Investigating Image Inpainting via the Alternating Direction Method of Multipliers

Jonathan Tuck
Stanford University
jonathantuck@stanford.edu

Abstract

In many imaging applications, there exists potential for corruption of the images by sources of structured noise that completely loses original pixel information. The reconstruction of the original image from its corrupted observation is known as image inpainting. This paper seeks to investigate image inpainting using a particular algorithm, the alternating direction method of multipliers (ADMM), and analyzes ADMM's performance in image inpainting. Due to the ill-posedness of image inpainting, four priors were investigated in the ADMM implementation: total variation, non-local means, BM3D, and the recursive Gaussian filter. For each prior investigated, this paper uses an open-source ADMM solver and compares two performance metrics, the PSNR and SSIM, for a variety of images and corruption models.

1. Introduction

In many imaging applications, there exists potential for corruption of the images by sources of structured noise that completely loses original pixel information. It is important to be able to reconstruct the image as accurately as possible, to be able to convey the original image information. This process of reconstructing images that have large portions of their image completely lost, known as image inpainting, allows for the estimation and restoration of those missing pixels. A simple example of image inpainting can be seen in Figure 1 [1]. As can be seen in Figure 1, the inpainting process may leave some artifacts behind and is not usually perfect; in fact, the accuracy of the inpainting is typically related to one's choice of prior (*i.e.*, a total variation prior will typically work better on an image with sparse gradients than on an image without sparse gradients.)

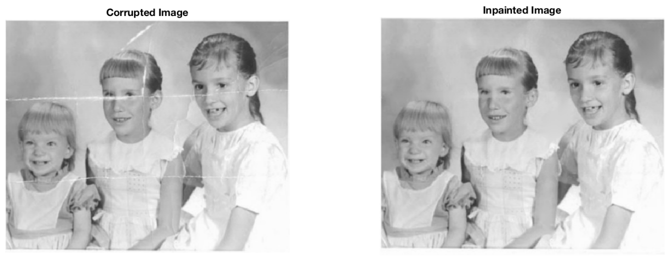

Figure 1. Two images of three children, one with image-degrading creases (left) and its inpainted counterpart (right.) Note the artifacts along the horizontal of the middle child's eyes, where inpainting occurred.

2. Related Work

2.1. ADMM

Recent work has been done in the field of efficient algorithms, and the alternating direction method of multipliers (ADMM) has been one of the more widespread algorithms used in the past few years [2]. In particular, ADMM solves the following optimization problem:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{subject to} & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c} \end{array} \tag{1}$$

where f and g are convex functions. The optimization problem is solved by forming the augmented lagrangian of the system,

$$L_\rho(\mathbf{x},\mathbf{z},\mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{z}-\mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{z}-\mathbf{c}\|_2^2, \tag{2}$$

where $\mathbf{y}$ is the dual variable of the problem, and performing the update procedure described in Algorithm 1 continuously until convergence. Here, prox() denotes the proximal operator [9]. As an aside, it is also common to replace $\mathbf{y}$ with the scaled dual variable $\mathbf{u} = (1/\rho)\mathbf{y}$ and perform ADMM with $\mathbf{u}$ being the update parameter.

```
Data: Functions $f, g$; Matrices $\mathbf{A}, \mathbf{B}$; Vector $\mathbf{c}$;
      convergence tolerance $\epsilon$
Result: Optimal value $\mathbf{x}$
while $\|\text{Residual}\|_2 = \|\mathbf{A}\mathbf{x}^k + \mathbf{B}\mathbf{z}^k - \mathbf{c}\|_2 > \epsilon$ do
    $\mathbf{x}^{k+1} := \underset{x}{\operatorname{argmin}}\, L_\rho(\mathbf{x}, \mathbf{z}^k, \mathbf{y}^k)$
        $= \underset{f,\rho}{\operatorname{prox}}(\mathbf{z}^k - (1/\rho)\mathbf{y}^k)$
        $= \underset{f,\rho}{\operatorname{prox}}(\mathbf{v})$
    $\mathbf{z}^{k+1} := \underset{z}{\operatorname{argmin}}\, L_\rho(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{y}^k)$
        $= \underset{g,\rho}{\operatorname{prox}}(\mathbf{A}\mathbf{x}^{k+1} + (1/\rho)\mathbf{y}^k)$
        $= \underset{g,\rho}{\operatorname{prox}}(\mathbf{v})$
    $\mathbf{y}^{k+1} := \mathbf{y}^k + \rho(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c})$.
end
```

Algorithm 1: ADMM algorithm.

One main benefit of ADMM is that convergence is guaranteed given that f and g are convex and L_0 (the normal lagrangian, or the augmented lagrangian with $\rho = 0$) has a saddle point. Another important benefit of ADMM is that because the algorithm requires little assumptions on the problem, ADMM naturally finds many applications in various spaces, such as in imaging and in deconvolution [13]. This lack of needed assumptions on the problem suggests that similar, or even the same, framework can be used for many seemingly unrelated problems. In addition, in an attempt to spread its wide uses, ADMM has been implemented in an open-source solver that produces relatively high-quality results [5].

ADMM has been used as a solution guaranteeing fixed-point convergence for many denoising algorithms. This past work suggests that using ADMM in an image inpainting framework would yield an accurate solution to the image restoration problem for many classes of images (*e.g.*, for images with sparse gradients.)

3. Methodology

3.1. Problem Formulation

In order to use ADMM as the algorithm to implement image inpainting, it is first required to formulate the image inpainting problem as an optimization problem in the form that ADMM solves.

Suppose there exists an uncorrputed image $\mathbf{x} \in \mathbb{R}^n$, where n is the number of pixels in the image. In the image inpainting scenario, we observe the corrupted image $\mathbf{y} = \mathbf{K}\mathbf{x}$, where $\mathbf{K} \in R^{n \times n}$ is a diagonal masking matrix. That is, $\mathbf{K}_{ii} = 1$ if the i^{th} pixel in $\mathbf{x}$ is observed, and is equal to 0 otherwise. In addition, assume that we make some prior assumption on $\mathbf{x}$ that is contained in the prior function Γ. Thus, the image inpainting problem can be formulated as:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \|\mathbf{K}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\Gamma(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{x} - \mathbf{z} = \mathbf{0}, \end{aligned} \tag{3}$$

where $\mathbf{z} \in \mathbb{R}^n$. Here, Problem (3) is equivalent to Problem (1) if we take $f(\mathbf{x}) = \|\mathbf{K}\mathbf{x} - \mathbf{b}\|_2^2$, $g(\mathbf{z}) = \Gamma(\mathbf{z})$, $\mathbf{A} = \mathbf{I}$, $\mathbf{B} = -\mathbf{I}$, and $\mathbf{c} = \mathbf{0}$. Also note that the prior function Γ can be thought of as a regularizer, and that the constraint that $\mathbf{x} - \mathbf{z} = \mathbf{0}$ implies that Γ is in fact a regularizer on $\mathbf{x}$. The constant λ can be thought of as a tradeoff parameter: picking larger values for λ will give the regularization more relative weight in the minimization, but will thus give less relative weight to the idea that $\|\mathbf{K}\mathbf{x} - \mathbf{b}\|_2^2$ should be small.

