DUKE UNIVERSITY

JAN 2024

Compressed Sensing Image Reconstruction

Arnav Nayak
ECE 580: Introduction to Machine Learning

Dr. Stacy Tantum

Overview

Project Description

Background

Image Reconstruction - Why it Matters?

The ability of computers to produce high-quality signal reconstructions from noisy, corrupted, or incomplete data sources marks a significant breakthrough in bridging the gap between humans and digital information. In a world increasingly driven by data, the accurate reconstruction of images is essential for **unlocking new dimensions of insights** and **improving decision-making** across a wide array of applications. These applications include medical imaging, security surveillance, digital forensics, and content creation. Take medical imaging as an example: image reconstruction enables the generation of high-quality images for clinical purposes. Deep-learning based image reconstruction methods are being used to enhance imaging performance and improve diagnosis.[1] Overall, image reconstruction technology has expanded the limits of what is visually interpretable, **enriching our interaction with the digital world**.

[1]Yaqub M, Jinchao F, Arshid K, Ahmed S, Zhang W, Nawaz MZ, Mahmood T. Deep Learning-Based Image Reconstruction for Different Medical Imaging Modalities. Comput Math Methods Med. 2022 Jun 16;2022:8750648. doi: 10.1155/2022/8750648. PMID: 35756423; PMCID: PMC9225884.

Background

Discrete Cosine Transform (DCT) Coefficients

The success of any predictive model greatly depends on the inputs being used to train. When developing models, it's important that we choose the correct set of features that characterizes our dataset properly and maintains its dimensionality. Performing quality **feature engineering** will ensure that our model accurately learns underlying patterns within the dataset and generalize better to unseen data.

For an image, utilizing the raw pixel value of an image as input to our model introduces unnecessary complexity and introduces noise that we aim to eliminate. Instead, we find a cleaner representation of our image through the **Discrete Cosine Transform (DCT)**. The DCT allows us to represent an image as a weighted sum of sinusoids of varying magnitude and frequencies, where the weights are the DCT coefficients. Mathematically, the Discrete Cosine Transform is a technique that transforms spatial domain data into frequency domain data, effectively concentrating most of the signal's energy into a few low-frequency components. In fact, JPEG image compression is centered around its use of DCT coefficients to represent the image.[2] But for the purposes of this project, DCT coefficients are a set of numbers that best **characterize an image**, and will be used as the **weights** for our models.

[2]The MathWorks Inc. (2024). Discrete Cosine Transform Documentation, Natick, Massachusetts: The MathWorks Inc. https://www.mathworks.com/help/images/discrete-cosine-transform.html

Background

Image Reconstruction - Compressed Sensing

A common goal of signal processing is to reconstruct a signal from a series of sampling measurements. This task may seem impossible at first because there is no way to reconstruct the signal during the times that the signal is not measured. However, with prior assumptions about the signal, it turns out to be possible to perfectly reconstruct a signal from a series of measurements. One of these assumptions is **sparsity** - which requires our signal to be sparse in some domain. Enforcing the assumption of sparsity in our signal in order to reconstruct it is a process known as **compressed sensing**.[3]

Why does Image Reconstruction work well with Compressed Sensing?

In the context of image reconstruction, we know that our complete image can be represented by a weighted sum of sinusoids, with the DCT coefficients as the weights. Since compressed sensing relies on sparsity in our signal, our models will need to rely on the assumption that the DCT coefficients of the image are sparse. Now, this assumption is unrealistic for the entire image as the image most likely contains a spectrum of frequency content that results in many non-zero coefficients. However, if we break our image into **small image blocks** (chips), then we can assume that each chip will have only select frequencies within it, resulting in **sparse DCT coefficients**. As a result, we can effectively use compressed sensing to reconstruct our image block by block.

[3]F.H.P. Fitzek, F. Granelli, P. Seeling (Eds.), Computing in Communication Networks, Academic Press (2020), pp. 197-215, https://doi.org/10.1016/B978-0-12-820488-7.00023-2

Project Goals

Compressed Sensing Image Reconstruction - Objectives

The goal of this project is to delve into image reconstructing by creating a predictive model to predict the values of missing pixels. Specifically, we want to

- Create a **basis function representation** of an image chip by rasterizing 2D DCT basis functions into a basis vector matrix
- Use **supervised learning** to train a **regression model** using the basis functions present in the chip as input features and the pixels present in the image as outputs.
- Use **L1-Regularization (Lasso)** to enforce a sparsity constraint on the undetermined linear system found while training the regression model.
- Estimate the value of missing pixels of the chip by using the missing basis functions as input to our model, whose weights now approximate the **DCT coefficients** of the complete chip.
- Optimize the selection of regularization parameter through **cross validation** with random subsets.
- Evaluate the impact of **median filtering** on reconstruction quality by running simulations with our model architecture.

Project Goals

Compressed Sensing Image Reconstruction - Subject Images and Other tools

The **Fishing Boat** (*Figure 1*) and **Nature** (*Figure 2*) grayscale images shown to the right are used in the simulations to test our model's validity and performance. These images contain a variety of **spatial frequency** and **textures**, which allows us to test our model against a variety of image characteristics.

The **field test** image (*Figure 3*) is a corrupted grayscale image with about **65%** of the pixels missing. Once we've implemented our LASSO reconstruction algorithm, we will apply it to this image to estimate its missing pixels.

We will be using Python to create our models. We will be using *scikit-learn*[4] to assist our implementations of the regression model and the Lasso regularization. We will also make use of the tools provided to us by *numpy*[5] and *matplotlib*[6] libraries for computation and visualization.

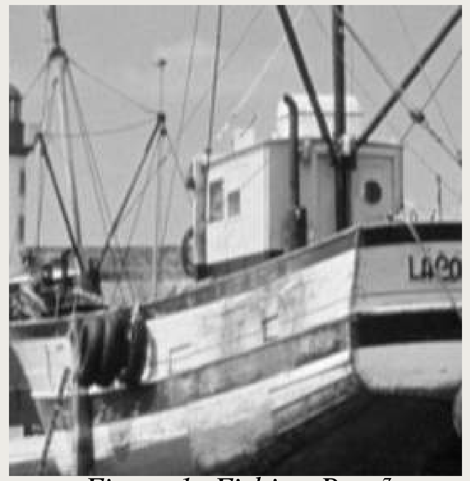


Figure 1: Fishing Boat[7]

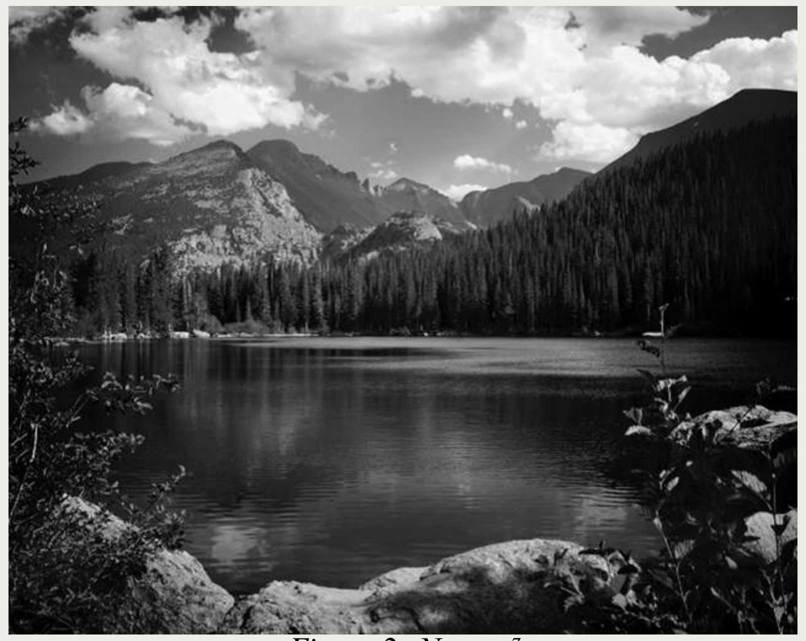


Figure 2: Nature[7]

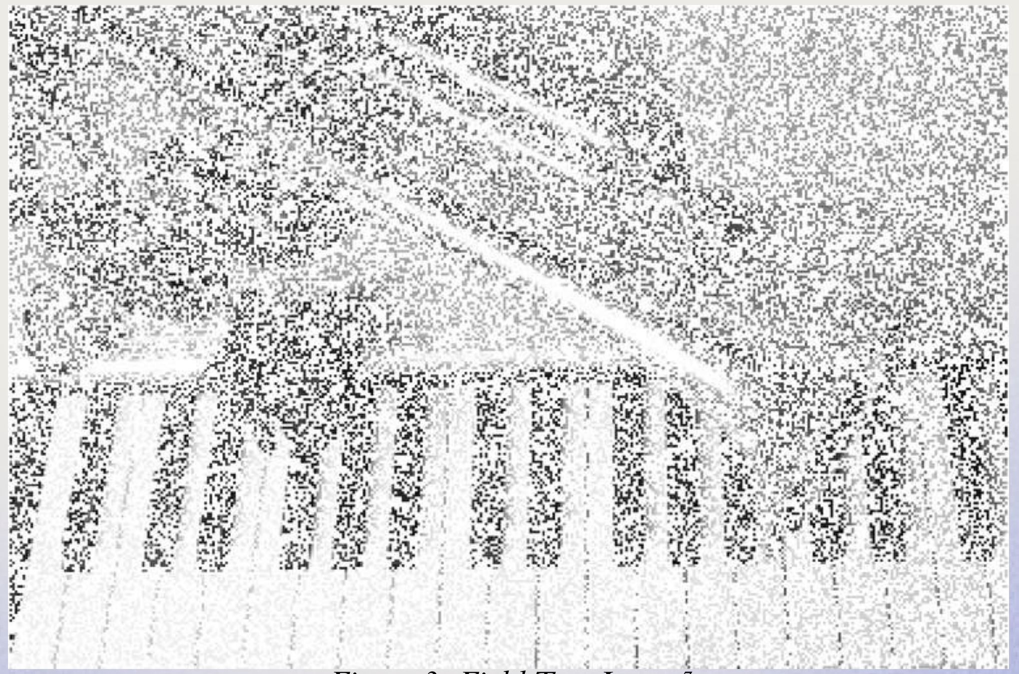


Figure 3: Field Test Image[7]

[4]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[5]Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.
[6]J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.
[7]Tantum, Stacy. "Report Guidance: Compressed Sensing Image Recovery" Duke University. Feb 2024.

Project Goals

Note - Other Applications

The modeling framework outlined can be used for a vast number of applications other than image reconstruction. One specific scenario is **astronomical signal denoising** in space exploration. In astronomy, astronomical images and other signals are often contaminated with noise when they reach Earth. If we can reconstruct corrupted or noisy parts of an incoming astronomical signal, we can enable **higher quality imaging** and improve the **accuracy of celestial observations**.[8]

We can use a similar modeling approach to tackle this challenge. We can treat the astronomical signal from a telescope as a signal that, in some domain, exhibits **sparsity**. By applying the **compressed sensing** framework, we can represent our signal in terms of a **basis function representation** and train a model on the available data with **regularized regression**. When a new set of telescope data is acquired, the model can be used to mitigate noise and reconstruct corrupted sections of the signal.

[8]J. Zhang, Y. Chen, H. Zhang and X. Shi, "Compressed Sensing for Astronomical Image Compression and Denoising," 2020 Chinese Control And Decision Conference (CCDC), Hefei, China, 2020, pp. 1162-1167, doi: 10.1109/CCDC49329.2020.9164087.

Mathematical Formulation

DCT Basis Representation

Discrete Cosine Transform - What is it?

Given a ***KxK*** image, we can represent the image as a **weighted sum of 2D basis functions**. The image (2D pixel intensity) model is shown below:

$$\hat{t}_n = \hat{w}_1 f_1(x_n, y_n) + \hat{w}_2 f_2(x_n, y_n) + \cdots + \hat{w}_D f_D(x_n, y_n)$$

Figure 4: Image Model[9]

In image reconstruction, the choice of **basis** is crucial because it determines how efficiently the image information is represented and how effectively the reconstruction can be performed. We will use the **Discrete Cosine Transform** as our choice of basis. When an image is represented by the Discrete Cosine Transform (DCT), the **frequency components** of the image are mapped in a way that reflects the rate of change across the image's spatial domain. As a result, high and low frequencies are represnted differently. We will be using the parametric expression for the DCT shown on the right, where ***x*** and ***u*** span the pixels horizontally from *1* to *K* and ***y*** and ***v*** span the pixels vertically from *1* to *K*.

$$\alpha_u \cdot \beta_v \cdot \cos\left(\frac{\pi(2x-1)(u-1)}{2\cdot P}\right) \cdot \cos\left(\frac{\pi(2y-1)(v-1)}{2\cdot Q}\right)$$

$$\alpha_u = \begin{cases} \sqrt{\frac{1}{P}} & (u=1) \\ \sqrt{\frac{2}{P}} & (2 \le u \le P) \end{cases} \qquad \beta_v = \begin{cases} \sqrt{\frac{1}{Q}} & (v=1) \\ \sqrt{\frac{2}{Q}} & (2 \le v \le Q) \end{cases}$$

Figure 5: DCT Parametric Expression[9]

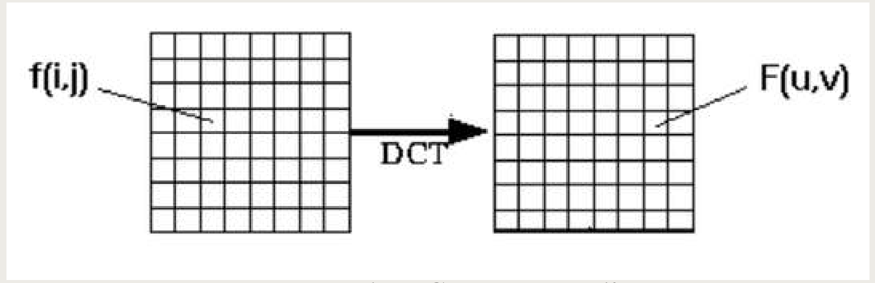


Figure 6: DCT Mapping[10]

[9]Tantum, Stacy. "Compressed Sensing Image Recovery Lecture Slides" Duke University. Feb 2024.
[10]Marshall, Dave. "The Discrete Cosine Transform (DCT)." Cardiff University, users.cs.cf.ac.uk/dave/Multimedia/node231.html. Accessed 29 Feb. 2024.

DCT Basis Representation

Discrete Cosine Transform - How are we using it?

For a given K, the parametric expression for the DCT coefficients provide a way to represent our image as a **weighted sum of basis chips**, which are simply 2D functions that represent different spatial **frequencies and positions** within the image. In order to represent the entire set of basis chips in a structured form, each basis chip is rasterized into a column vector and placed as a column in the basis vector matrix. Using a basis vector matrix facilitates linear operations on the image data and allows for efficient storage since we only have to compute it once. The basis vector matrices for *K=8* and *K=16* are shown below.

