Compressed Sensing Image Recovery

ECE 580 Mini Project 1

Bryan Boyd

Duke University

Duke

Problem Description - Overview

- The goal of this project is to create an algorithm which restores an image with corrupted/missing pixels, by using regression to estimate the pixel color for black and white images.
- This is a very important issue to be solved for both engineering and society. For engineering, being able to recover missing pixels allows for less-intensive sensing, since one could only sample some percentage of total values and recover the rest using this algorithm. For society, many old photographs deteriorate over time, and to help these meaningful memories last for families, image recovery is necessary.
- Another reason why this is an interesting problem is because this presents us with the task of solving a linear system that is ill-poised or underdetermined. Namely, we have less constraints than necessary to solve our regression problem, since we have multiple missing pixels. To circumvent this issue, other constraints must be imposed such as smoothness or sparsity, which will be discussed later in this report.

Original

Corrupted

Recovered

Orange pixels represent corrupted pixels.

Duke

Problem Description - Subject Images

- To test how well the recovery algorithm is working, we used two different subject images named fishing_boat.bmp and nature.bmp. These are two known images, which were corrupted and recovered throughout the design phase of the algorithm to test different parts of the algorithm. (Note in the images shown below nature.bmp has been shrunk to 40% of its true size to fit on the slide).
- These two images are suitable for this project, since they have very diverse spatial frequency of pixel colors and have a variety of quality and subject matter. As we can see in both images, there are areas that have many different pixel colors, areas that are all the same, some where the colors somewhat blend and others where the neighboring colors are all distinct.
- The images themselves are also different shapes and sizes (fishing_boat.bmp is 192 x 200 pixels whereas nature.bmp is 640 x 512 pixels), and the fishing boat image has more details in the foreground while the nature image has more details in the background. The fishing_boat.bmp image also includes some finer ropes which are randomly distributed throughout the image, whereas nature.bmp has a lot more clusters with the trees and mountains. These two images themselves cover a lot of the different traits seen in a large array of images, making them two perfect choices for our training set.

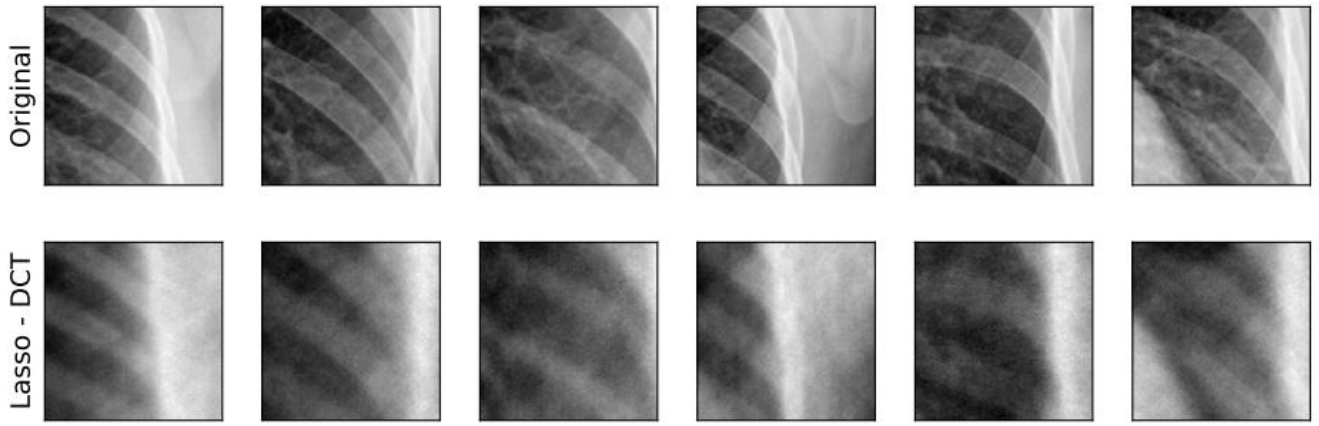
fishing_boat.bmp

nature.bmp

Duke

Other Potential Application Areas

- To solve the reconstruction problem, the mathematical approach we take involves using a Discrete Cosine Transform to convert the image into essentially a sparse vector of weights, and then applying L1 regularization (LASSO) regression, which will be explained further in the next section.
- This process has many application areas outside of the realm of recovering natural images. In imaging there are multiple other examples such as:
  - Medical Imaging[1] – Less measurements needed, less cost and time for scans.
  - Estimating hydrological spatial properties[2] – In some environments it is very difficult to get a large amount of data. Using only a few data points the spatial property of a given region can be estimated.
- In addition to its usefulness in imaging techniques for multiple fields, this method of reconstruction can be helpful for reconstructing audio signals which may have been corrupted or damaged or even recorded quite a while ago, so the speech needs to be enhanced.[3]



X-Ray images reconstructed using DCT and LASSO.[1]

Duke

Mathematical Formulation – Discrete Cosine Transform

- The Discrete Cosine Transform (DCT) is a mathematical transformation which uses a basis vector matrix to convert pixel values in the spatial domain to coefficients in the spatial frequency domain.
- When a DCT is applied to a natural image, many of the coefficients go to zero, making it a sparse representation of the image which is desirable for efficient image compression.
- We can use these characteristics of the DCT for image recovery, by breaking our larger image into smaller N x N chips and applying the DCT to the pixels with known intensities to approximate the coefficients using regularized regression. Then these approximated coefficients can be used on the missing pixels to approximate their intensities. These chips can all be combined to form the final reconstructed image.
- To create a DCT basis vector matrix, we combine the basis vectors for each spatial frequency pair (u, v) together to form a matrix. To create these basis vectors, we first create a basis chip. Each value in the basis chip is determined based on the spatial location, spatial frequency and two coefficients as shown by the equation below. These basis chips are transformed into basis vectors by stacking each column on top of each other to create a single column vector. Variable pair (x, y) refers to the spatial location of the specific pixel within the chip in local coordinates.