We can also derive an algorithm for ADMM like in Algorithm 1. We first note that for this problem, the augmented lagrangian can be expressed as

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = \|\mathbf{K}\mathbf{x} - \mathbf{b}\|_2^2 + \Gamma(\mathbf{z}) + \mathbf{y}^T(\mathbf{x} - \mathbf{z}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z}\|_2^2. \tag{4}$$

Then, the $\mathbf{x}$-update can be written as

$$\begin{aligned} \mathbf{x}^{k+1} &= \underset{x}{\operatorname{argmin}}\, L_\rho(\mathbf{x}, \mathbf{z}^{\mathbf{k}}, \mathbf{y}^{\mathbf{k}}) \\ &= \underset{f,\rho}{\operatorname{prox}}(\mathbf{v}), \quad \mathbf{v} = \mathbf{z}^k - (1/\rho)\mathbf{y}^k \\ &= \underset{x}{\operatorname{argmin}}\, \frac{1}{2}\|\mathbf{K}\mathbf{x} - \mathbf{b}\|_2^2 + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z}\|_2^2 \\ &= (\mathbf{K}^T\mathbf{K} + \rho\mathbf{I})^{-1}(\mathbf{K}^T\mathbf{b} + \rho\mathbf{v}). \end{aligned} \tag{5}$$

The $\mathbf{z}$-update can be derived, but requires the prior Γ. In particular,

$$\begin{aligned} \mathbf{z}^{k+1} &= \underset{\Gamma,\rho}{\operatorname{prox}}(\mathbf{v}) \\ &= \underset{z}{\operatorname{argmin}}\, L_\rho(\mathbf{x}^{\mathbf{k+1}}, \mathbf{z}, \mathbf{y}^{\mathbf{k}}) \\ &= \underset{z}{\operatorname{argmin}}\, \lambda\Gamma(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{v} - \mathbf{z}\|_2^2, \quad \mathbf{v} = \mathbf{x} + (1/\rho)\mathbf{y} \end{aligned} \tag{6}$$

The $\mathbf{y}$-update is trivial, and is expressed as

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \rho(\mathbf{x}^{k+1} - \mathbf{z}^{k+1}). \tag{7}$$

It is important to note that for the image inpainting scenario, the prior only affects the $\mathbf{z}$-update step.

3.2. Implementation

ADMM was implemented using the "Plug and Play" ADMM package for MATLAB [5]. This open-source implementation of ADMM is advantageous because it allows one to simply implement a relatively fast ADMM algorithm for a wide variety of purposes without requiring one to derive the update steps for each situation.

3.3. Priors

For this paper, four particular priors are investigated: the total variation (TV) prior, the non-local means (NLM) prior, the Block Matching and 3D Filtering (BM3D) prior, and the recursive Gaussian filter (RF) prior. These priors were chosen for their storied usefulness throughout image processing, particularly in denoising [3] [4] [6] [7] [10].

3.3.1 TV Prior

Explicitly, the TV prior is written as follows:

$$\Gamma(z) = \lambda \|Dz\|_1, \tag{8}$$

where $D = [D_x^T D_y^T]$ is the finite difference matrix for the horizontal and vertical image gradients [13]. In addition, the $\mathbf{z}$-update for image inpainting using ADMM with a TV-prior is

$$\begin{aligned} \mathbf{z}^{k+1} &:= \underset{\Gamma,\rho}{\mathrm{prox}}(\mathbf{v}) \\ &= S_{\lambda/\rho}(\mathbf{v}) \\ &= \begin{cases} \mathbf{v} - \lambda/\rho & \mathbf{v} > \lambda/\rho \\ 0 & |\mathbf{v}| \leq \lambda/\rho \\ \mathbf{v} + \lambda/\rho & \mathbf{v} < \lambda/\rho \end{cases}, \end{aligned} \tag{9}$$

where the function $S_{\lambda/\rho}$, also known as the soft-thresholding function, is applied element-wise [12].

Intuitively, the TV prior is used when one knows that the image being inpainted has sparse gradients.

3.3.2 NLM Prior

The NLM prior is a self similarity prior: that is, the NLM prior is used when one knows that the image being inpainted has subsections that look similar to other parts of the image. Unlike the TV prior, the NLM prior is implemented using an algorithm [4]. Given a corrupted image $\mathbf{v} = \{v(i) | i \in I\}$, the NLM algorithm, Algorithm 2 yields a weighted average $\Gamma(i)$. The notation $v(N_i)$ and $v(N_j)$ denotes the pixels in neighborhoods i and j, respectively; $\|\cdot\|_{2,a}$ denotes the weighted Gaussian norm with standard deviation a; and h is a constant picked to alter the degree of filtering [4].

```
Data: Noisy image $\mathbf{v} \in \mathbb{R}^n$, Gaussian standard
      deviation $a \in \mathbb{R}$, degree of filtering $h \in \mathbb{R}$
Result: Denoised image $\Gamma$
for Each pixel index $i \in \{1, \ldots, n\}$ do
    $Z(i) = \sum_j \exp \frac{\| v(N_i) - v(N_j) \|_{2,a}^2}{h^2}$
    $w(i,j) = \frac{1}{Z(i)} \exp \frac{\| v(N_i) - v(N_j) \|_{2,a}^2}{h^2}$
    $\Gamma(i) = \sum_{j \in I} w(i,j) v(j)$
end
```

Algorithm 2: NLM algorithm.

We can call the NLM algorithm like a function [12], and thus can put it in place of the $\mathbf{z}$-update:

$$\begin{aligned} \mathbf{z}^{k+1} &:= \underset{\Gamma,\rho}{\mathrm{prox}}(\mathbf{v}) \\ &= \mathrm{NLM}(\mathbf{v}, \lambda, \rho). \end{aligned} \tag{10}$$

3.3.3 BM3D Prior

Like the NLM prior, the BM3D prior is a self similarity prior. The assumption made when one uses a BM3D prior is that the 3D linear transformation of the image to be inpainted has a locally sparse structure. Also, like NLM, BM3D is implemented using an algorithm. The general steps in the algorithm can be summarized below [8]:

```
Data: Noisy image $X$, side length $N$
Result: Denoised image $\hat{X}$
for Each patch of side length $N$ do
    Find all similar patches in image and group
      them into a 3D block.
end
for each 3D block do
    Take the 3D linear transform of the block.
    Denoise the 3D linear transform of the block
      via hard thresholding.
    Take the inverse 3D linear transform of the
      denoised block transform.
end
Aggregate all of the blocks into the final
  denoised image.
```

Algorithm 3: BM3D algorithm.

Again, like in the NLM case, we can call the BM3D algorithm like a function, and thus can put it in place of the **z**-update:

$$\begin{aligned} \mathbf{z}^{k+1} &:= \underset{\Gamma,\rho}{\mathrm{prox}}(\mathbf{v}) \\ &= \mathrm{BM3D}(\mathbf{v}, \lambda, \rho). \end{aligned} \tag{11}$$

3.3.4 RF Prior

The recursive filter is a filter that essentially takes an image $X(t)$, transforms it into a new domain, filters that newly-transformed signal with a Gaussian filter, and inverts the transformation. In particular, the algorithm used to implement a recursive filter on an image $X(t)$ over each of its c channels can be summarized in Algorithm 4 [7].

```
Data: Noisy image $X$
Result: Denoised image $\hat{X}$
for each channel $i \in \{1, \ldots, c\}$ do
    1. Compute transform
       $$T(u) = \int_0^u \left(1 + \frac{\sigma_s}{\sigma_r} \sum_{k=1}^{c} \left|\frac{dX_k}{dt}\right|\right) dt,$$
       where $\sigma_s$ and $\sigma_r$ are the standard deviations of the space and range of the image domain, respectively.
    2. Transform $X(t)$ into $X(T(u))$.
    3. Filter $X(T(u))$ by a Gaussian filter $G(u)$ and invert back into original domain to obtain recursive filter estimate $\hat{X}(t)$. That is,
       $$\hat{X}(t) = T^{-1}\{X(T(u))G(u)\}.$$
end
```

Algorithm 4: RF algorithm.