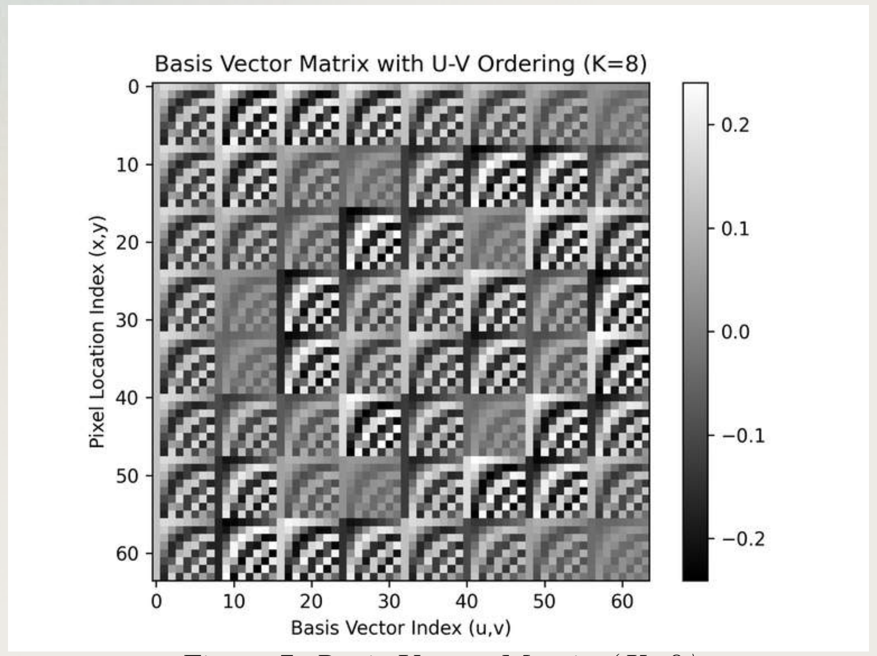


Figure 7: Basis Vector Matrix (K=8)

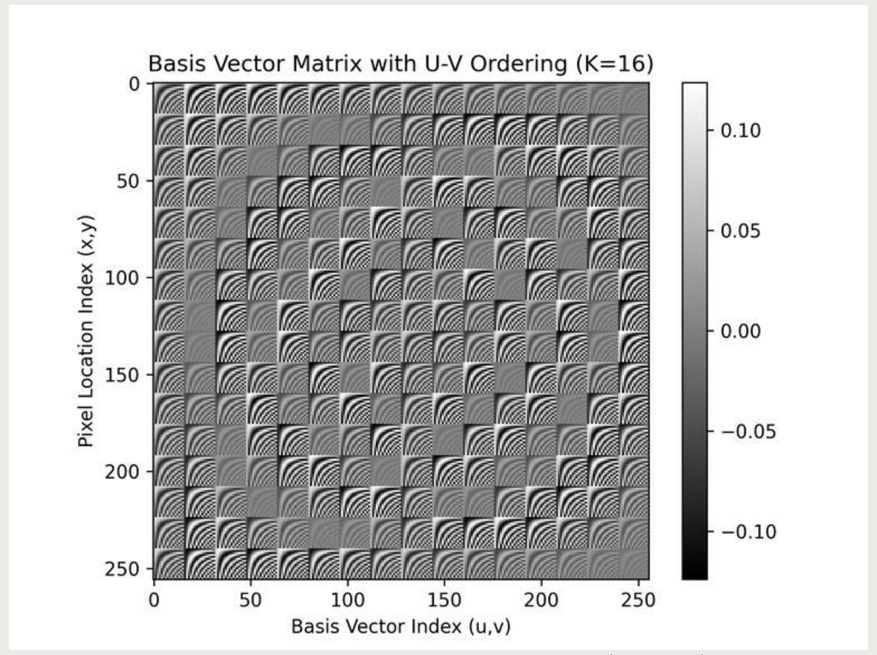


Figure 8: Basis Vector Matrix (K=16)

As mentioned previously, compressed sensing requires a constraint of **sparsity** (we'll see why shortly). This means that we need to find an accurate representation of our image in a sparse domain. Naturally, an entire image will have a variety of frequency content. This means the DCT coefficients of our entire image will most likely not be sparse. However, if we split our image into **chips/blocks** of size *KxK*, these chips are assumed to have a sparse DCT representation because of their lower frequency content. As a result, our image reconstruction algorithm will reconstruct our image chip by chip, where each chip will have its own regularized regression model.

LASSO (L1-Norm Regularization)

Pixel Estimation - Why we need sparsity?

As described by our **image model**, a complete image (or chip), ***C**,* can be represented by the matrix multiplication of the basis vector matrix ***B*** and the DCT coefficients $\boldsymbol{\kappa}$, since the complete image is now a weighted sum of the basis functions (*Figure 9*).

When we have a chip with missing pixels (a sample of the complete image vector ***C*** and corresponding basis functions within ***B***), we can fit a regressive model to the sampled data to calculate a **set of DCT coefficients that approximate our complete image**. We can then use our estimation of the DCT coefficients to approximate the values of the missing pixels and reconstruct our chip.

However, the sampled data we want to fit our regressive model to is an **undetermined linear system**. Specifically, the system $\boldsymbol{D = A \cdot \kappa}$ has an **infinite number of solutions.** Thus, in order to solve the system and estimate our DCT weights, we need to enforce that our weight vector $\boldsymbol{\kappa}$ is sparse. This will allow us to find a solution to fit the sampled data. Again, this further highlights why the DCT is a well-suited choice of basis for image reconstruction.

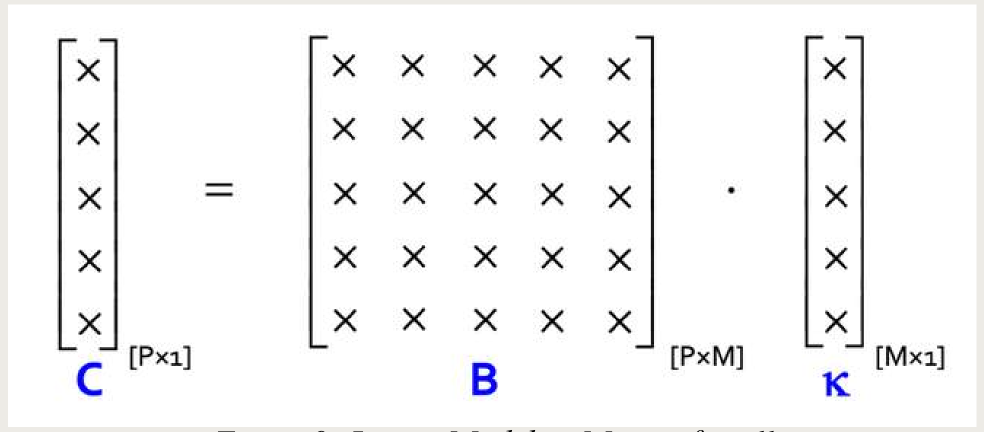


Figure 9: Image Model in Matrix form[11]

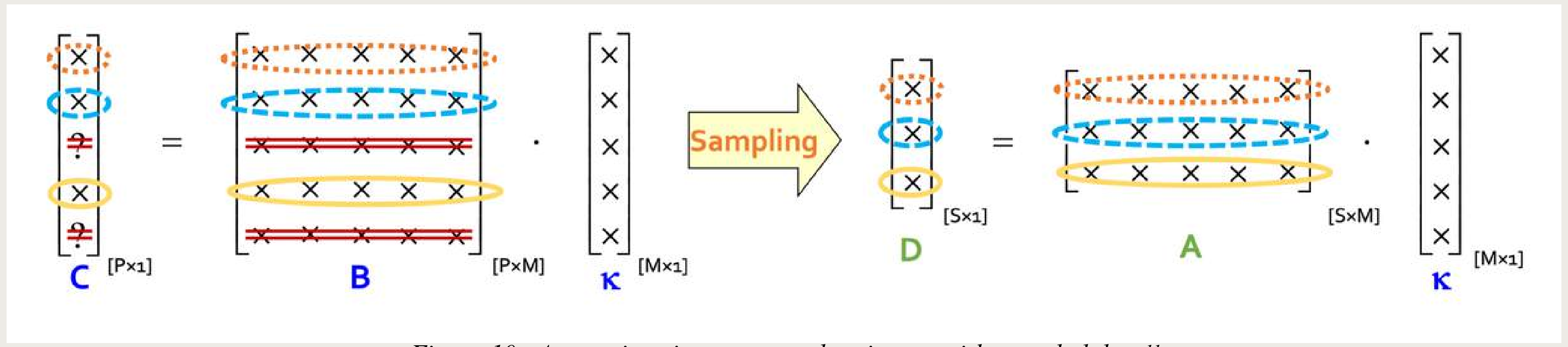


Figure 10: Approximating our complete image with sampled data[11]

[11]Tantum, Stacy. “Compressed Sensing Image Recovery Lecture Slides” Duke University. Feb 2024.

LASSO (L1-Norm Regularization)

LASSO - What is it?

In order to enforce sparsity in our weight vector and solve our system (fit a regressive model to our sampled data), we will use **L1-Norm Regularization**, or **LASSO**.[12] LASSO is a **regularization technique** that adds a penalty term to the standard least squares objective used in regression. The penatly term is proportional to the L1 norm of the coefficient vector. This changes the loss function objective to both **minimize the sum of the squared error** between the observed and predicted values while also **minimizing the values of the weights** (enforce sparsity). This is shown mathematically below:

$$\min_{\boldsymbol{\kappa}} \left\| \mathbf{A}\boldsymbol{\kappa} - \mathbf{D} \right\|_2^2 + \lambda \left\| \boldsymbol{\kappa} \right\|_1$$

Figure 11: LASSO Objective Function[13]

We choose this as our regularization approach for two main reasons. Firstly, this technique fits well with our constraint of sparsity in compressed sensing. Since LASSO penalizes the absolute value (**L1-Norm**) of our weights, LASSO is inherently **sparsity inducing**. Secondly, regularization techniques like LASSO help **prevent overfitting** by penalizing overly complex models, allowing our reconstruction to **generalize better**. Once we calculate our sparse weight vector ***κ*****,** we can multiply it by the basis functions of corresponding missing pixels to reconstruct our complete chip.

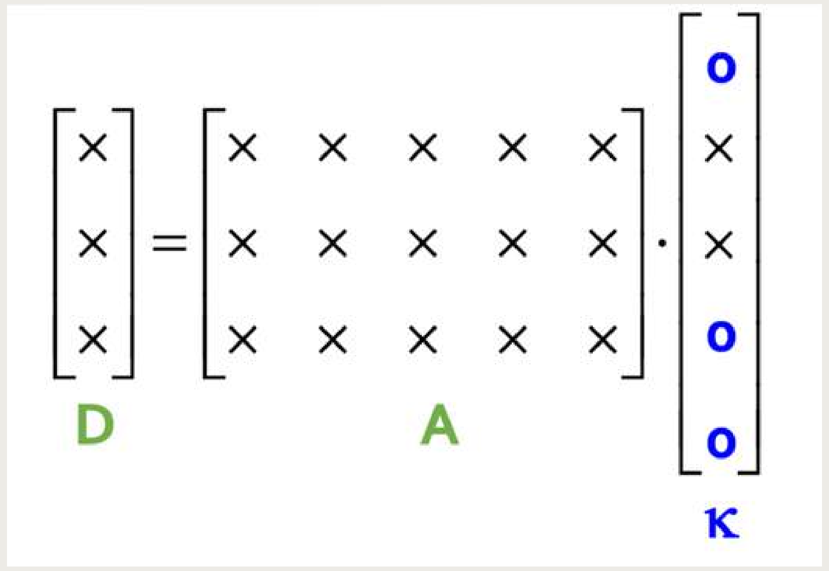


Figure 12: Sampled Pixels with Sparse κ[13]

Note: We exclude the constant in our basis vector matrix from regularization to prevent biased regression.

[12]Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267–288.
[13]Tantum, Stacy. "Compressed Sensing Image Recovery Lecture Slides" Duke University. Feb 2024.

Image Reconstruction

Cross Validation - Selecting Regularization Strength

The strength of the LASSO regularization depends on the **hyperparameter** λ. This hyperparameter controls the trade off between the goodness of fit and the degree of regularization. Instead of using a single split to find our **optimal regularization strength**, we can find the optimal regularization strength through **cross validation with random subsets**.

For each chip, we iterate through ***M=20*** cross-validation folds. On each fold, we split the ***S*** pixels present in the chip into a **testing set** of ***m=floor(S/6)*** pixels, while the rest are used as our **training set**. We train our regularized regression model on the training set of pixels and use the model to predict the pixels in the testing set. We can calculate the **mean squared error** between the predicted pixels and the pixels within the testing set to measure the performance of our model with a particular regularization strength. We can then choose the regularization strength that minimizes our mean square error and use the corresponding model to predict our missing pixels.

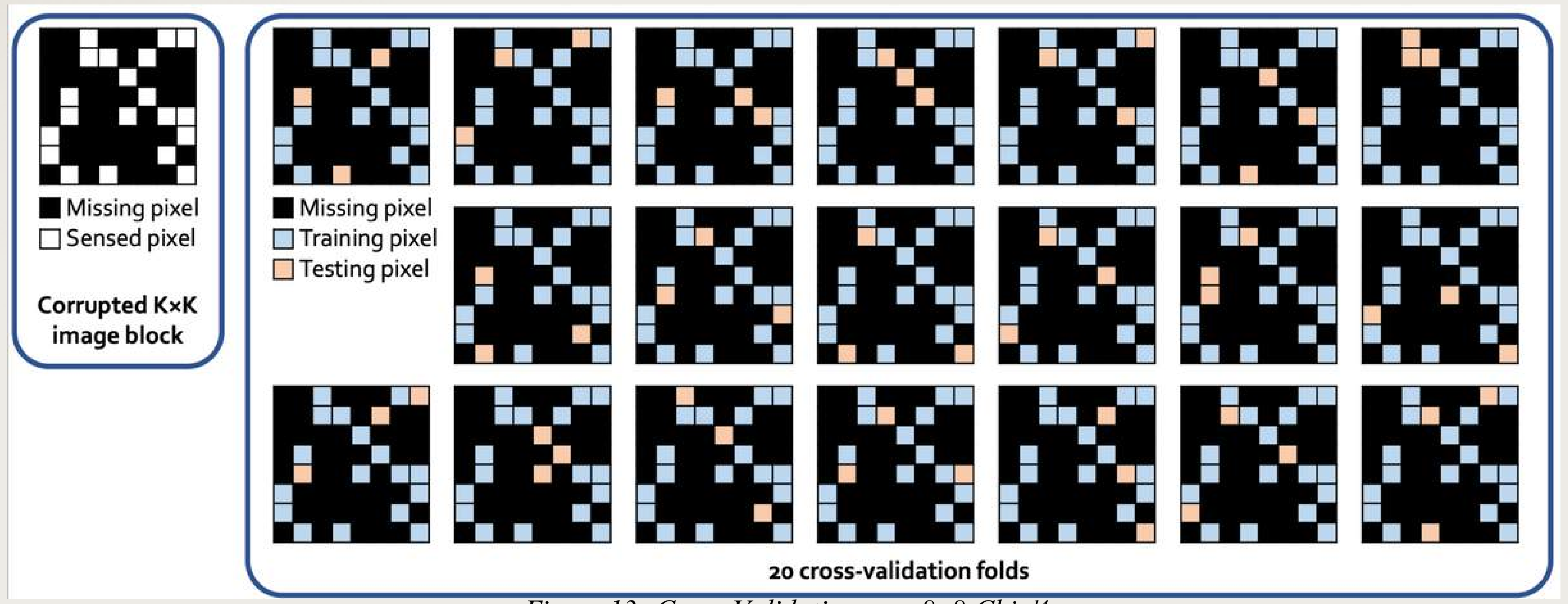


Figure 13: Cross Validation on a 8x8 Chip[14]

[14]Tantum, Stacy. "Compressed Sensing Image Recovery Lecture Slides" Duke University. Feb 2024.

Image Reconstruction

Image Reconstruction - Overview

At this point, we have everything we need to reconstruct our image. Since we're relying on the **sparsity** of our image's DCT coefficients, our algorithm breaks up our image into ***KxK* chips** (*Figure 14*). Each chip will contain some number of missing pixels. If we're using a test image, we can synthesize corrupted chips by choosing to sense ***S*** pixels within the chip (used in simulations). For each chip, we run **cross validation with random subsets**, in which we split our present pixels into a training set and testing set. We train a **regression model** with **LASSO regularization** in each iteration of cross validation and measure its performance by calculating the **mean squared error (MSE)** between the predicted pixels and the testing set. Once we've found our optimal regularization strength for the chip, we can use our model to predict the missing pixels and reconstruct our chip. We reconstruct every chip in the image to recover our entire image.

