blurred with a 9×9 gaussian blur of standard deviation 4 and salt & pepper noise with density 0.1 was applied. To deblur the image each algorithm was run for 1000 iterations using the best hyperparameters found in Section REFERENCE.

(a) The man with hat image, without blur.



(b) The man with hat image with a 9×9 gaussian blur with standard deviation 4.

(c) The man with hat image deblurred with Primal Douglas-Rachford.

(d) The man with hat image deblurred with Primal-Dual Douglas-Rachford.

(e) The man with hat image deblurred with ADMM.

(f) The man with hat image deblurred with Chambolle-Pock.

Figure 20: The man with hat image before blurring (a), after blurring (b) and after deblurring with each algorithm (c)-(f). The image was blurred with a 9×9 gaussian blur of standard deviation 4 and salt & pepper noise with density 0.1 was applied. To deblur the image each algorithm was run for 1000 iterations using the best hyperparameters found in Section REFERENCE.

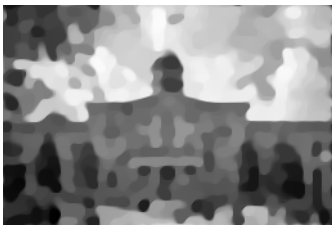
(a) The mcgill image, without blur.

(b) The mcgill image with a 9×9 gaussian blur with standard deviation 4.

(c) The mcgill image deblurred with Primal Douglas-Rachford.

(d) The mcgill image deblurred with Primal-Dual Douglas-Rachford.

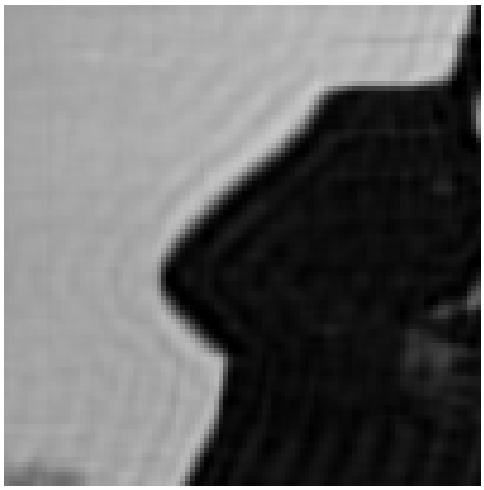
(e) The mcgill image deblurred with ADMM.



(f) The mcgill image deblurred with Chambolle-Pock.

Figure 21: The mcgill image before blurring (a), after blurring (b) and after deblurring with each algorithm (c)-(f). The image was blurred with a 9×9 gaussian blur of standard deviation 4 and salt & pepper noise with density 0.1 was applied. To deblur the image each algorithm was run for 1000 iterations using the best hyperparameters found in Section REFERENCE.

(a) A zoomed in view of the cameraman image, deblurred with Primal Douglas-Rachford.



(b) A zoomed in view of the cameraman image, deblurred with ADMM.

Figure 22: A zoomed in view of the deblurred cameraman image, obtained with Primal Douglas-Rachford(a) and ADMM (b).