Like in the NLM and BM3D cases, we can call the RF algorithm like a function, and thus can put it in place of the **z**-update:

$$\begin{aligned} \mathbf{z}^{k+1} &:= \underset{\Gamma,\rho}{\mathrm{prox}}(\mathbf{v}) \\ &= \mathrm{RF}(\mathbf{v}, \lambda, \rho). \end{aligned} \tag{12}$$

3.4. Test Images

In this paper, three test images are compared, each containing an important structure to test. The first picture tested is a picture of the Stanford logo, which contains sparse gradients and self-similarity. The second picture tested is a picture of artwork, which contains sparse gradients. The third picture tested is a picture of two basketball players, which contains a locally sparse transform. The three clean (uncorrupted) images can be seen below.

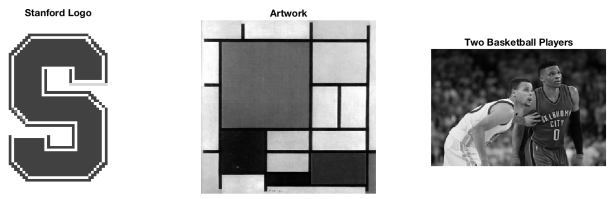

Figure 2. The three test images for the project.

3.5. Corruption Patterns

As image inpainting fundamentally seeks to restore images after corruption, it is imperative to test a variety of images with a particular pattern of corruption.

In particular, this project looks at four types of corruption masks: a random Gaussian mask, a bar mask, a small square, and a large square. These masks remove parts of the image in a structured manner (for reference, the black pixels denote masked pixels, and the white pixels denote uncorrupted pixels.) All four masks can be seen below.

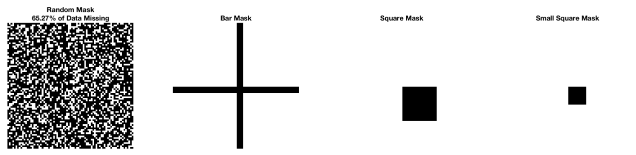

Figure 3. The four masks used in modeling the image inpainting scenario. The black colored pixels denote pixels that are removed from the original image.

All four of these masks have a basis in real-life corruption. The random mask models a noisy corruption that completely eliminates some pixels of the image. The bar mask models a corruption due to some real-world deformation of an image, much like the corruption seen in Figure 1. The square masks model complete data loss in chunks of one area of an image, but the varying square areas determine how much of the corruption can be inpainted.

3.6. Comparison Metrics

The two metrics that we use to quantitatively determine the performance of each prior in the image inpainting scenario include the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) [11].

3.6.1 PSNR

The PSNR is one of the more traditional ways to measure image quality. The PSNR measures the strength of a signal compared to its noise, and is typically measured in decibels. For the purposes of this paper, the PSNR between an image $Y \in \mathbb{R}^{m\times n}$ and its clean counterpart $X \in \mathbb{R}^{m\times n}$ is calculated as shown:

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{\max^2(X)}{\mathrm{MSE}}\right) \tag{13}$$

where the MSE (mean-square error) is:

$$\mathrm{MSE} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[X(i,j)-Y(i,j)]^2. \tag{14}$$

3.6.2 SSIM

Over the past decade, the SSIM has been used throughout image processing to quantify the similarity between two images [11]. The SSIM for two images, X and Y, can be explicitly calculated as:

$$\mathrm{SSIM}(X,Y) = \frac{(2\mu_X\mu_Y + C_1)(2\sigma_{XY}+C_2)}{(\mu_X^2+\mu_Y^2+C_1)(\sigma_X^2+\sigma_Y^2+C_2)}, \tag{15}$$

where μ_X is the average of X, μ_Y is the average of Y, σ_X is the variance of X, σ_Y is the variance of Y, σ_{XY} is the covariance of X and Y, and C_1, C_2 are normalization constants.

4. Results and Discussion

4.1. Resulting Images

Appendix A includes all images before corruption, after corruption, and after inpainting. The analysis of these images can be seen in the following subsections.

4.2. PSNR and SSIM Comparisons

Tables 3 through 10 from Appendix B include the PSNR and SSIM values for each of the resulting images after running ADMM with the respective prior.

It is not necessarily wise to quantitatively compare how well each prior performed for a given image over the four corruption masks, as each mask corrupts a different number of pixels and in different ways (*e.g.*, randomly vs. in a square). Fundamentally, this will lead to widely differing PSNRs and SSIMs for a particular image over the corruption masks. Of larger importance, then, is comparing how well each prior performed for a given mask over all of the test images.

In this regard, there are many insights to be found from looking at the tables in Appendix B. Looking at PSNR values, for the random mask, the BM3D prior performed the best over all three images. For the bar mask, the TV prior performed the best for the basketball player image, the NLM prior performed best for both the artwork image and Stanford logo image. For the larger square mask, the TV prior performed the best for the basketball player image, and the RF prior performed the best for both the artwork image and the Stanford logo image. For the smaller square image, the TV prior performed the best for the basketball player image, and the BM3D prior performed the best for both the artwork image and the Stanford logo image.

When looking at the SSIM values, all values coincide, except for a few differences: particularly, that the RF prior did not perform the best in any scenario, and that BM3D performed better than NLM for the artwork image. This result does make sense, as BM3D is commonly thought of as more effective than NLM.

This comparison data for both the PSNR and the SSIM values can be summarized in Tables 1 and 2.

Table 1. Best image inpainting priors for each mask when comparing PSNR.

PSNR	Random	Bars	Square	Small Square
Artwork	BM3D	NLM	RF	BM3D
Stanford Logo	BM3D	NLM	RF	BM3D
Basketball Players	BM3D/TV	TV	TV	TV

Table 2. Best image inpainting priors for each mask when comparing SSIM.

SSIM	Random	Bars	Square	Small Square
Artwork	BM3D	BM3D	BM3D	BM3D
Stanford Logo	BM3D	NLM	NLM	TV/ NLM/ BM3D
Basketball Players	BM3D / TV	TV	TV	TV

On the qualitative side, it is appropriate to compare how well each prior performed for a given image over the four corruption masks, as PSNR does not necessarily exactly correspond to a visually pleasing image.

First, to highlight some commonalities for all three of the images, the large square mask corrupted too much of the same area of image, causing none of the images with this corruption mask to reconstruct the masked area properly. The reconstructions with this corruption mask tended to properly reconstruct near the borders of the corruption mask, but left the area near the center of the corruption mask nearly unaltered. Most of the images did not have this problem for the smaller square mask.

For the images of the artwork, the BM3D prior worked qualitatively the best on the random corruption mask and the bar corruption mask. For the two square corruption masks, the RF prior worked the best. These results are unsurprising, as the artwork exhibits much self-similarity.

For the images of the Stanford logo, the NLM prior worked qualitatively the best on all four of the corruption masks. This intuitively makes sense, as by inspection, the image looks like it has many patches of self-similarity. It is also interesting to note how badly the TV performed in the inpainting process: there exists significant corruption in all of the reconstructions where TV was used. This is especially surprising, as the image looks incredibly sparse of gradients.