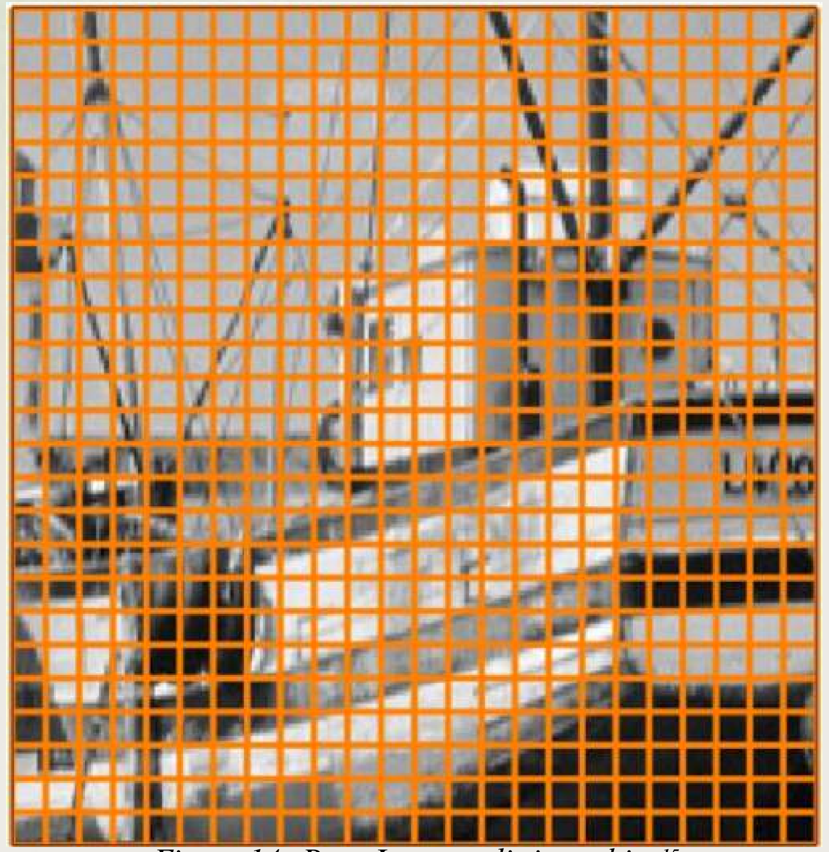
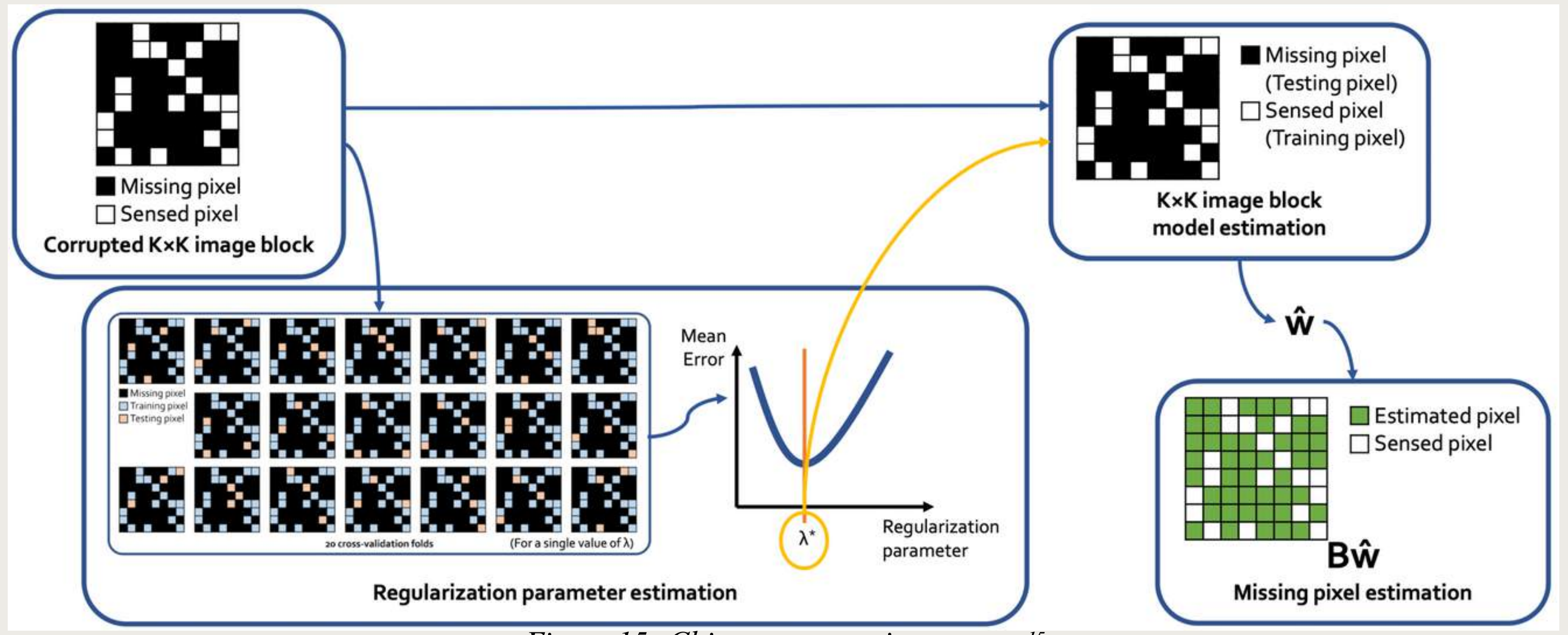


Figure 14: Boat Image split into chips[15]



■ Missing pixel
□ Sensed pixel
Corrupted K×K image block
■ Missing pixel (Testing pixel)
□ Sensed pixel (Training pixel)
K×K image block model estimation
ŵ
Missing pixel
Training pixel
Testing pixel
20 cross-validation folds
(For a single value of λ)
Mean Error
λ*
Regularization parameter
Regularization parameter estimation
■ Estimated pixel
□ Sensed pixel
Bŵ
Missing pixel estimation

Figure 15: Chip reconstruction process[15]

[15]Tantum, Stacy. "Compressed Sensing Image Recovery Lecture Slides" Duke University. Feb 2024.

Image Reconstruction

Post Processing - Median Filtering

After reconstructing our image, we can improve the quality of our image with **median filtering**. Median filtering is a nonlinear filtering technique that replaces each pixel with the median pixel value of its neighboring pixels within a kernel window. This process **removes impulsive noise** within the image, **smoothing out** our reconstructed image.[16] We choose median filtering over other filtering approaches such as frequency selective filters (low-pass and high-pass) because median filtering preserves the structural features and detail of an image. It does not blur the image or attenuate high frequency components, which are important for image quality and sharpness. We will compare the error of the recovered image with median filtering and without median filtering.

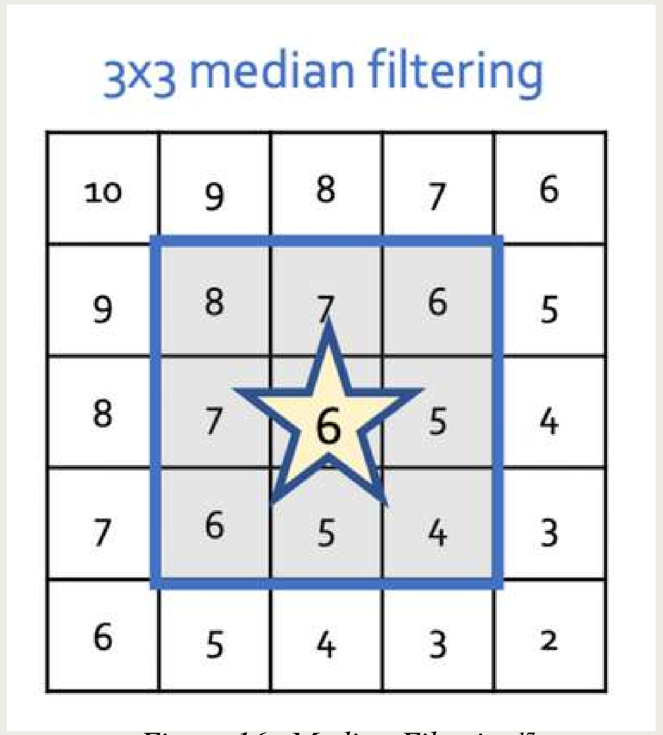


Figure 16: Median Filtering[17]

[16]Fisher, R. "Median Filter." The University of Edinburgh, homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm. Accessed 29 Feb. 2024.
[17]Tantum, Stacy. "Compressed Sensing Image Recovery Lecture Slides" Duke University. Feb 2024.

Simulations

Image Corruption Process

Synthesizing Corrupted Images - Boat

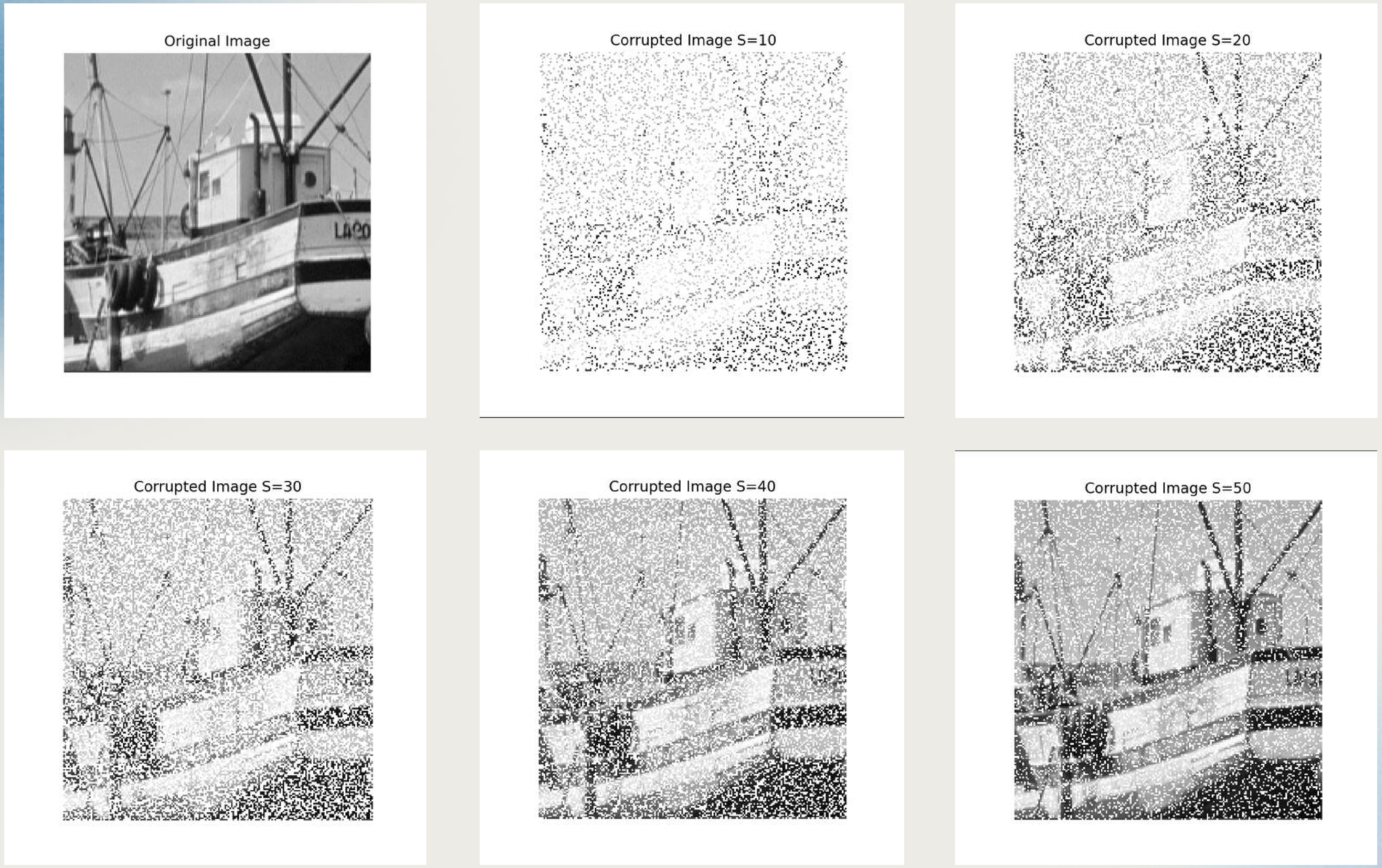


Figure 17: Corrupted Boat Images

Image Corruption Process

Synthesizing Corrupted Images - Nature

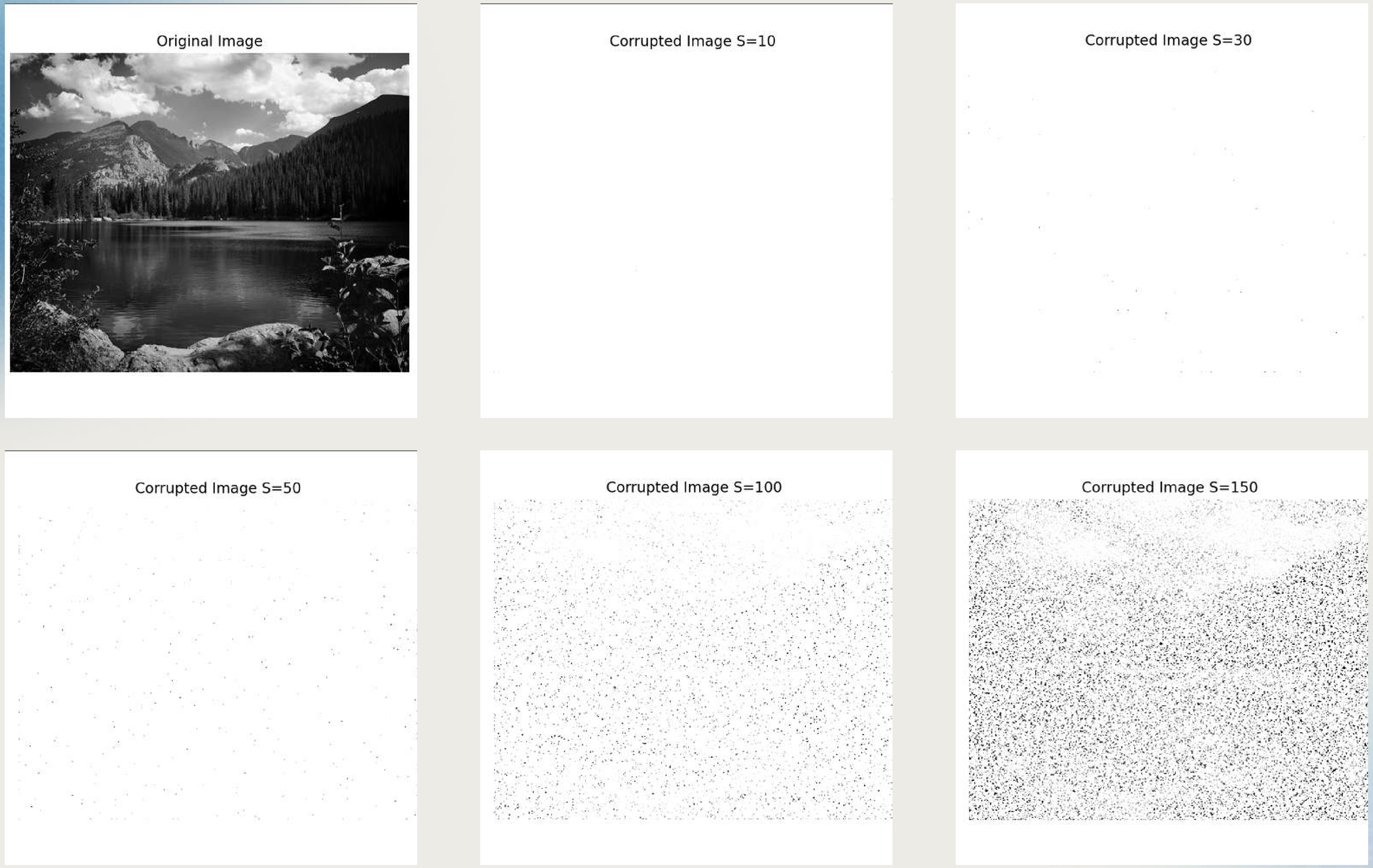


Figure 18: Corrupted Nature Images

Image Corruption Process

Synthesizing Corrupted Images - Observations

For our simulations, we can **synthesize corrupted images** from the subject images by choosing to "sense" ***S*** pixels in each block. This means that for each block of size ***KxK***, we randomly keep ***S*** pixels and treat the rest as missing. We will then test our image reconstruction algorithm on our corrupted images and measure its performance. For our simulations, we synthesized the following images:

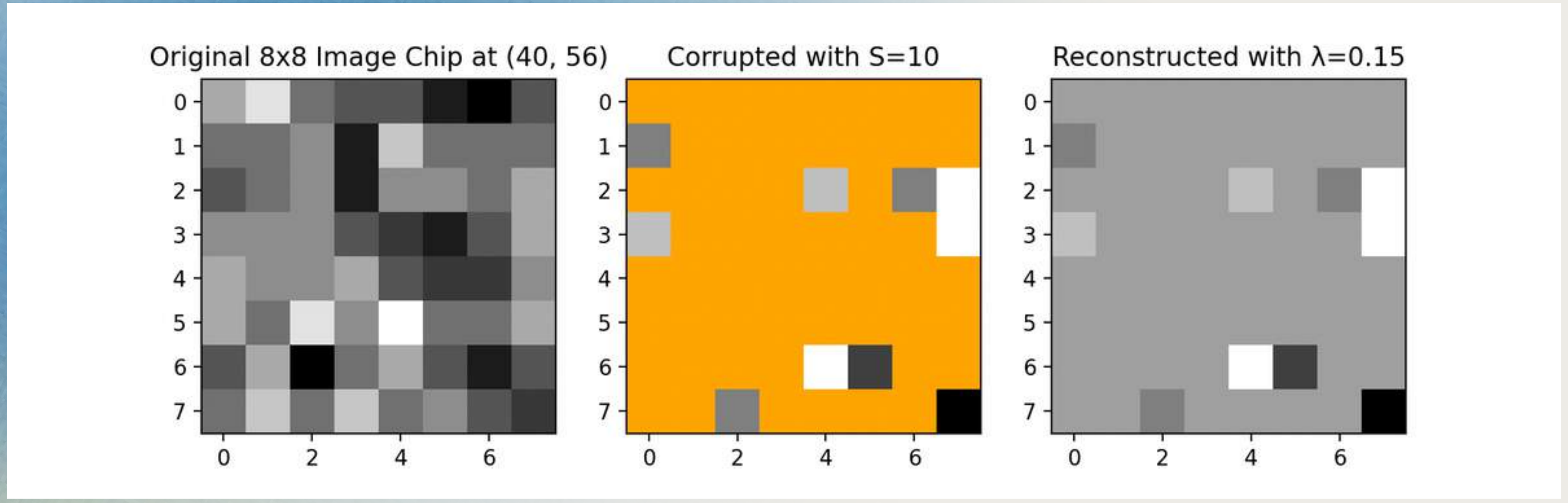
- Boat: ***S={10, 20, 30, 40, 50}***, ***K=8***
- Nature: ***S={10, 30, 50, 100, 150}, K=16***

Figure 17 and *Figure 18* show the synthesized corrupted images for the boat and nature images respectively. We can see that when ***S*** is low, the image is highly corrupted with most pixels missing. When ***S*** is high, we are choosing to keep more pixels per block, and so our image is less corrupt. Intuitively, we expect that our image reconstruction to be **more difficult** and **erroneous** as ***S* decreases**, as there is less information available about the image.