For the images of the two basketball players, the TV prior worked qualitatively the best on all four of the corruption masks. This result is unsurprising, as the image looks like it has extremely sparse gradients.

4.3. Choices of ρ and λ

For all of the images produced using this methodology, a value of $\rho = 1$ was found to work well. Increasing or decreasing $\rho = 1$ by small amounts rarely will change the final inpainted image, as ρ is associated with convergence rates.

The choice of λ has a significant effect on the quality of the inpainted image, however. For each test image, plots of both PSNR and SSIM vs. λ were generated (see Appendix C for all of these plots.) Based on these plots, it was decided that for the TV and NLM priors, $\lambda = 0.01$ would be used, and for the BM3D and RF priors, $\lambda = 0.005$ would be used.

5. Future Work

Future work for ADMM-based image inpainting is abundant. For example, the results of Tables 1 and 2 suggest that there exist be a relationship between the performance of image inpainting and the corruption model used, which is unintuitive as the prior in general is meant to model prior information on the *image* rather than the corruption. Potential future work might be to investigate whether the relationship between image inpainting performance and particular corruption models can be quantified.

For another example of potential future work, consider that as is clear from Section 4, for each image there exists a prior that quantitatively performs the best, and a value of λ that produces the best results. With the advent of machine learning and neural networks, it might be possible to implement a system where based on the structure of the image, an algorithm might be able to use images from a much larger dataset to determine the best prior and values of ρ and λ to use. Such a system would be able to even develop a hybrid prior, or a prior that combines previously-derived priors.

6. Conclusion

In conclusion, ADMM has been shown to be an efficient and accurate method to implement image inpainting. In addition, it has been shown that the prior chosen significantly affects the performance of ADMM, and that changing the corruption mask also affects which prior performs best. Finally, the choice of the convergence parameter ρ and the prior tradeoff parameter λ significantly affect the performance of the reconstruction using ADMM.

References

[1] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 417–424, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, Jan. 2011.

[3] A. Buades, B. Coll, and J. M. Morel. On image denoising methods. Technical report, Technical Note, CMLA (Centre de Mathematiques et de Leurs Applications, 2004.

[4] A. Buades, B. Coll, and J. M. Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65 vol. 2, June 2005.

[5] S. H. Chan, X. Wang, and O. A. Elgendy. Plug-and-play ADMM for image restoration: Fixed point convergence and applications. *CoRR*, abs/1605.01710, 2016.

[6] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising with block-matching and 3d filtering, 2006.

[7] E. S. L. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, pages 69:1–69:12, New York, NY, USA, 2011. ACM.

[8] M. Lebrun. An Analysis and Implementation of the BM3D Image Denoising Method. *Image Processing On Line*, 2:175–213, 2012.

[9] N. Parikh and S. Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, Jan. 2014.

[10] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, Nov. 1992.

[11] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[12] G. Wetzstein. EE 367 / CS 448I Computational Imaging and Display (Winter 2017) Notes: Compressive Imaging and Regularized Image Reconstruction (lecture 11), January 2017.

[13] G. Wetzstein. EE 367 / CS 448I Computational Imaging and Display (Winter 2017) Notes: Image Deconvolution (lecture 6), January 2017.

A. Inpainted Images

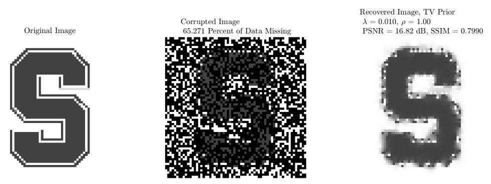
Original Image
Corrupted Image
65.271 Percent of Data Missing
Recovered Image, TV Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 16.82 dB, SSIM = 0.7990

Figure 4. Original image, corrupted image with random mask, and inpainted image of Stanford logo with TV Prior.

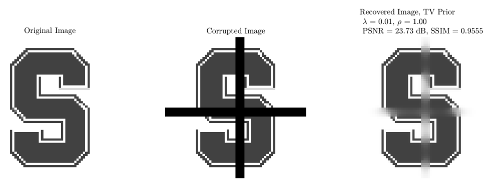
Original Image
Corrupted Image
Recovered Image, TV Prior
$\lambda = 0.01$, $\rho = 1.00$
PSNR = 23.73 dB, SSIM = 0.9555

Figure 5. Original image, corrupted image with bar mask, and inpainted image of Stanford logo with TV Prior.

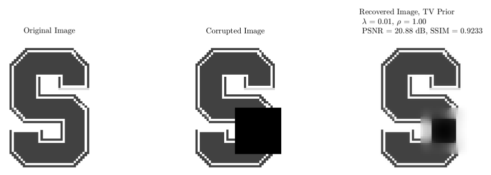
Original Image
Corrupted Image
Recovered Image, TV Prior
$\lambda = 0.01$, $\rho = 1.00$
PSNR = 20.88 dB, SSIM = 0.9233

Figure 6. Original image, corrupted image with large square mask, and inpainted image of Stanford logo with TV Prior.

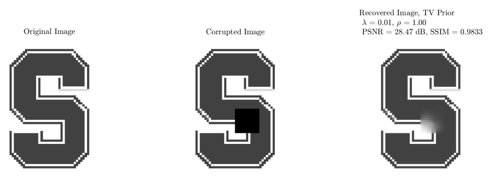
Original Image
Corrupted Image
Recovered Image, TV Prior
$\lambda = 0.01$, $\rho = 1.00$
PSNR = 28.47 dB, SSIM = 0.9833

Figure 7. Original image, corrupted image with small square mask, and inpainted image of Stanford logo with TV Prior.

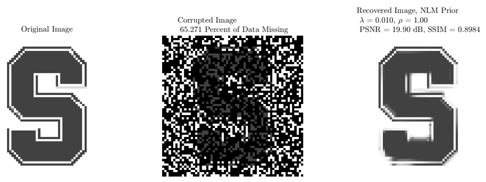
Original Image
Corrupted Image
65.271 Percent of Data Missing
Recovered Image, NLM Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 19.90 dB, SSIM = 0.8984

Figure 8. Original image, corrupted image with random mask, and inpainted image of Stanford logo with NLM Prior.

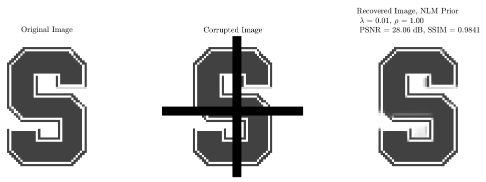
Original Image
Corrupted Image
Recovered Image, NLM Prior
$\lambda = 0.01$, $\rho = 1.00$
PSNR = 28.06 dB, SSIM = 0.9841

Figure 9. Original image, corrupted image with bar mask, and inpainted image of Stanford logo with NLM Prior.

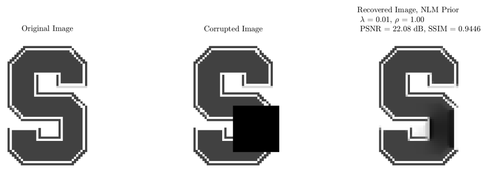
Original Image
Corrupted Image
Recovered Image, NLM Prior
$\lambda = 0.01$, $\rho = 1.00$
PSNR = 22.08 dB, SSIM = 0.9446

Figure 10. Original image, corrupted image with large square mask, and inpainted image of Stanford logo with NLM Prior.