Note: Choosing to sense ***S=10*** pixels using a ***K=16*** means we only choose to sense *10* pixels in a block with *256* pixels and treat the remaining *246* pixels as missing. This results in an image with almost no information retained (as shown in the Nature subject image in *Figure 18*).

Single Chip Reconstruction

Chip Level Reconstruction - Boat



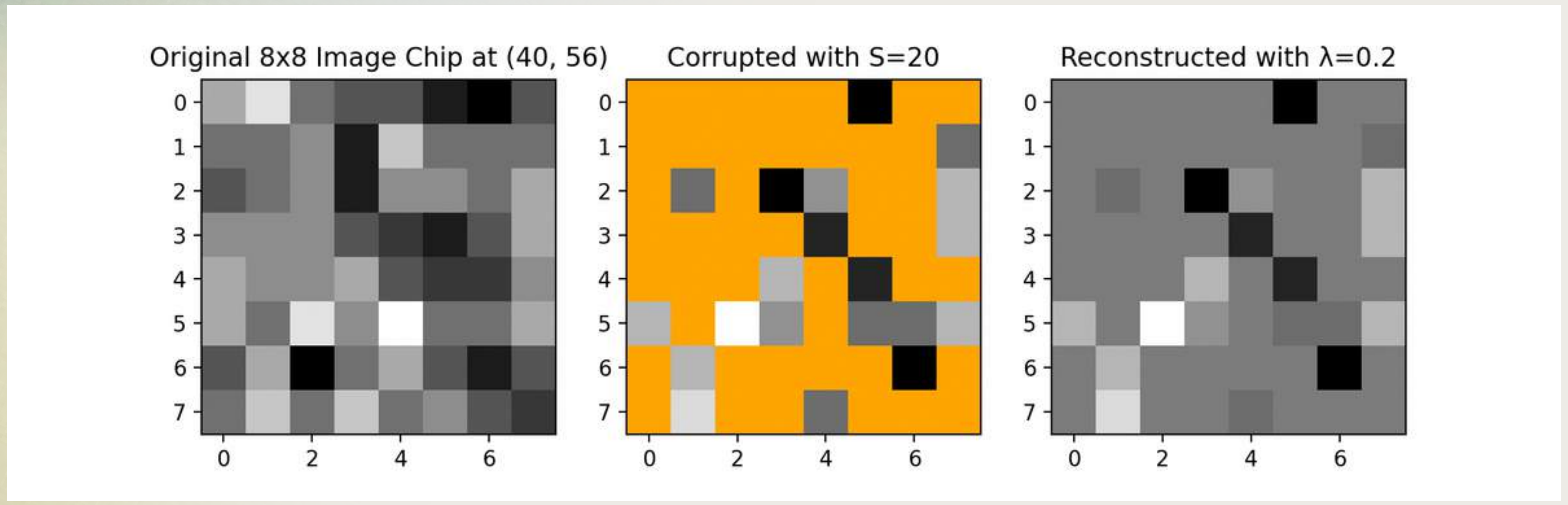
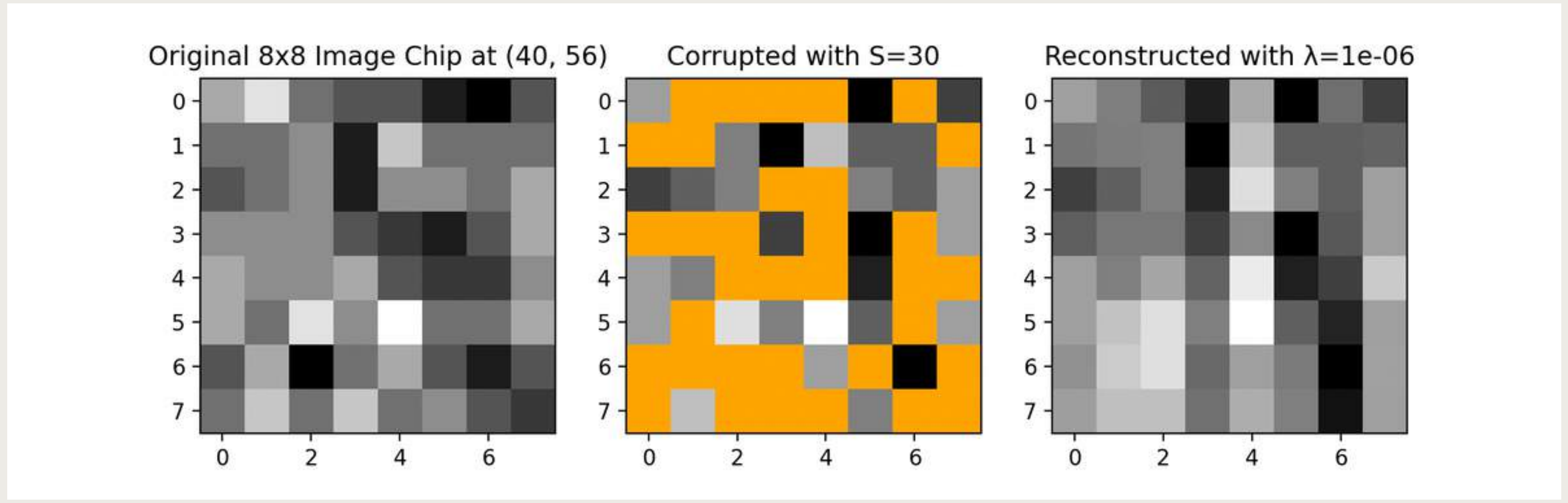
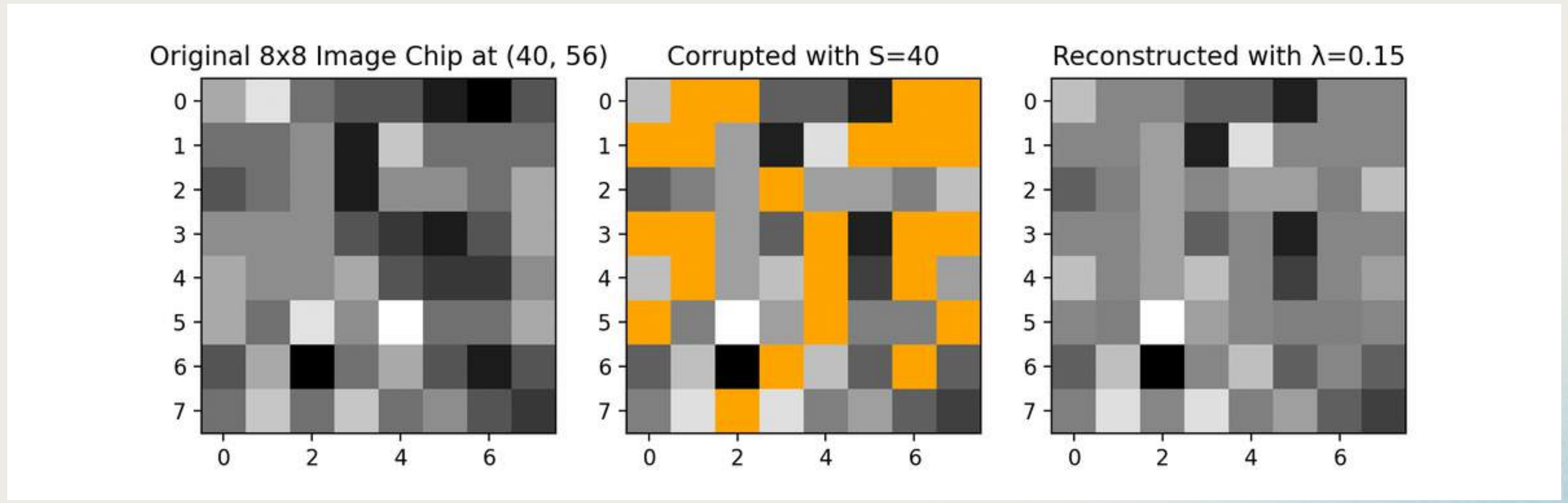
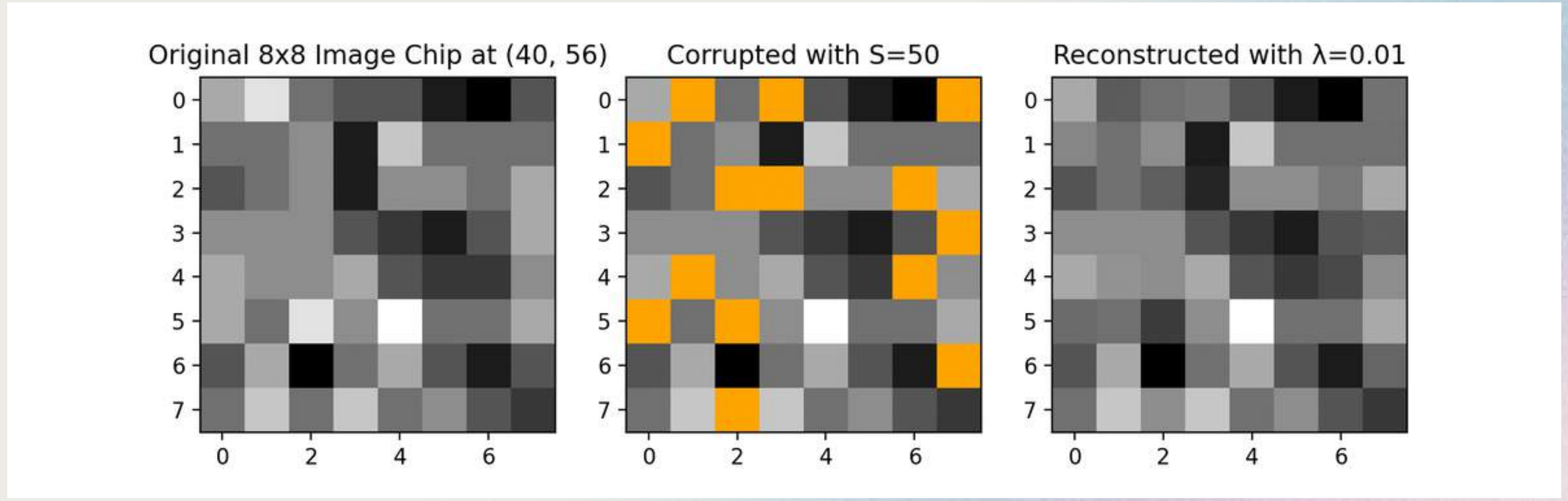