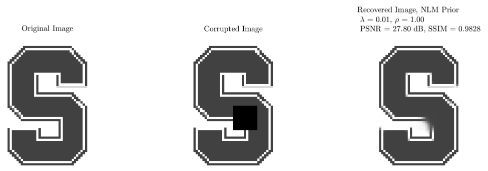
Original Image
Corrupted Image
Recovered Image, NLM Prior
$\lambda = 0.01$, $\rho = 1.00$
PSNR = 27.80 dB, SSIM = 0.9828

Figure 11. Original image, corrupted image with small square mask, and inpainted image of Stanford logo with NLM Prior.

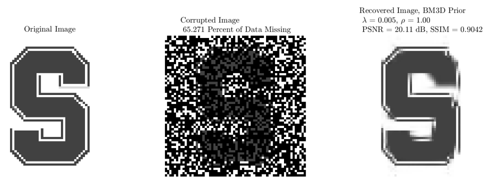
Original Image
Corrupted Image
65.271 Percent of Data Missing
Recovered Image, BM3D Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 20.11 dB, SSIM = 0.9042

Figure 12. Original image, corrupted image with random mask, and inpainted image of Stanford logo with BM3D Prior.

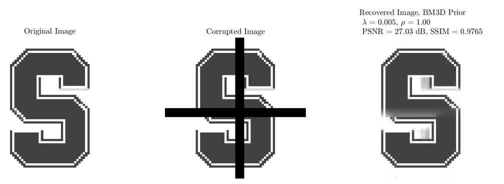
Original Image
Corrupted Image
Recovered Image, BM3D Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 27.03 dB, SSIM = 0.9765

Figure 13. Original image, corrupted image with bar mask, and inpainted image of Stanford logo with BM3D Prior.

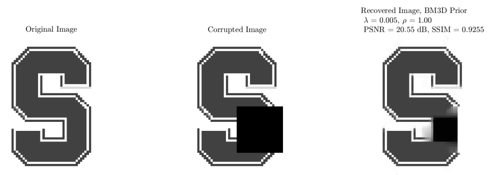
Original Image
Corrupted Image
Recovered Image, BM3D Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 20.55 dB, SSIM = 0.9255

Figure 14. Original image, corrupted image with large square mask, and inpainted image of Stanford logo with BM3D Prior.

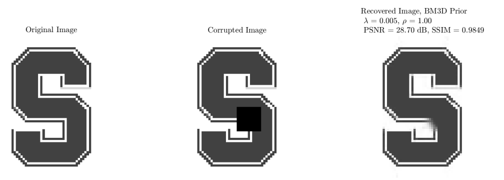
Original Image
Corrupted Image
Recovered Image, BM3D Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 28.70 dB, SSIM = 0.9849

Figure 15. Original image, corrupted image with small square mask, and inpainted image of Stanford logo with BM3D Prior.

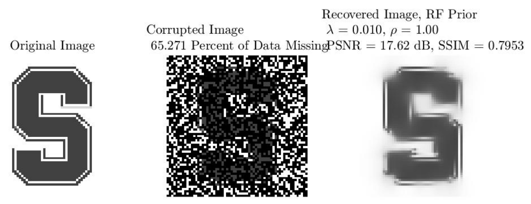
Original Image
Corrupted Image
65.271 Percent of Data Missing
Recovered Image, RF Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 17.62 dB, SSIM = 0.7953

Figure 16. Original image, corrupted image with random mask, and inpainted image of Stanford logo with RF Prior.

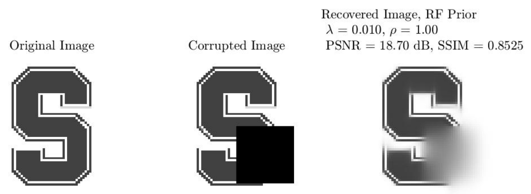
Original Image
Corrupted Image
Recovered Image, RF Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 18.70 dB, SSIM = 0.8525

Figure 18. Original image, corrupted image with large square mask, and inpainted image of Stanford logo with RF Prior.

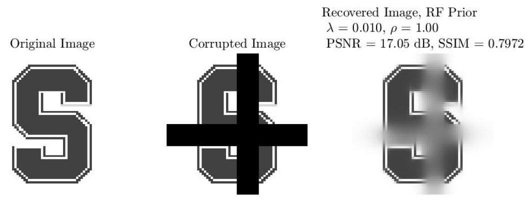
Original Image
Corrupted Image
Recovered Image, RF Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 17.05 dB, SSIM = 0.7972

Figure 17. Original image, corrupted image with bar mask, and inpainted image of Stanford logo with RF Prior.

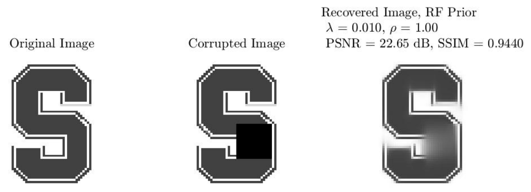
Original Image
Corrupted Image
Recovered Image, RF Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 22.65 dB, SSIM = 0.9440

Figure 19. Original image, corrupted image with small square mask, and inpainted image of Stanford logo with RF Prior.

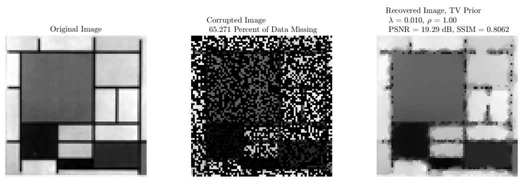
Original Image
Corrupted Image
65.271 Percent of Data Missing
Recovered Image, TV Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 19.29 dB, SSIM = 0.8062

Figure 20. Original image, corrupted image with random mask, and inpainted image of art with TV Prior.

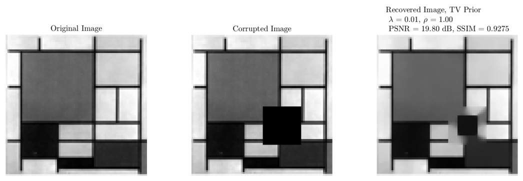
Original Image
Corrupted Image
Recovered Image, TV Prior
$\lambda = 0.01$, $\rho = 1.00$
PSNR = 19.80 dB, SSIM = 0.9275

Figure 22. Original image, corrupted image with large square mask, and inpainted image of art with TV Prior.

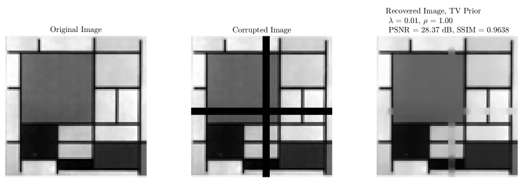
Original Image
Corrupted Image
Recovered Image, TV Prior
$\lambda = 0.01$, $\rho = 1.00$
PSNR = 28.37 dB, SSIM = 0.9638

Figure 21. Original image, corrupted image with bar mask, and inpainted image of art with TV Prior.

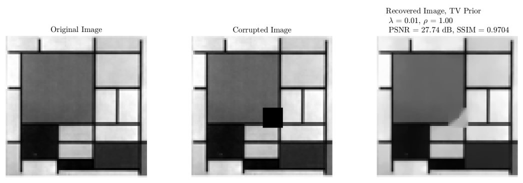
Original Image
Corrupted Image
Recovered Image, TV Prior
$\lambda = 0.01$, $\rho = 1.00$
PSNR = 27.74 dB, SSIM = 0.9704

Figure 23. Original image, corrupted image with small square mask, and inpainted image of art with TV Prior.

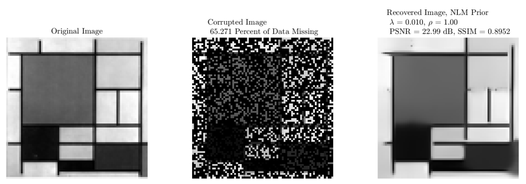
Original Image
Corrupted Image
65.271 Percent of Data Missing
Recovered Image, NLM Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 22.99 dB, SSIM = 0.8952

Figure 24. Original image, corrupted image with random mask, and inpainted image of art with NLM Prior.

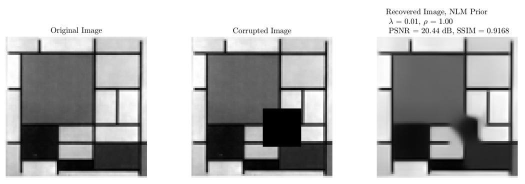
Original Image
Corrupted Image
Recovered Image, NLM Prior
$\lambda = 0.01$, $\rho = 1.00$
PSNR = 20.44 dB, SSIM = 0.9168

Figure 26. Original image, corrupted image with large square mask, and inpainted image of art with NLM Prior.

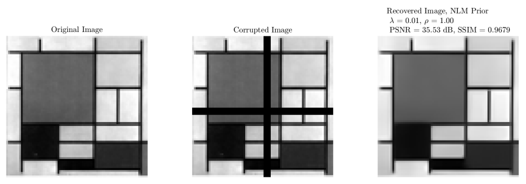
Original Image
Corrupted Image
Recovered Image, NLM Prior
$\lambda = 0.01$, $\rho = 1.00$
PSNR = 35.53 dB, SSIM = 0.9679

Figure 25. Original image, corrupted image with bar mask, and inpainted image of art with NLM Prior.

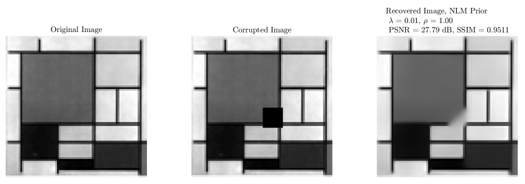
Original Image
Corrupted Image
Recovered Image, NLM Prior
$\lambda = 0.01$, $\rho = 1.00$
PSNR = 27.79 dB, SSIM = 0.9511

Figure 27. Original image, corrupted image with small square mask, and inpainted image of art with NLM Prior.

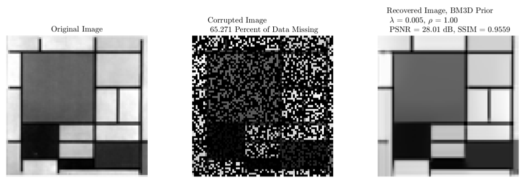
Original Image
Corrupted Image
65.271 Percent of Data Missing
Recovered Image, BM3D Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 28.01 dB, SSIM = 0.9559

Figure 28. Original image, corrupted image with random mask, and inpainted image of art with BM3D Prior.

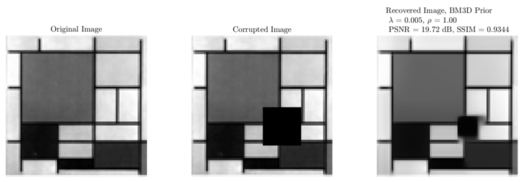
Original Image
Corrupted Image
Recovered Image, BM3D Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 19.72 dB, SSIM = 0.9344

Figure 30. Original image, corrupted image with large square mask, and inpainted image of art with BM3D Prior.

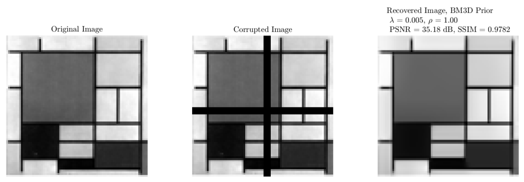
Original Image
Corrupted Image
Recovered Image, BM3D Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 35.18 dB, SSIM = 0.9782

Figure 29. Original image, corrupted image with bar mask, and inpainted image of art with BM3D Prior.

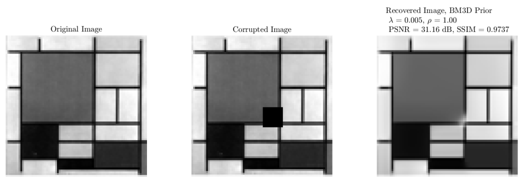
Original Image
Corrupted Image
Recovered Image, BM3D Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 31.16 dB, SSIM = 0.9737

Figure 31. Original image, corrupted image with small square mask, and inpainted image of art with BM3D Prior.

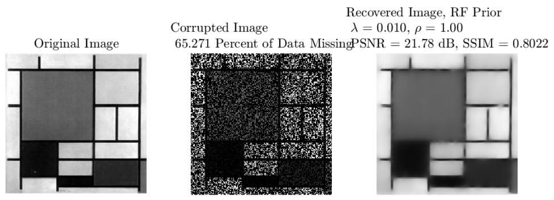
Original Image
Corrupted Image
65.271 Percent of Data Missing
Recovered Image, RF Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 21.78 dB, SSIM = 0.8022

Figure 32. Original image, corrupted image with random mask, and inpainted image of art with RF Prior.

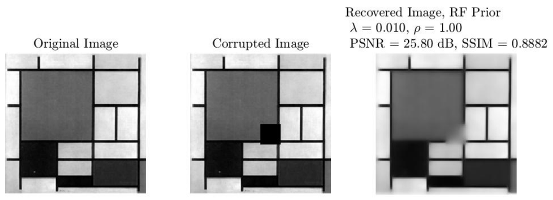
Original Image
Corrupted Image
Recovered Image, RF Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 25.80 dB, SSIM = 0.8882

Figure 34. Original image, corrupted image with large square mask, and inpainted image of art with RF Prior.

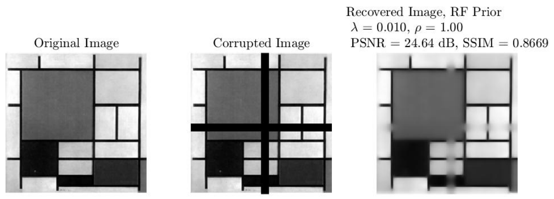
Original Image
Corrupted Image
Recovered Image, RF Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 24.64 dB, SSIM = 0.8669

Figure 33. Original image, corrupted image with bar mask, and inpainted image of art with RF Prior.

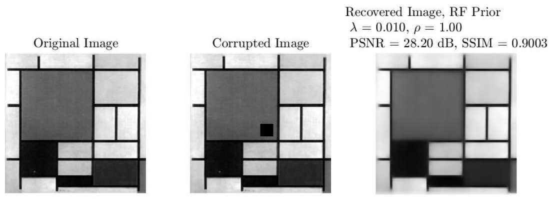
Original Image
Corrupted Image
Recovered Image, RF Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 28.20 dB, SSIM = 0.9003

Figure 35. Original image, corrupted image with small square mask, and inpainted image of art with RF Prior.

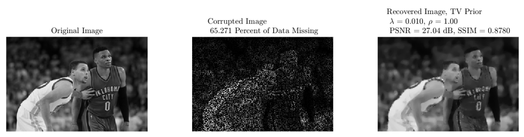
Original Image
Corrupted Image
65.271 Percent of Data Missing
Recovered Image, TV Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 27.04 dB, SSIM = 0.8780

Figure 36. Original image, corrupted image with random mask, and inpainted image of two basketball players with TV Prior.