Figure 19: Boat Image Chip Reconstruction

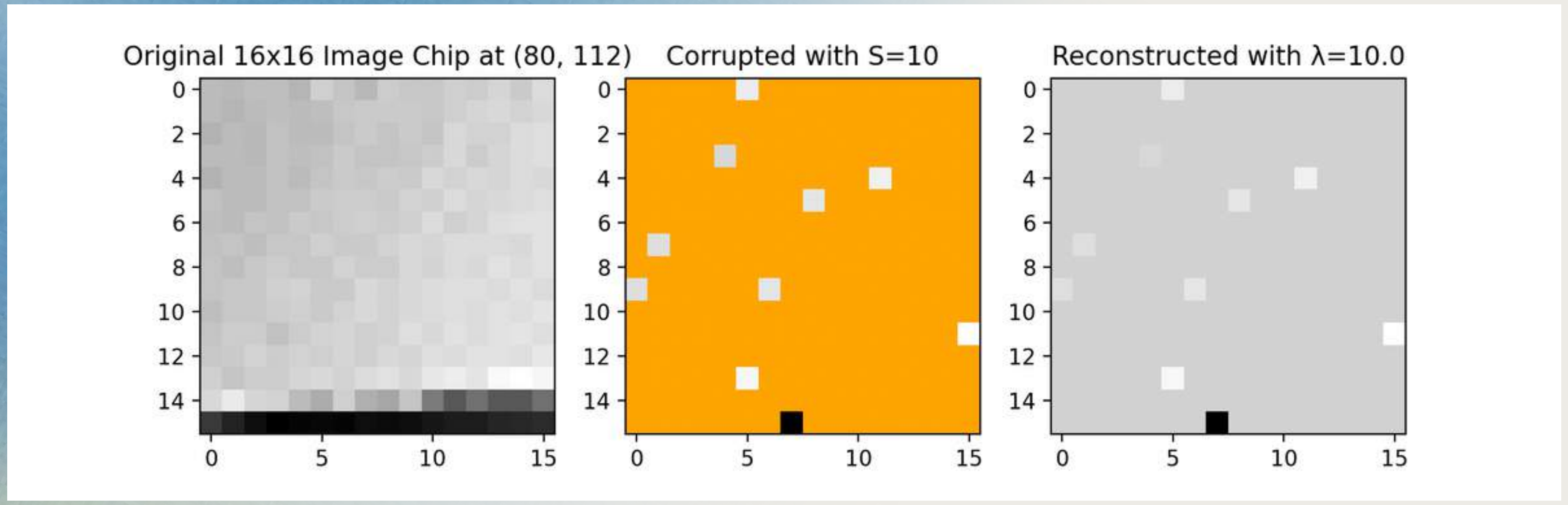


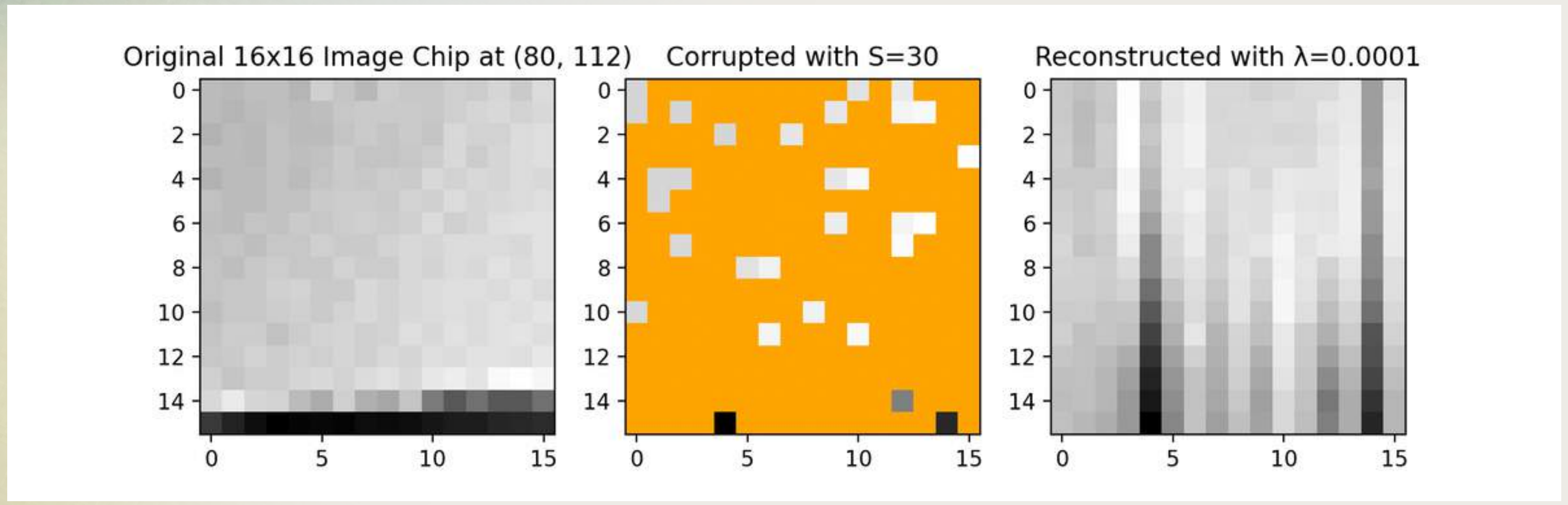


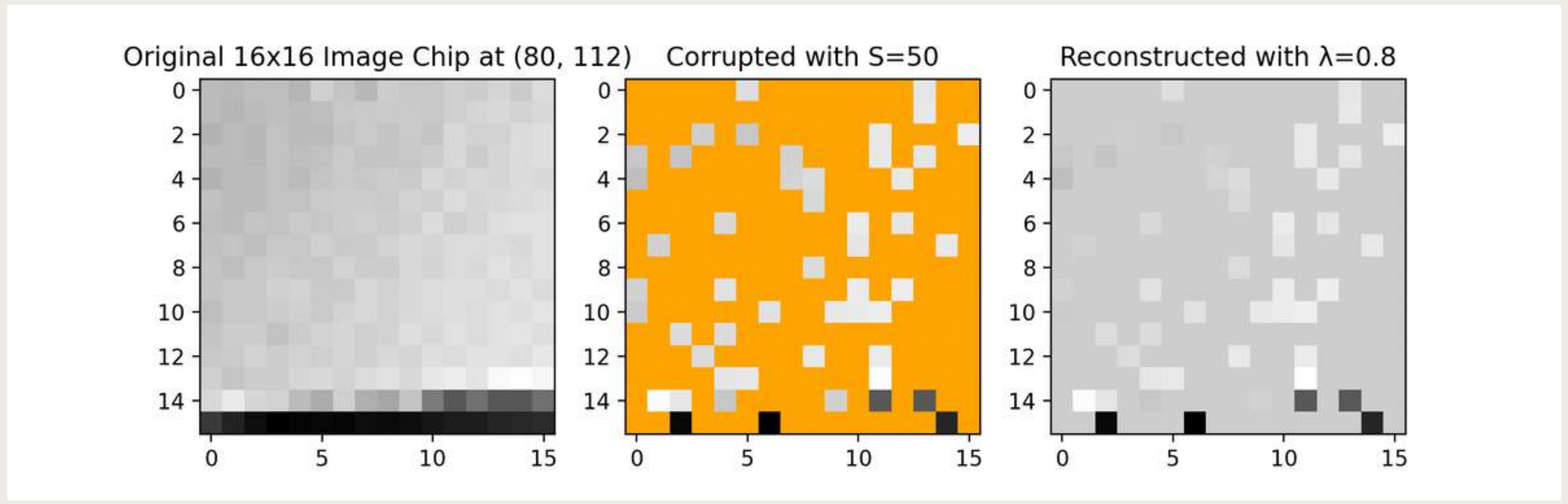


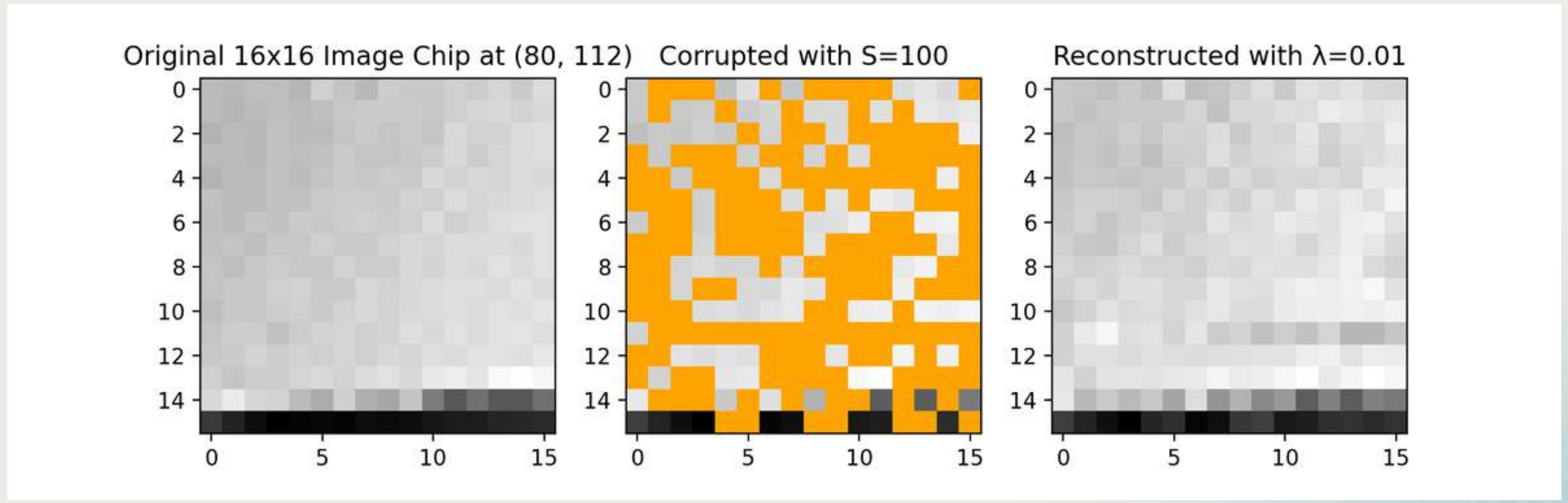
Single Chip Reconstruction

Chip Level Reconstruction - Nature









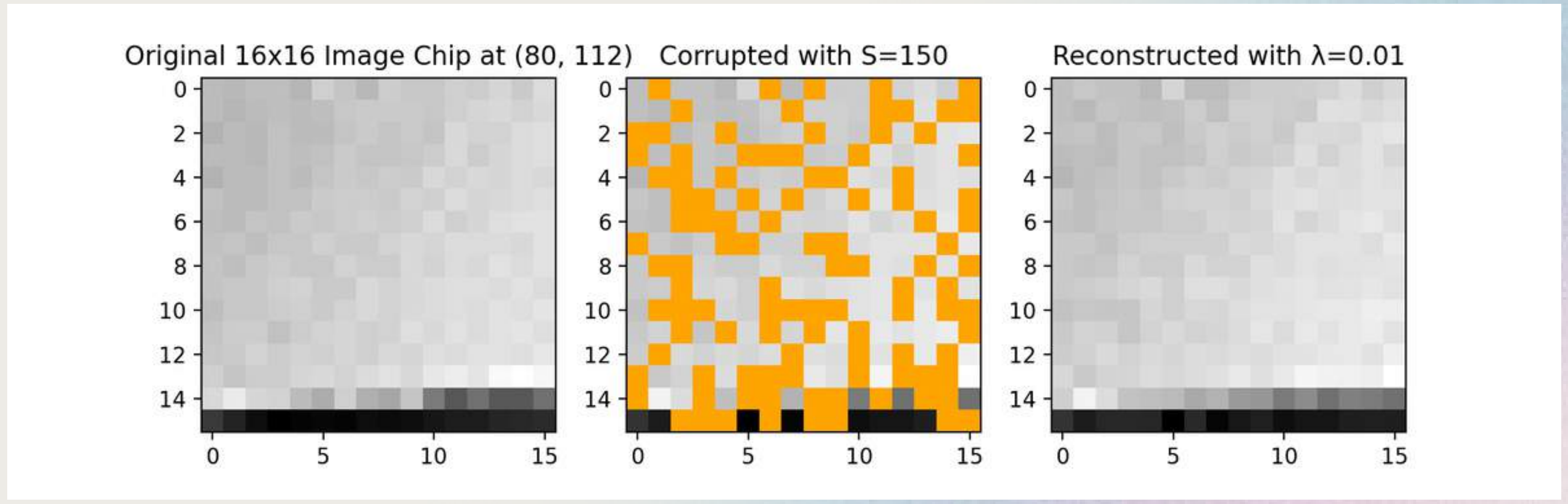


Figure 20: Nature Image Chip Reconstruction

Single Chip Reconstruction

Chip Level Reconstruction - Observations

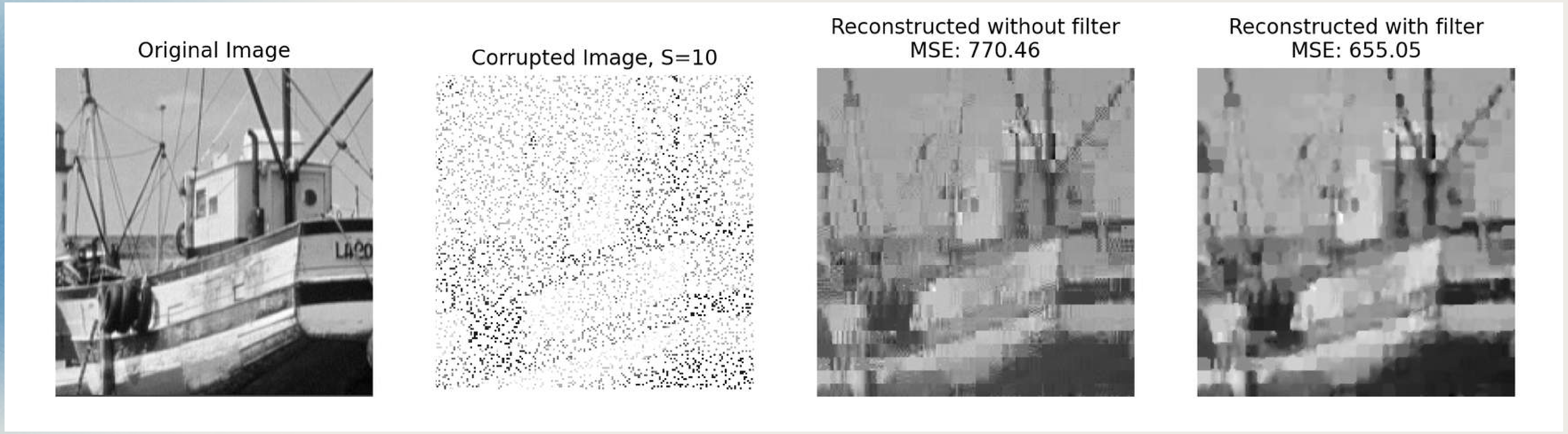
We can visualize our reconstruction algorithm on the chip level to test our model's **functionality** and **performance**. *Figures 19* and *20* show the **chip level reconstruction process** for the boat and nature images respectively. Each plot highlights the reconstruction process for a sampled chip within the image. The **original chip** is plotted along with the synthesized **corrupted chip** for each value of ***S*** (missing pixels are displayed in orange) and the **reconstructed chip**. For each synthesized corrupted chip, we perform our cross validation process to determine the optimal regularization strength and choose that to reconstruct the chip, which is displayed in each plot on the right.

We can see that for high values of ***S*** *(**S=50*** for boat*, **S=150*** for nature*)*, the reconstructed chip seems to be **almost indistinguishable** from the original chip. The pixels that are being predicted are very close in color to the original pixels (lighter pixels are predicted to have a lighter tone, darker pixels are predicted to have a darker tone).

As we decrease ***S*** for both images, we can see the similarity between the reconstructed chip and the original chip decrease. For example, the reconstruction for the nature image with ***S=30*** looks different from the original. This makes sense as sensing a lower number of pixels means that our model has **less information** about the **frequency variation** of the chip. We can see a good example of this loss of information with the ***S=10*** reconstruction of the nature image (*Figure 20*). Our regressive model is being trained on only 10 pixels, 9 of which are white. As a result, the reconstruction is mostly an average of these lighter tones.

Image Reconstruction Results

Subject Image Reconstruction - Boat



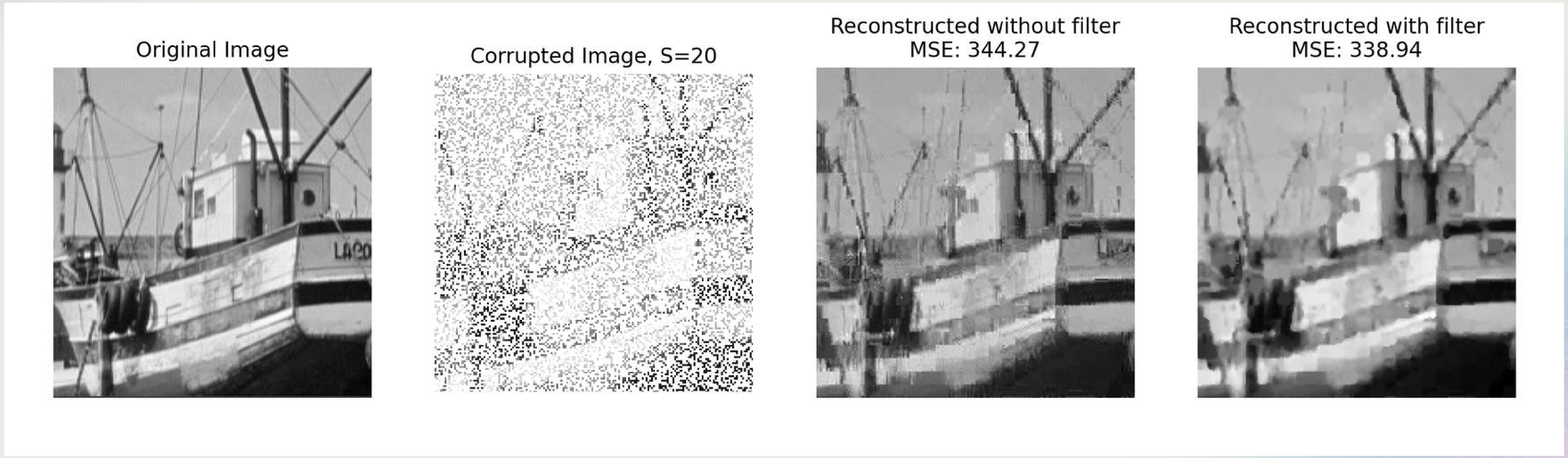
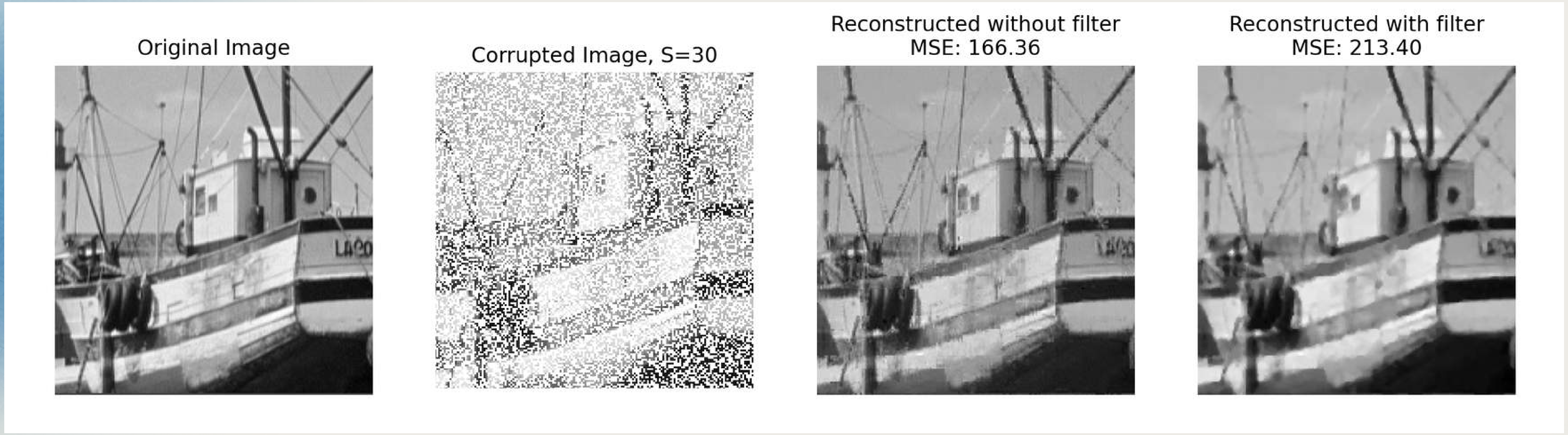