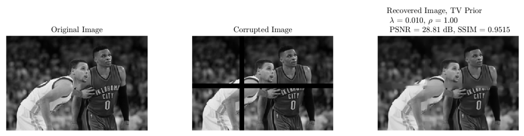
Original Image
Corrupted Image
Recovered Image, TV Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 28.81 dB, SSIM = 0.9515

Figure 37. Original image, corrupted image with bar mask, and inpainted image of two basketball players with TV Prior.

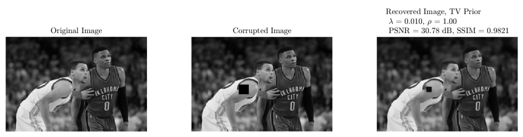
Original Image
Corrupted Image
Recovered Image, TV Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 30.78 dB, SSIM = 0.9821

Figure 38. Original image, corrupted image with large square mask, and inpainted image of two basketball players with TV Prior.

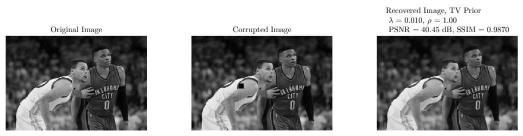
Original Image
Corrupted Image
Recovered Image, TV Prior
$\lambda = 0.010$, $\rho = 1.00$
PSNR = 40.45 dB, SSIM = 0.9870

Figure 39. Original image, corrupted image with small square mask, and inpainted image of two basketball players with TV Prior.

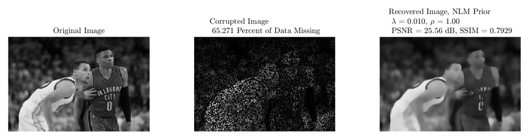
Original Image
Corrupted Image
65.271 Percent of Data Missing
Recovered Image, NLM Prior
λ = 0.010, ρ = 1.00
PSNR = 25.56 dB, SSIM = 0.7929

Figure 40. Original image, corrupted image with random mask, and inpainted image of two basketball players with NLM Prior.

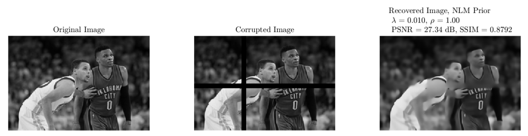

Figure 41. Original image, corrupted image with bar mask, and inpainted image of two basketball players with NLM Prior.

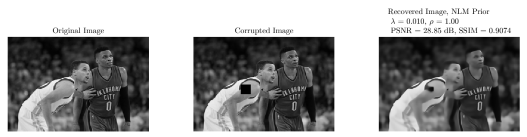

Figure 42. Original image, corrupted image with large square mask, and inpainted image of two basketball players with NLM Prior.

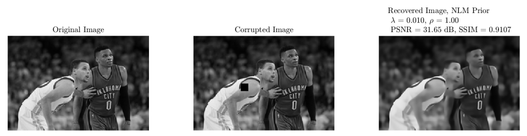

Figure 43. Original image, corrupted image with small square mask, and inpainted image of two basketball players with NLM Prior.

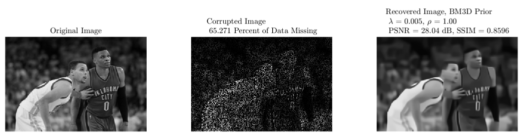
Original Image
Corrupted Image
65.271 Percent of Data Missing
Recovered Image, BM3D Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 28.04 dB, SSIM = 0.8596

Figure 44. Original image, corrupted image with random mask, and inpainted image of two basketball players with BM3D Prior.

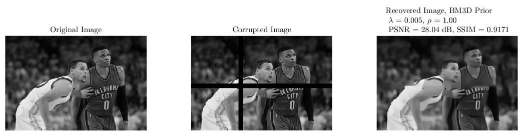
Original Image
Corrupted Image
Recovered Image, BM3D Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 28.04 dB, SSIM = 0.9171

Figure 45. Original image, corrupted image with bar mask, and inpainted image of two basketball players with BM3D Prior.

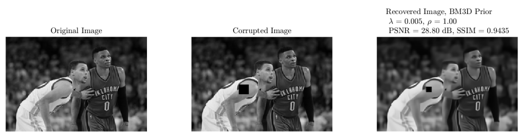
Original Image
Corrupted Image
Recovered Image, BM3D Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 28.80 dB, SSIM = 0.9435

Figure 46. Original image, corrupted image with large square mask, and inpainted image of two basketball players with BM3D Prior.

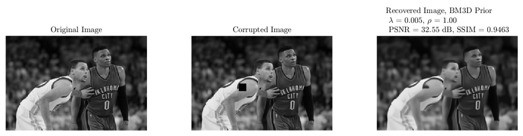
Original Image
Corrupted Image
Recovered Image, BM3D Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 32.55 dB, SSIM = 0.9463

Figure 47. Original image, corrupted image with small square mask, and inpainted image of two basketball players with BM3D Prior.

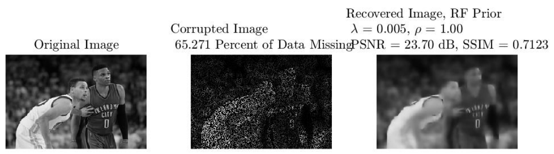
Original Image
Corrupted Image
65.271 Percent of Data Missing
Recovered Image, RF Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 23.70 dB, SSIM = 0.7123

Figure 48. Original image, corrupted image with random mask, and inpainted image of two basketball players with RF Prior.

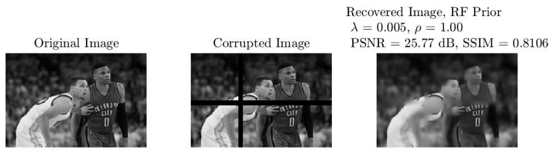
Original Image
Corrupted Image
Recovered Image, RF Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 25.77 dB, SSIM = 0.8106

Figure 49. Original image, corrupted image with bar mask, and inpainted image of two basketball players with RF Prior.

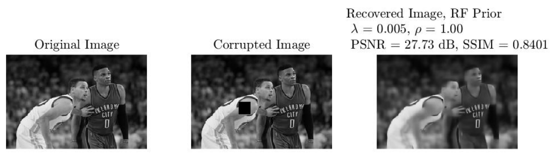
Original Image
Corrupted Image
Recovered Image, RF Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 27.73 dB, SSIM = 0.8401

Figure 50. Original image, corrupted image with large square mask, and inpainted image of two basketball players with RF Prior.

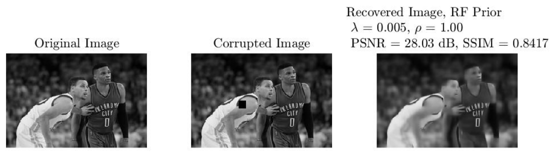
Original Image
Corrupted Image
Recovered Image, RF Prior
$\lambda = 0.005$, $\rho = 1.00$
PSNR = 28.03 dB, SSIM = 0.8417

Figure 51. Original image, corrupted image with small square mask, and inpainted image of two basketball players with RF Prior.

B. Noise Measurement Tables

Table 3. PSNR Values, TV Prior

PSNR [dB]	Random	Bars	Square	Small Square
Artwork	19.29	28.37	19.80	27.74
Stanford Logo	16.82	23.73	20.88	28.47
Basketball Players	27.04	28.81	30.78	40.45

Table 4. PSNR Values, NLM Prior

PSNR [dB]	Random	Bars	Square	Small Square
Artwork	22.99	35.53	20.44	27.79
Stanford Logo	19.90	28.06	22.08	27.80
Basketball Players	25.56	27.34	28.85	31.65

Table 5. PSNR Values, BM3D Prior

PSNR [dB]	Random	Bars	Square	Small Square
Artwork	28.01	35.18	19.72	31.16
Stanford Logo	20.11	27.03	20.55	28.70
Basketball Players	28.04	28.04	28.80	32.55

Table 6. PSNR Values, RF Prior

PSNR [dB]	Random	Bars	Square	Small Square
Artwork	21.78	24.64	25.80	28.20
Stanford Logo	17.62	17.05	22.65	18.70
Basketball Players	23.70	25.77	27.73	28.03

Table 7. SSIM Values, TV Prior

SSIM	Random	Bars	Square	Small Square
Artwork	.8062	.9638	.9275	.9704
Stanford Logo	.7990	.9555	.9233	0.9833
Basketball Players	.8780	.9515	.9821	.9870

Table 8. SSIM Values, NLM Prior

SSIM	Random	Bars	Square	Small Square
Artwork	.8952	.9679	.9168	.9511
Stanford Logo	.8984	.9841	.9446	.9828
Basketball Players	.7929	.8792	.9074	.9107

Table 9. SSIM Values, BM3D Prior

SSIM	Random	Bars	Square	Small Square
Artwork	.9559	.9782	.9344	.9737
Stanford Logo	.9042	.9765	.9255	.9849
Basketball Players	.8596	.9171	.9435	.9463

Table 10. SSIM Values, RF Prior

SSIM	Random	Bars	Square	Small Square
Artwork	.8082	.8669	.8882	.9003
Stanford Logo	.7953	.7972	.8525	.9440
Basketball Players	.7123	.8106	.8401	.8417

C. Graphs of PSNR and SSIM vs. λ

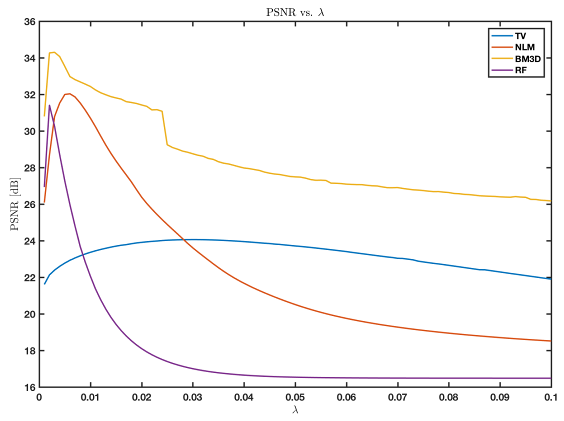

Figure 52. Plot of PSNR vs. λ for test image of artwork.

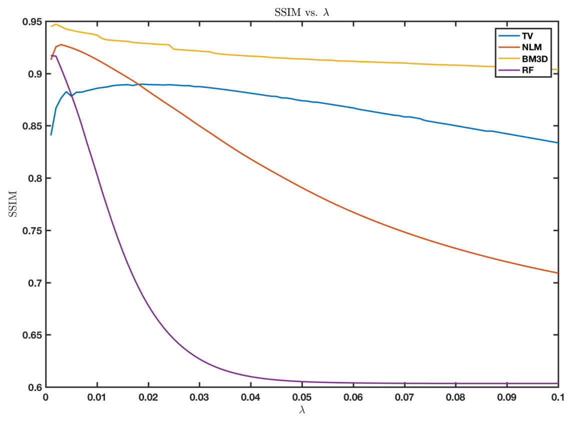

Figure 53. Plot of SSIM vs. λ for test image of artwork.

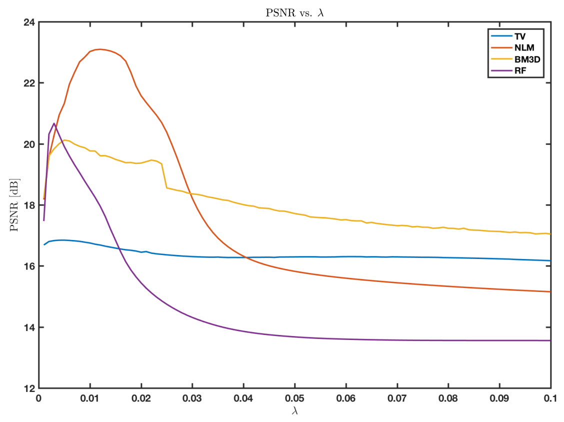

Figure 54. Plot of PSNR vs. λ for test image of Stanford logo.

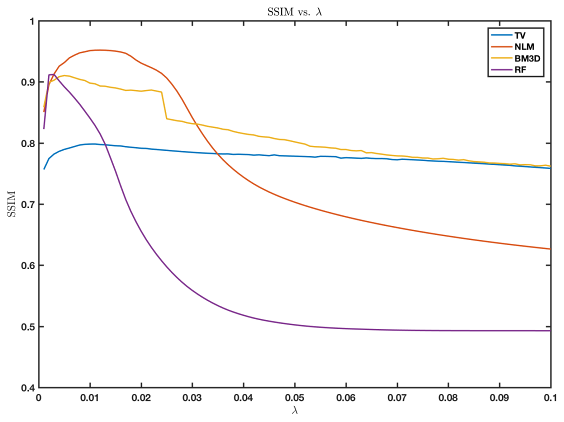

Figure 55. Plot of SSIM vs. λ for test image of Stanford logo.

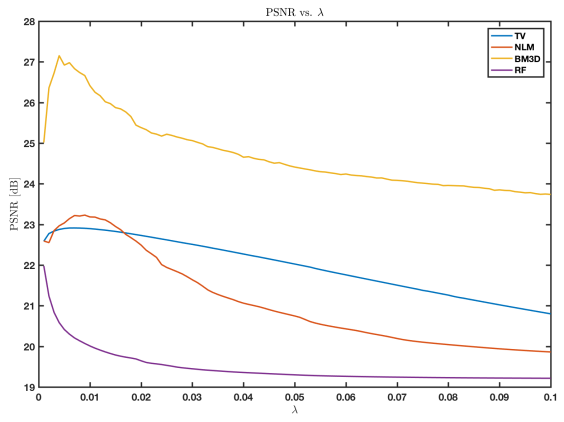

Figure 56. Plot of PSNR vs. λ for test image of Stanford logo.

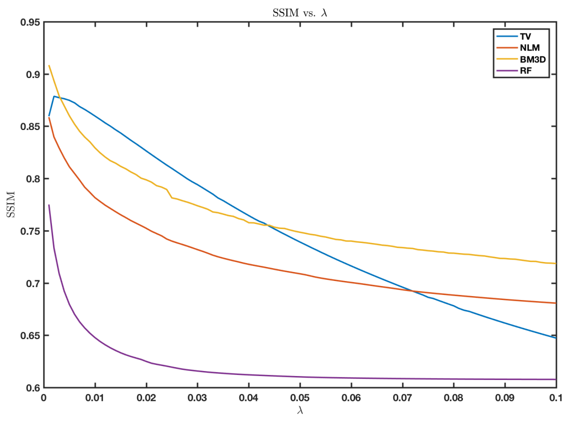

Figure 57. Plot of SSIM vs. λ for test image of Stanford logo.