Figure 21: Boat Image Reconstructions

Image Reconstruction Results

Subject Image Reconstruction - Boat



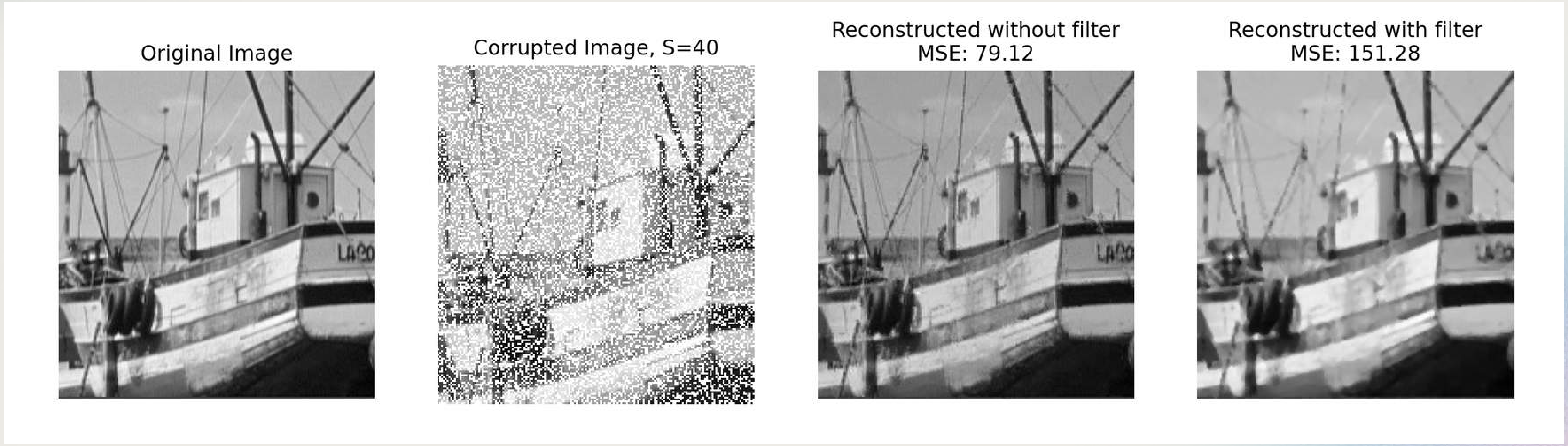


Figure 21: Boat Image Reconstructions

Image Reconstruction Results

Subject Image Reconstruction - Boat

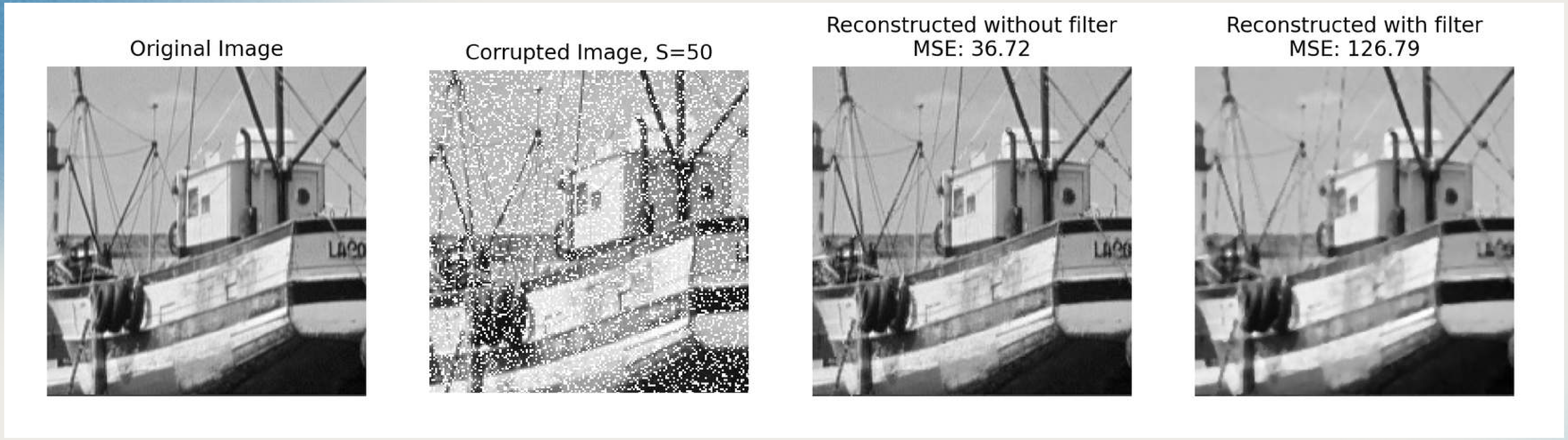
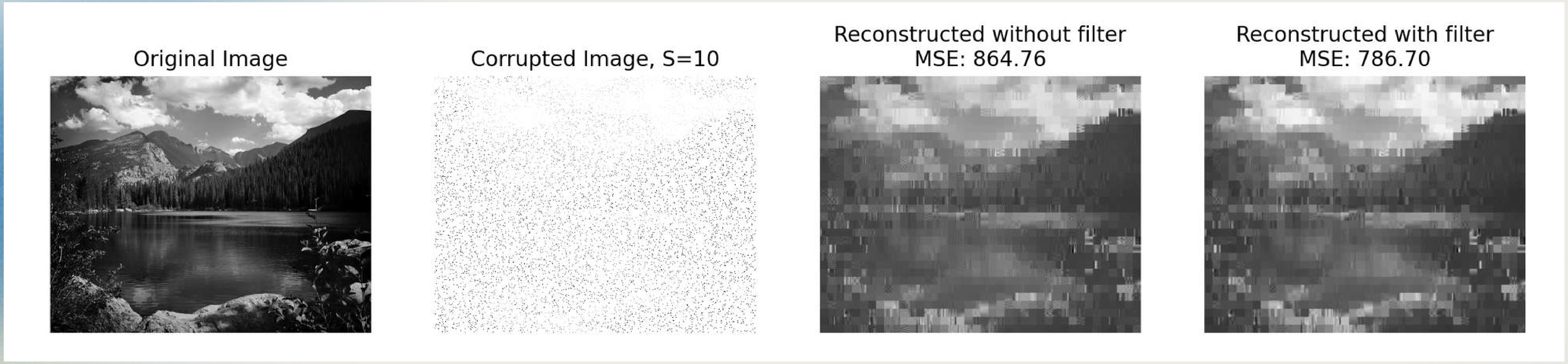


Figure 21: Boat Image Reconstructions

Image Reconstruction Results

Subject Image Reconstruction - Nature



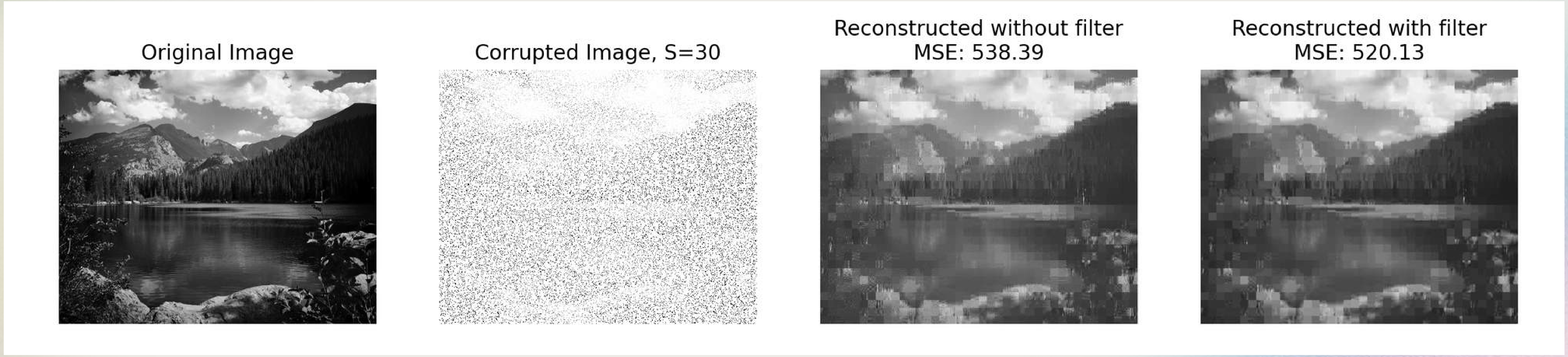


Figure 22: Nature Image Reconstructions

Image Reconstruction Results

Subject Image Reconstruction - Nature

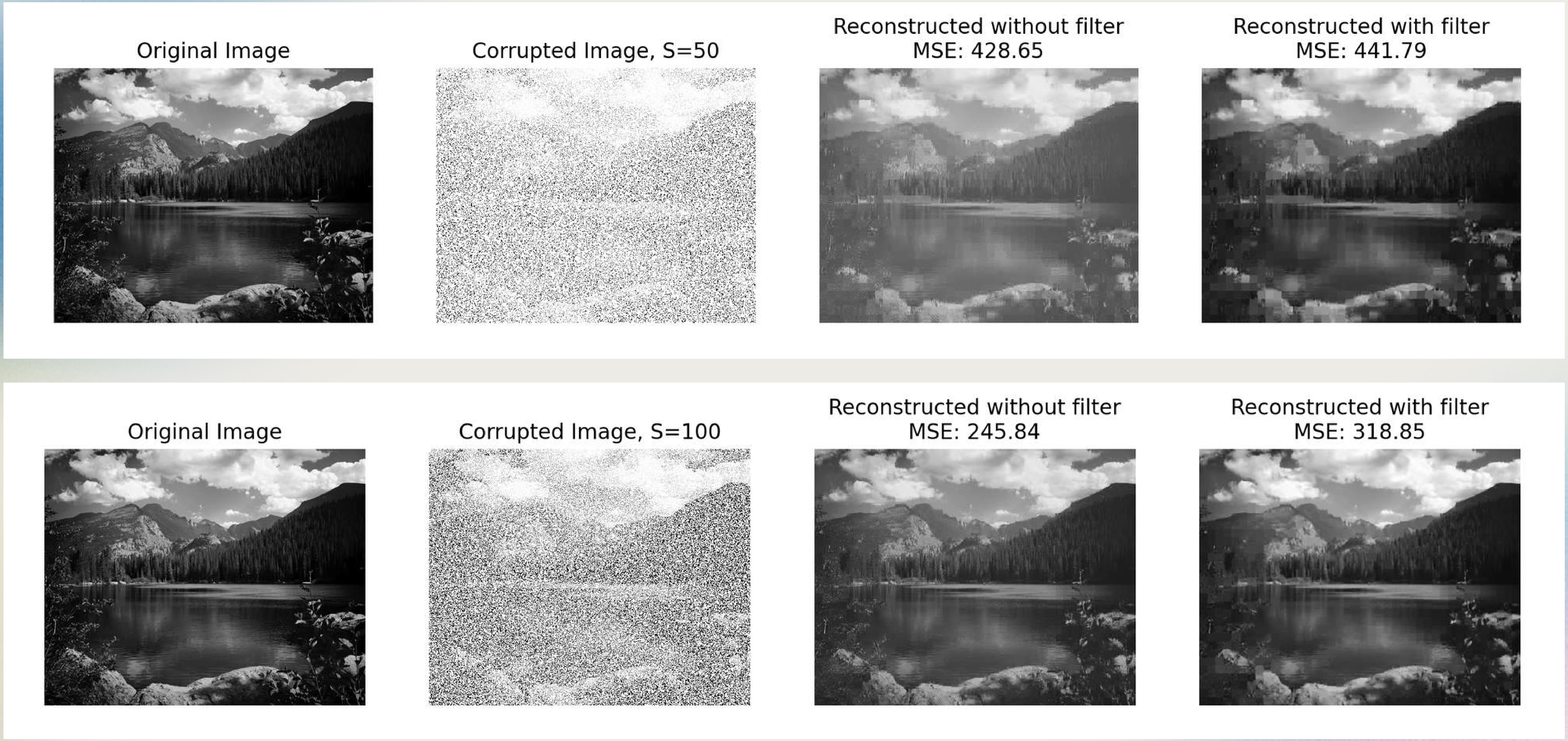


Figure 22: Nature Image Reconstructions

Image Reconstruction Results

Subject Image Reconstruction - Nature

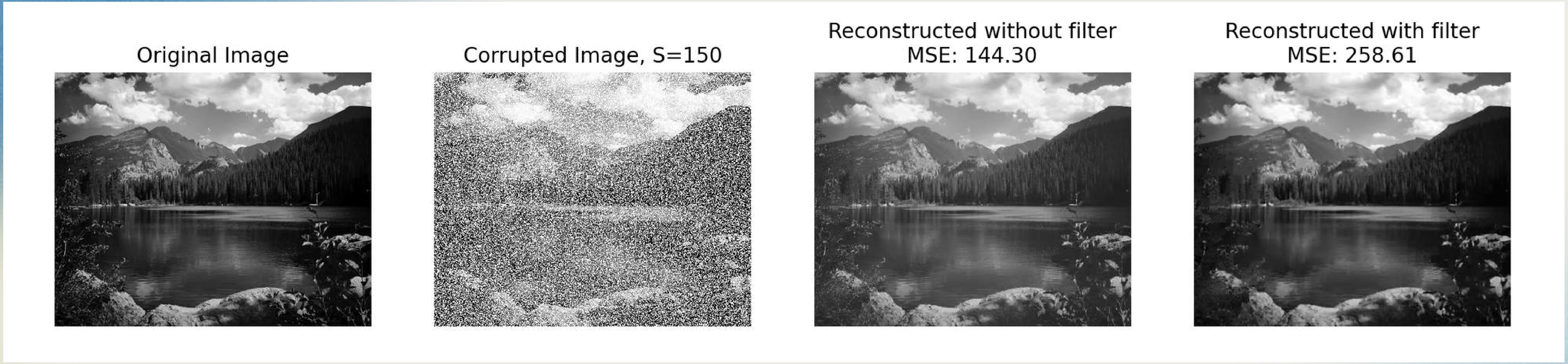


Figure 22: Nature Image Reconstructions

Image Reconstruction Results

Subject Image Reconstruction - Observations

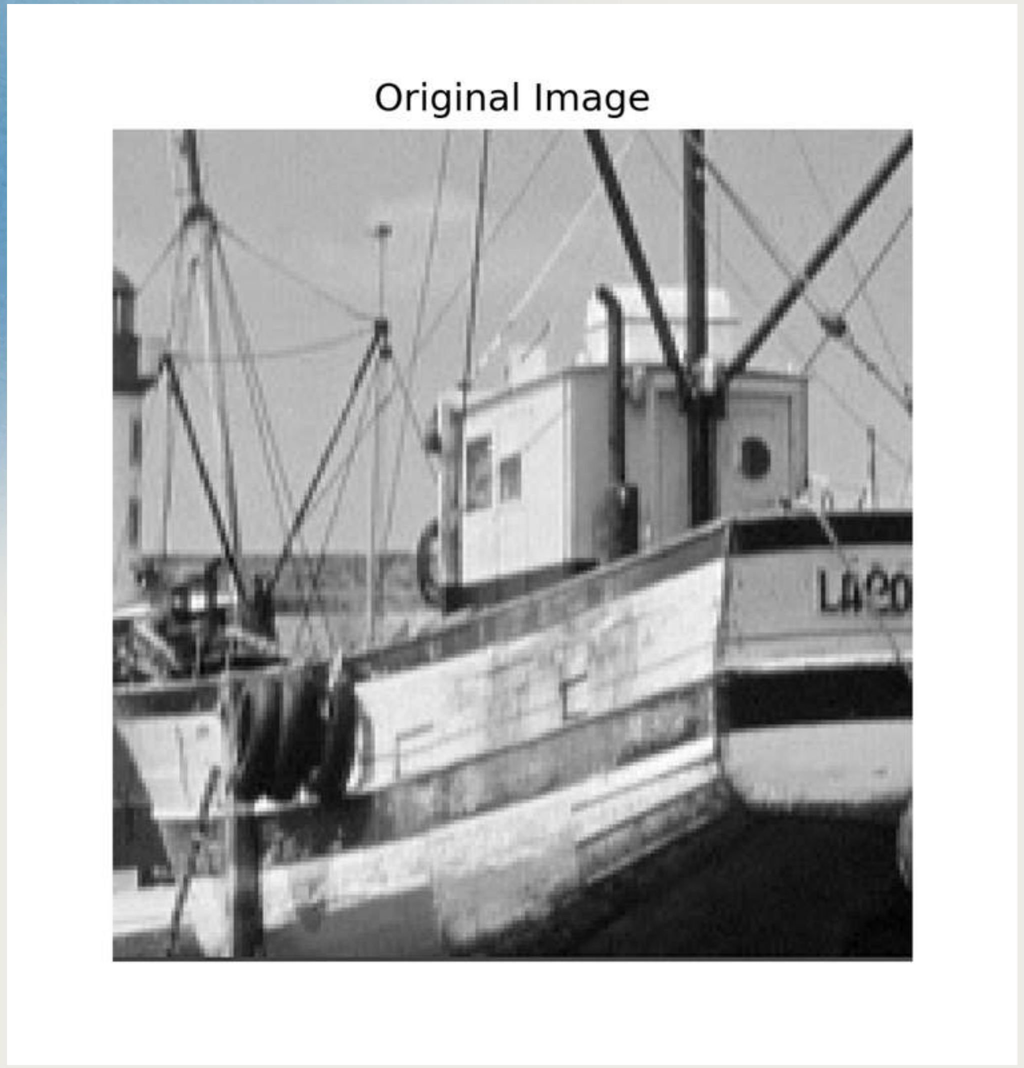
Results: It was **almost magical** to see the image reconstruction algorithm recover the image with the information presented to it. For example, it is impossible for a human to see a boat in the ***S=10*** corrupted boat image. However, the reconstruction clearly shows a boat with its original features present. The same is true with the ***S=10*** corrupted nature image; **the reconstruction reveals insight** about the landscape that wasn't present before. The reconstructions for lower values of ***S*** are blurry (***S=10*** for boat, ***S=10*** for nature), while the reconstructions for larger values of ***S*** are sharp (***S=50*** for boat, ***S=150*** for nature).

Reconstruction quality decreases as the degree of corruption increases. For both subject images, the **quality of the image increases** and the **MSE decreases** as we **increase *S***. For example, the ***S=10*** reconstruction of the nature image has an MSE of *786.70*. This MSE consistently drops as ***S*** increases, reaching *258.61* when ***S=150***. This aligns with our previous intuition: a lower number of missing pixels is can be reconstructed more accurately because there is less room for error and our model is trained with more information.

Median filtering doesn't always enhance image reconstruction. When ***S*** is low, **median filtering** tends to help the quality of the image. For example, the ***S=20*** reconstruction for the boat image has an MSE of *344.27* without median filtering and an MSE of *338.94*. In these cases, median filtering enhances the visual quality of the image as well, **making blurred sections sharper**. As ***S*** increases, however, median filtering seems to harm the quality of our image. For example, the ***S=150*** reconstruction for the nature image has an MSE of *144.30* without median filtering and an MSE of *258.61* with median filtering. This makes sense since applying median filtering with a low number of missing pixels will more likely distort true pixels than fix missing pixels.

Image Reconstruction Results

Regularization Strength Variance - Boat



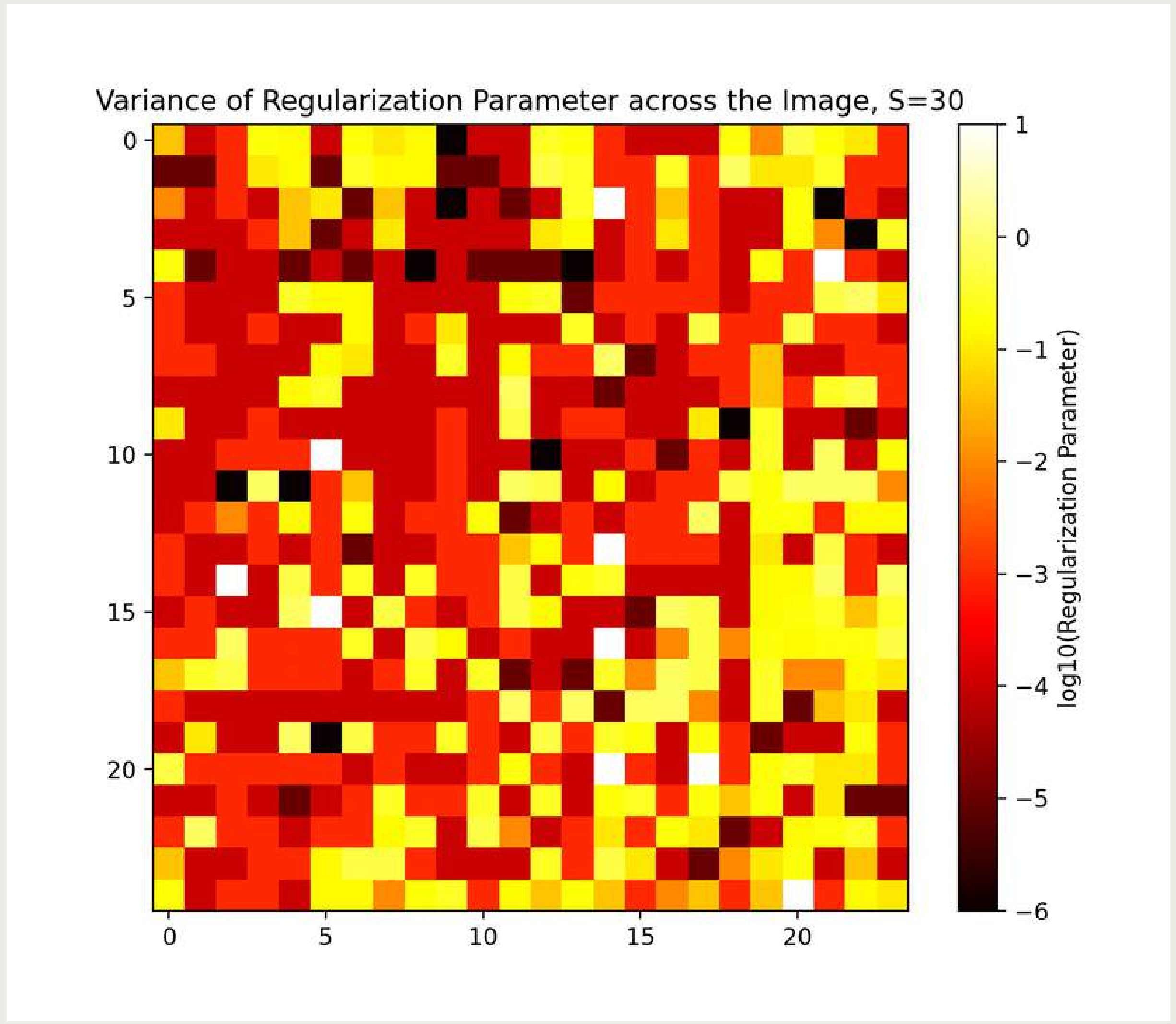
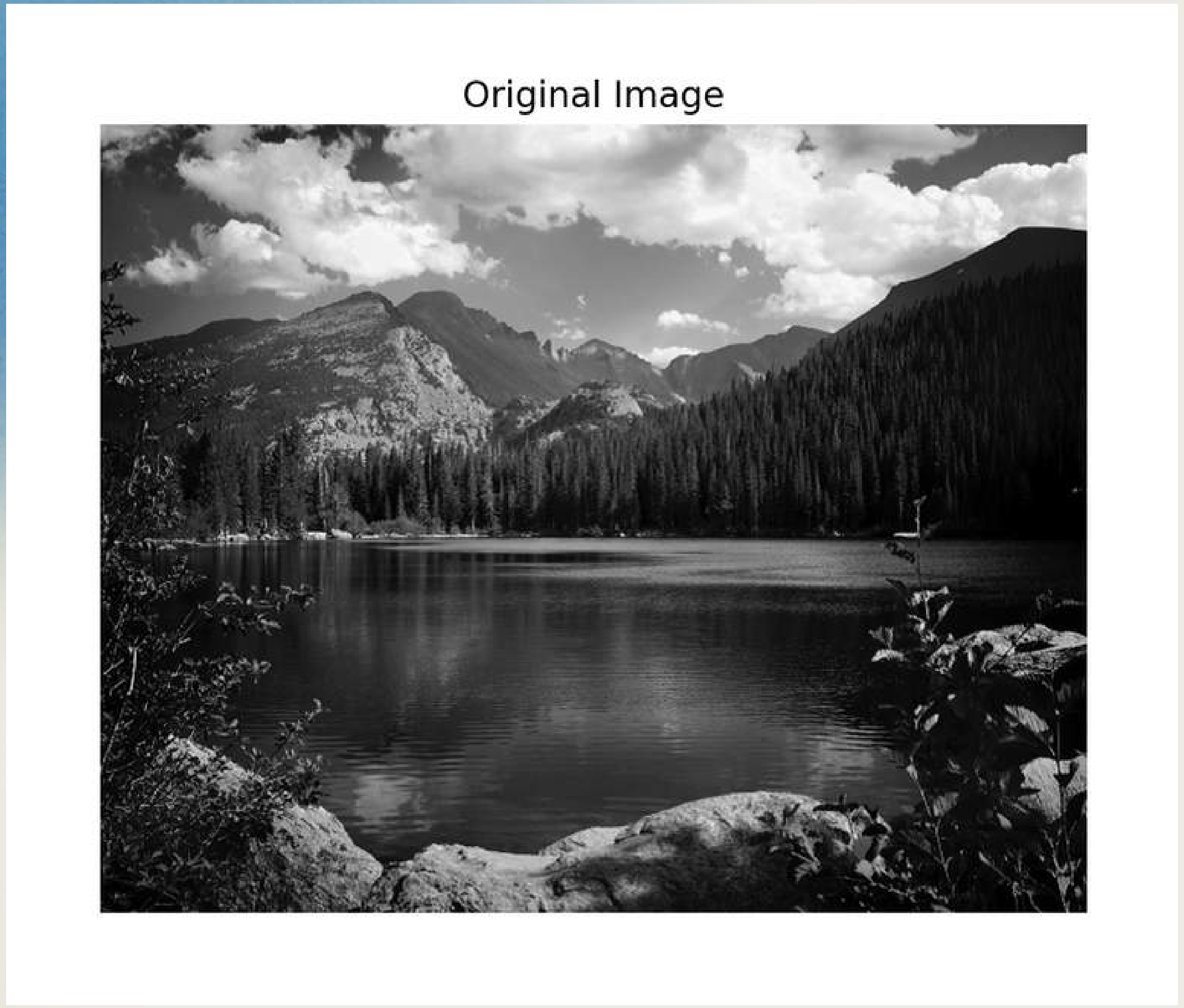


Figure 23: Regularization Parameter Variance - Boat

Image Reconstruction Results

Regularization Strength Variance - Nature



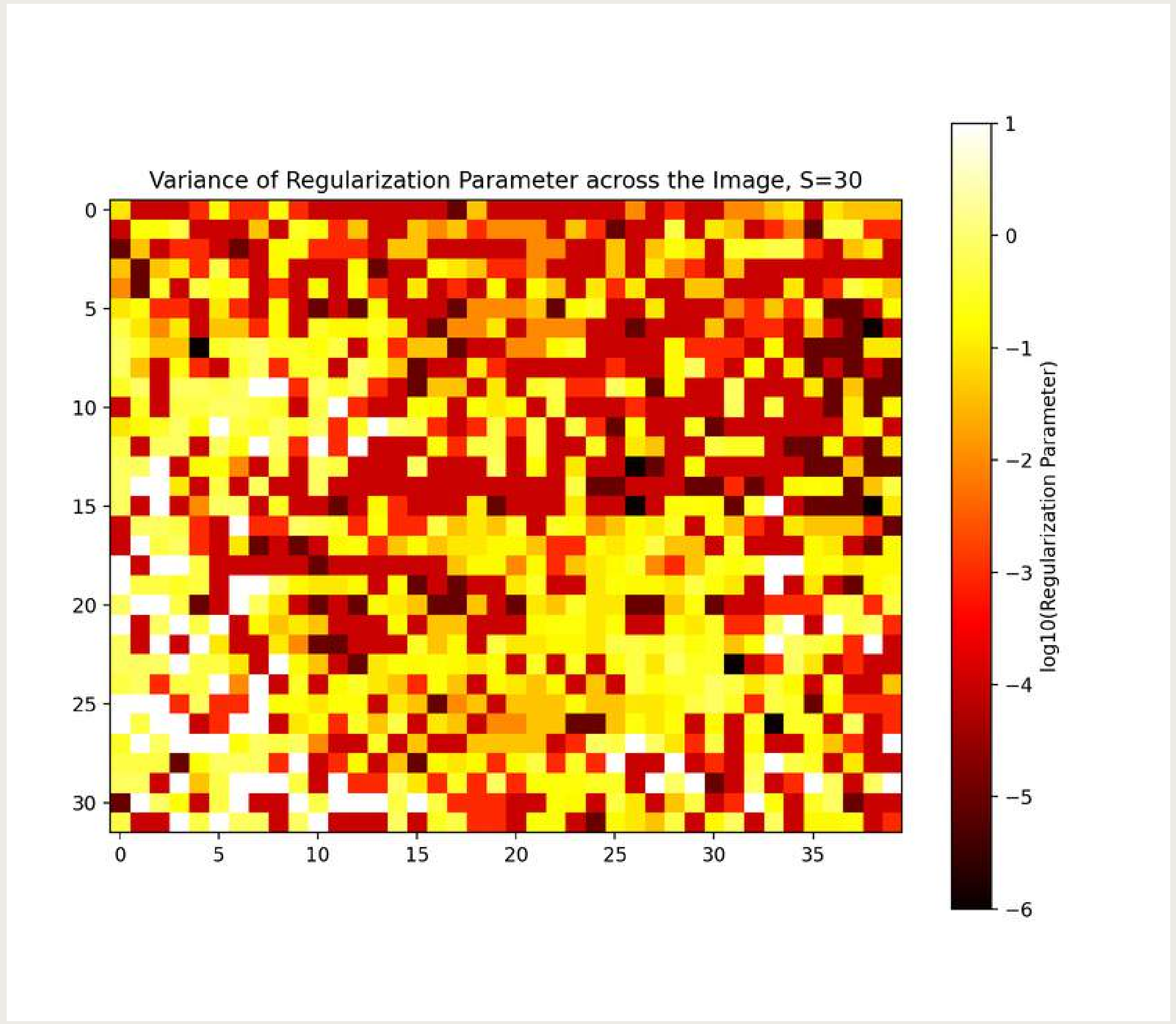


Figure 24: Regularization Parameter Variance - Nature

Image Reconstruction Results

MSE vs. Measure of Corruption

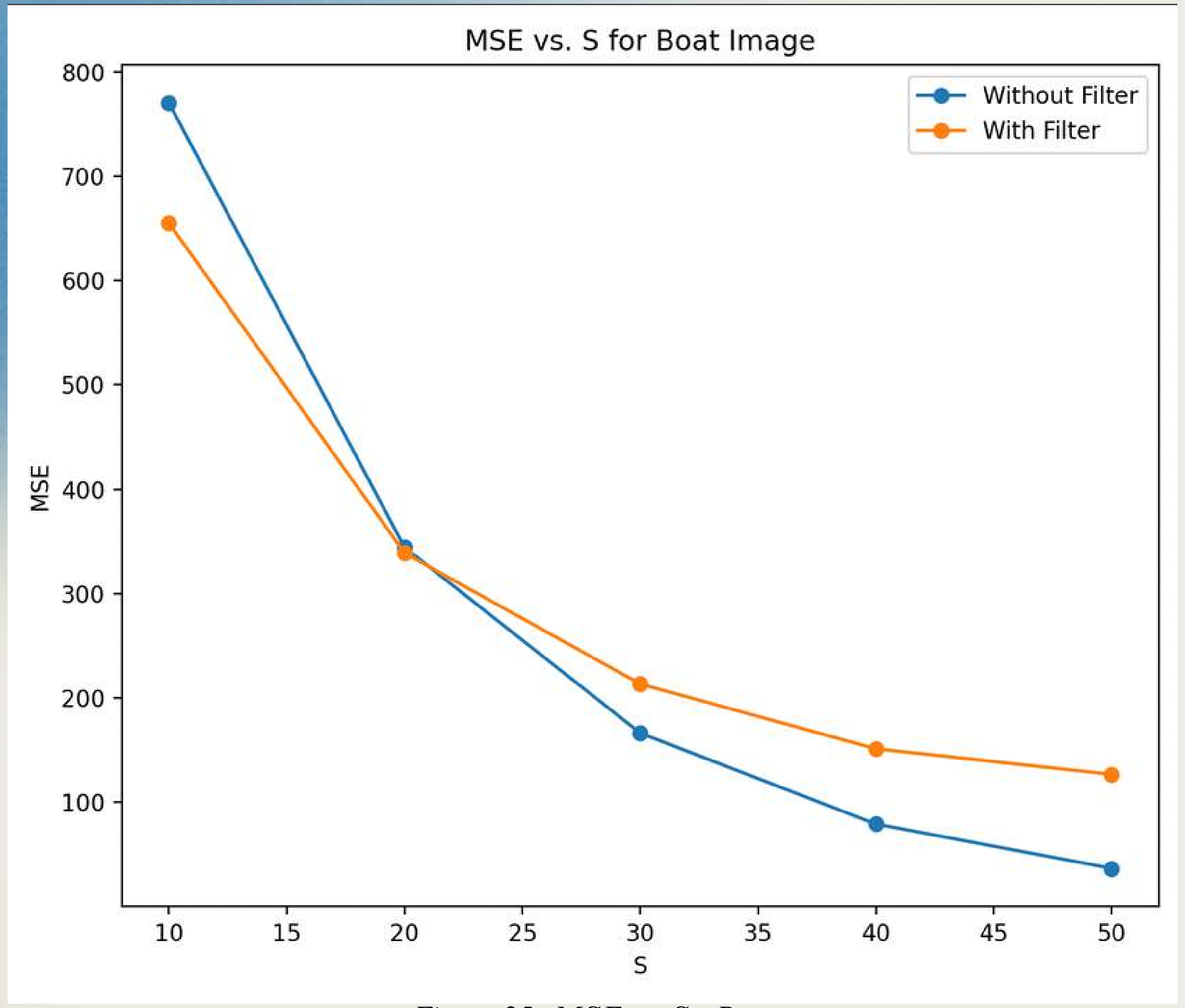


Figure 25: MSE vs. S - Boat

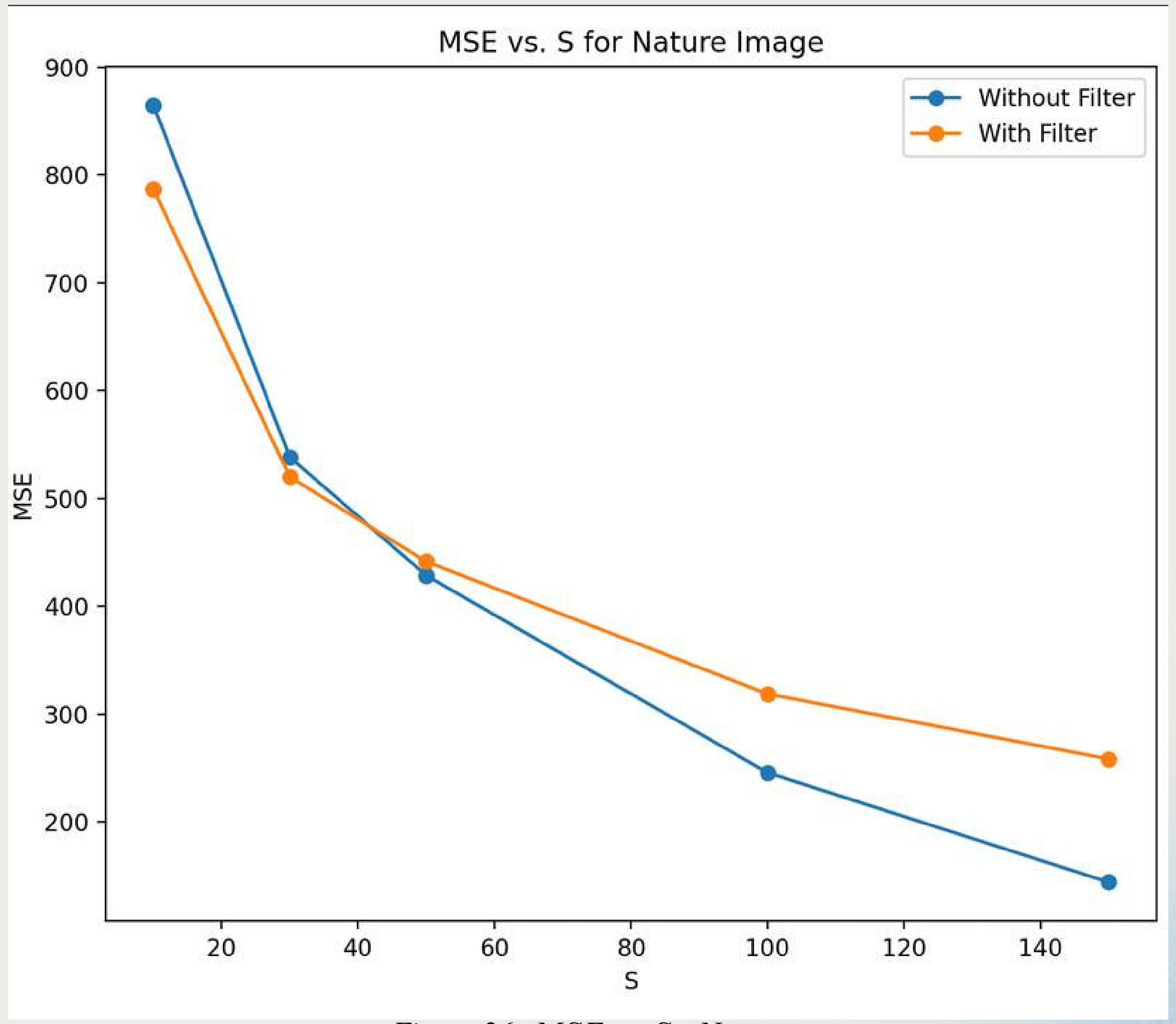


Figure 26: MSE vs. S - Nature

Image Reconstruction Results

Regularization Strength and MSE - Observations

The optimal regularization strength varies greatly on a block level within an image. The maps of optimal regularization parameters used on each block for the boat and nature images **highlight a distribution of values**. The distribution seems to follow a **normal distribution** of values, with most of the optimal values coming from the center of the logarithmic scale. For example, the map for the boat contains mostly light red and yellow, while the lighter and darker colors are less frequent. These maps show the **importance of using cross validation** to select the optimal hyperparameter.

The MSE results support our observation that reconstruction quality decreases as image corruption increases. The plots of **MSE vs. S** for both subject images show an **inverse relationship between *S* and MSE**. As mentioned previously, this makes sense since a lower degree of corruption means our models have more information to use for reconstruction. The MSE results also support our previous observation that median filtering only improves image quality up until a certain value of ***S***. The plots for both subject images show the MSE with filtering initially being lower than the MSE without filtering. The line describing MSE with filtering then cuts over the line describing MSE without filtering (***S*=*22*** for boat, ***S*=*42*** for nature).

Field Test

Results

Field Test - Corrupted

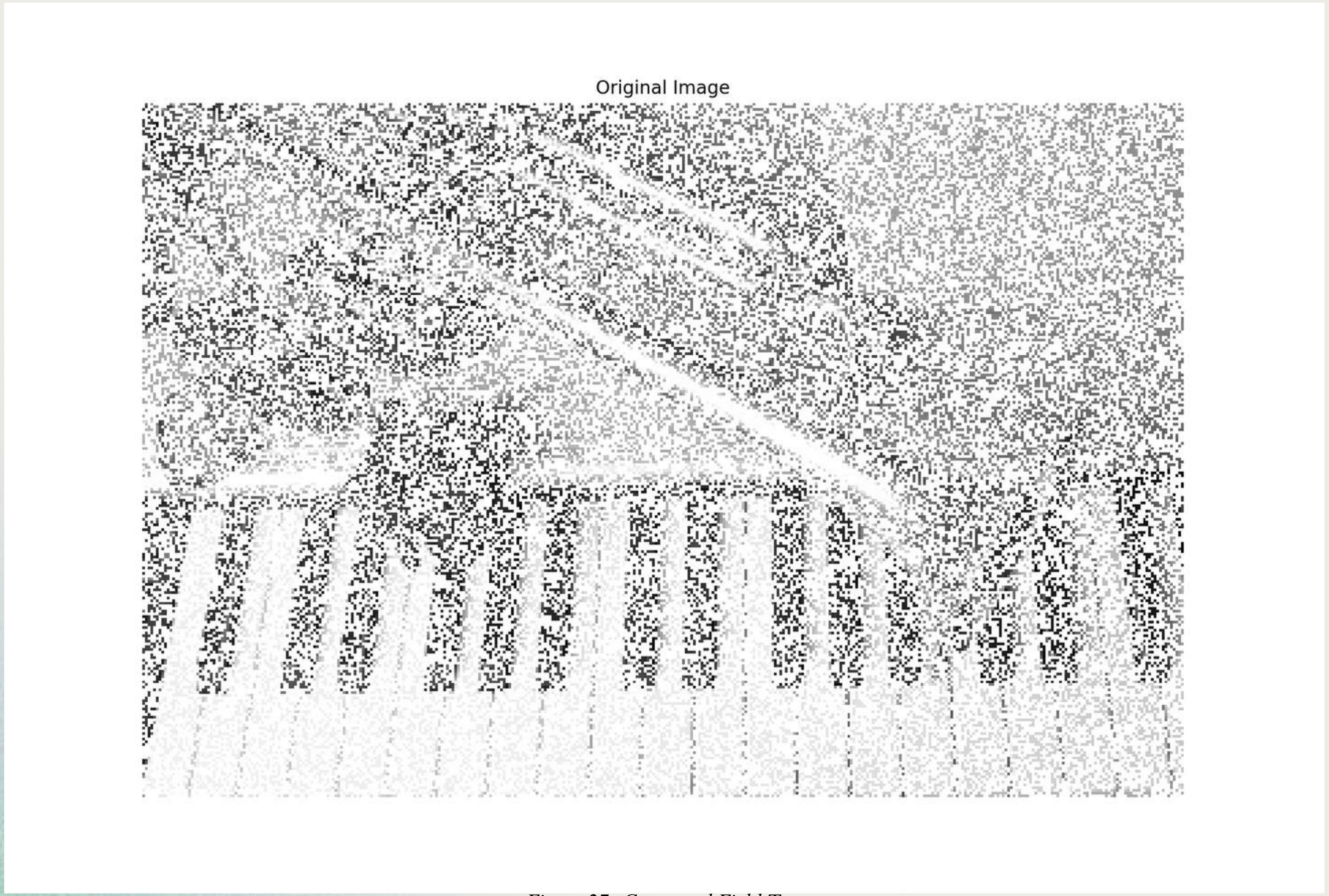


Figure 27: Corrupted Field Test

Results

Field Test - Reconstruction

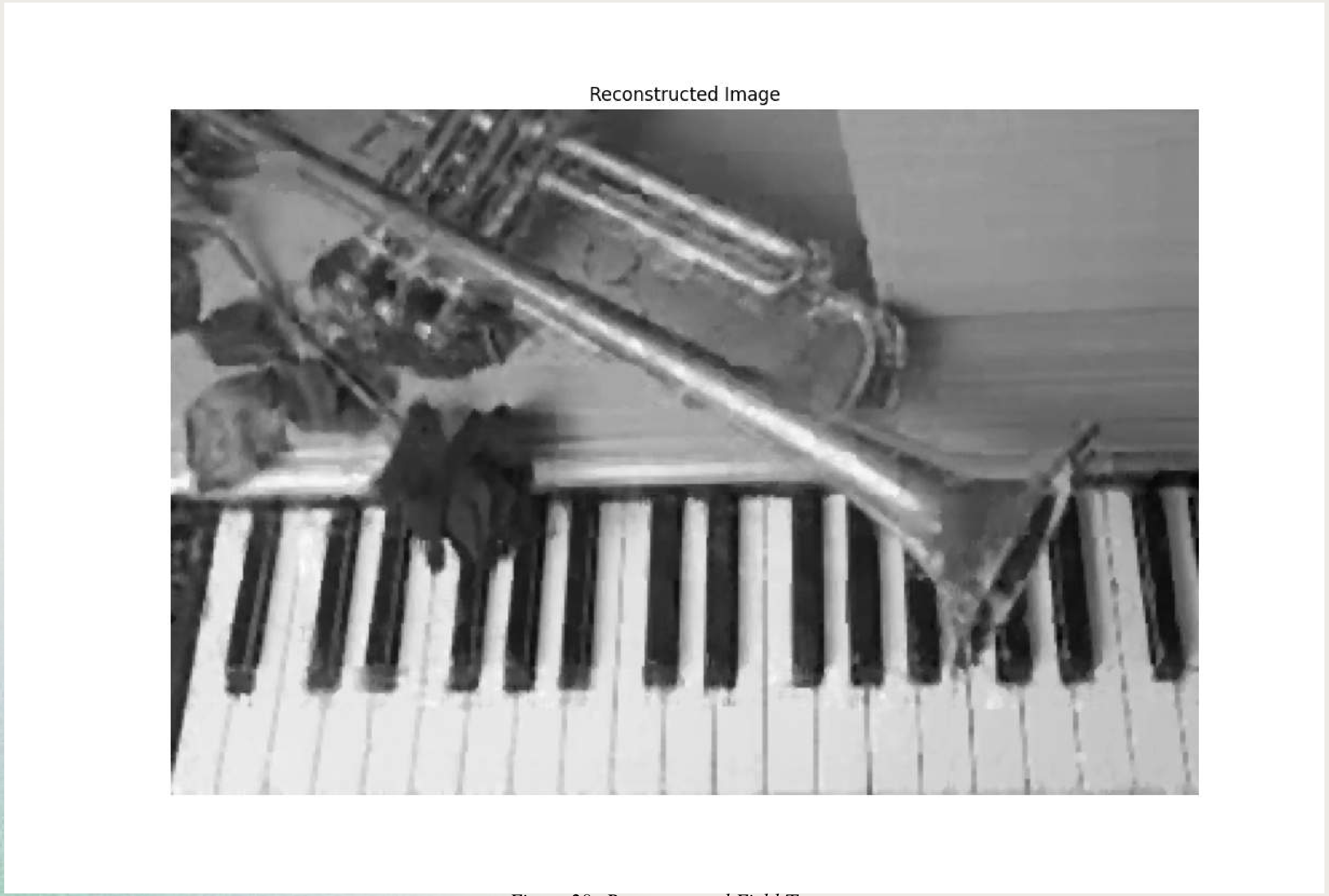


Figure 28: Reconstructed Field Test

Results

Field Test - Results

Again, the image reconstruction algorithm *(K=16, Median Filtering Size=3)* used almost feels like magic. The original corrupted image, which has 65% of its pixels missing, seems to only contain remnants of a piano. The recovered image displays a **beautiful reconstruction** of a full **piano** with a **rose** and **trumpet**. The reconstructed image contains proper lighting elements as well, contains shadows from the objects and reflections from light sources. This example emphasizes the power that image reconstruction has in order to **unlock new dimensions of insight** about our data.

Conclusions

Conclusions

About the Modelling Approach

How Regularized Regression Performs - In this project, we've shown how **regularized regression** can be used in a **compressed sensing framework** to predict missing pixels of an image. The regression approach used works very well and produces very impressive results that reveal new insight about our data. However, the applications of this approach isn't limited to image reconstruction. We can utilize the same modeling framework to reconstruct any corrupted signal. This highlights the power of a simple technique like regularized regression when used effectively (e.g. with cross validation and post-processing).

DCT Basis - The choice of basis is very important in the design of our algorithm. We require a choice of basis can represent our image **sparsely** and is **efficient** to calculate. The DCT is perfect for this task since it concentrates the most of the signal's energy into low frequency coefficients, making it suitable for sparse representation. The DCT is efficient as well because it contains the image signal into only a small number of coefficients, allowing for efficient representation.

Regularization Hyperparameter - The regularization strength controls the **trade-off** between fitting the training data and keeping the model regularized. The choice of our parameter relates to our understanding of the **bias-variance trade-off.** The more complex we make our model, the less generalizable it will be to unseen data. Regularization aims to minimize variance by encouraging sparsity in our weights. This means that a stronger regularization will lead to less overfitting. It is important to use **cross validation** to determine the optimal regularization strength to minimize the MSE of our reconstructions.

Conclusions

Lessons Learned

What would you do differently next time - Reflecting on what I learned, I found that the image reconstructions algorithm takes long to run. In the future, saving my results (plots, MSE, predictions, etc.) is more efficient as I can work with the results after running my algorithm once instead of re-running my algorithm to gain a new insight. I also found myself refactoring code into modules that made sense (CrossValidation, Image, BasisMatrix, etc.). This refactoring allowed me to implement new methods without hassle. This means that I should focus on writing cleaner code to begin with.

What worked really well, and you would do the same way next time - Implementing a LASSO model through ***sklearn's*** packages worked extremely well and I found no reason to implement these algorithms from scratch. Cross validation to determine the optimal hyperparamaters, regardless of the model, seems to lead to better results. I also think implementing post processing to my model's predictions is always beneficial. In this project, the post processing was simply a median filtering. But this post processing can take many forms depending on the context of the project. Finally, I think pre-allocating code is a very good practice. Storing my calculations for the basis vector matrix was crucial to making my reconstructions more efficient.

References

References

"If I have seen further, it is on standing on the shoulder of giants." -Issac Newton

Background Information

[1]Yaqub M, Jinchao F, Arshid K, Ahmed S, Zhang W, Nawaz MZ, Mahmood T. Deep Learning-Based Image Reconstruction for Different Medical Imaging Modalities. Comput Math Methods Med. 2022 Jun 16;2022:8750648. doi: 10.1155/2022/8750648. PMID: 35756423; PMCID: PMC9225884.

[2]The MathWorks Inc. (2024). Discrete Cosine Transform Documentation, Natick, Massachusetts: The MathWorks Inc. https://www.mathworks.com/help/images/discrete-cosine-transform.html

[3]F.H.P. Fitzek, F. Granelli, P. Seeling (Eds.), Computing in Communication Networks, Academic Press (2020), pp. 197-215, https://doi.org/10.1016/B978-0-12-820488-7.00023-2

[4]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[5]Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

[6]J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

[7]Tantum, Stacy. "Report Guidance: Compressed Sensing Image Recovery" Duke University. Feb 2024.

[8]J. Zhang, Y. Chen, H. Zhang and X. Shi, "Compressed Sensing for Astronomical Image Compression and Denoising," 2020 Chinese Control And Decision Conference (CCDC), Hefei, China, 2020, pp. 1162-1167, doi: 10.1109/CCDC49329.2020.9164087.

[9]Tantum, Stacy. "Compressed Sensing Image Recovery Lecture Slides" Duke University. Feb 2024.

[10]Marshall, Dave. "The Discrete Cosine Transform (DCT)." Cardiff University, users.cs.cf.ac.uk/dave/Multimedia/node231.html. Accessed 29 Feb. 2024.

[12]Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267–288.

[16]Fisher, R. "Median Filter." The University of Edinburgh, homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm. Accessed 29 Feb. 2024.

LASSO

[12]Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267–288.

Visualizations from Other Sources

[9]Tantum, Stacy. "Compressed Sensing Image Recovery Lecture Slides" Duke University. Feb 2024.

[10]Marshall, Dave. "The Discrete Cosine Transform (DCT)." Cardiff University, users.cs.cf.ac.uk/dave/Multimedia/node231.html. Accessed 29 Feb. 2024.

Toolboxes/Packages

[4]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[5]Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

[6]J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

Collaborations

Collaborations

“Great things in business are never done by one person” -Steve Jobs

I worked entirely independently for this project. I don’t know anyone else in the class and I found myself very invested in this project. In absence of other collaborators, I relied on the following resources:

- StackOverflow: debug code, look for specific functionality (e.g. plot a heatmap)
- Library Documentation: APIs for the libraries used in this project to understand functionality of specific methods
- ChatGPT: debug code (e.g. find the bug in model)