$$Basis\ Chip\ (u,v,x,y) = \alpha_u\,\beta_v\,cos\frac{\pi\,(2x-1)(u-1)}{2N}\,cos\frac{\pi\,(2y-1)(v-1)}{2N}$$

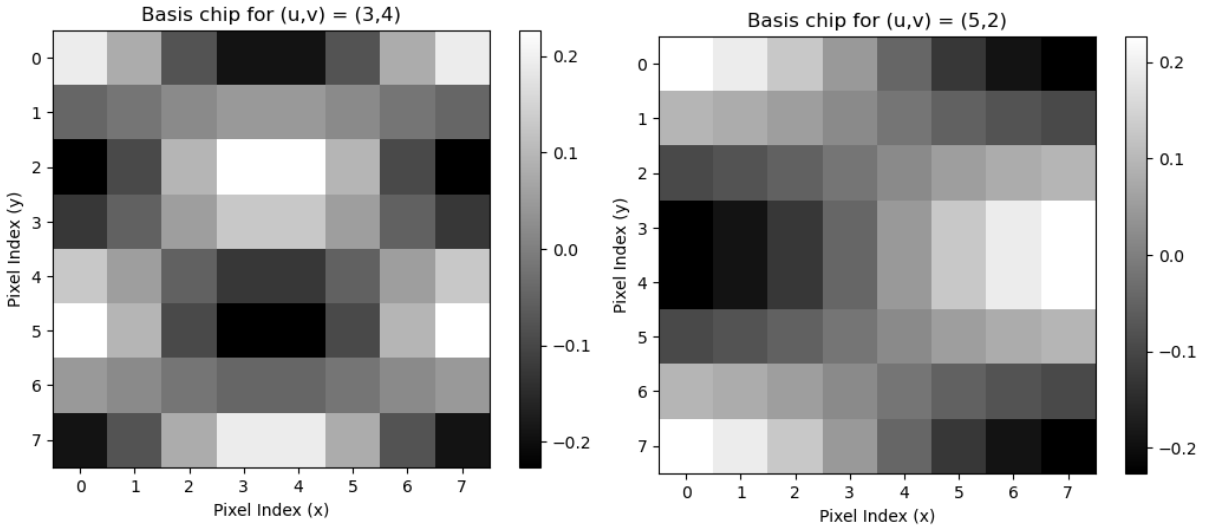
$$\alpha_u = \begin{cases} \sqrt{1/N}\ (u=1) \\ \sqrt{2/N}\ (2 \le u \le N) \end{cases} \qquad \beta_v = \begin{cases} \sqrt{1/N}\ (v=1) \\ \sqrt{2/N}\ (2 \le v \le N) \end{cases}$$

Duke

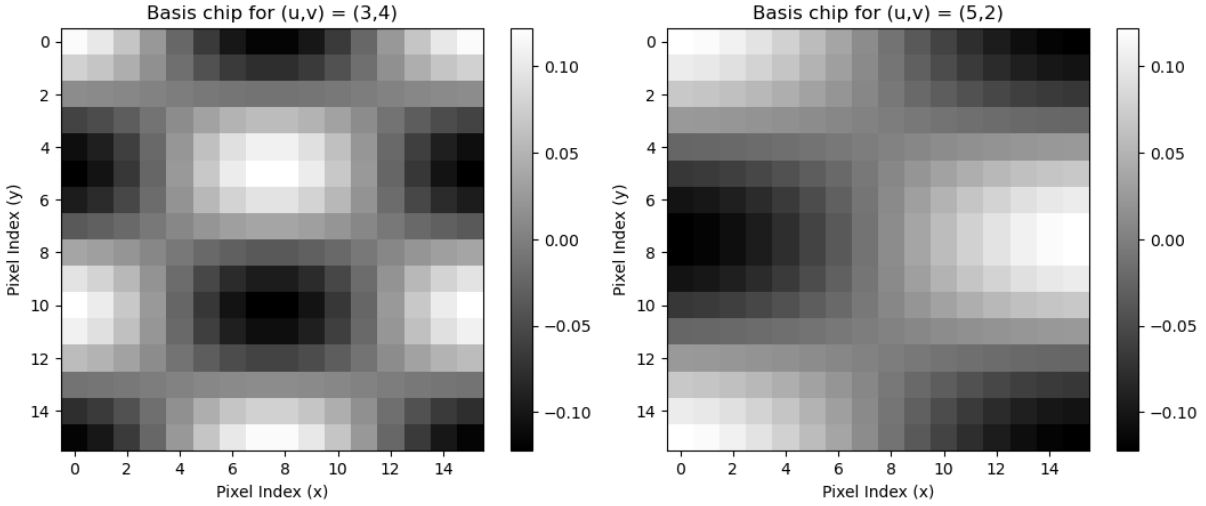
Mathematical Formulation – Discrete Cosine Transform

- Two examples of basis chips using the formula given on the previous slide are shown here for both 8x8 and 16x16 chip sizes.
- As can be seen, even though the chip sizes are different, the plot of intensities look very similar to each other.
- To convert these chips into vector, each column of the chip starting from left to right are stacked on top of each other to create a basis vector.

8x8 Basis Chip



16x16 Basis Chip

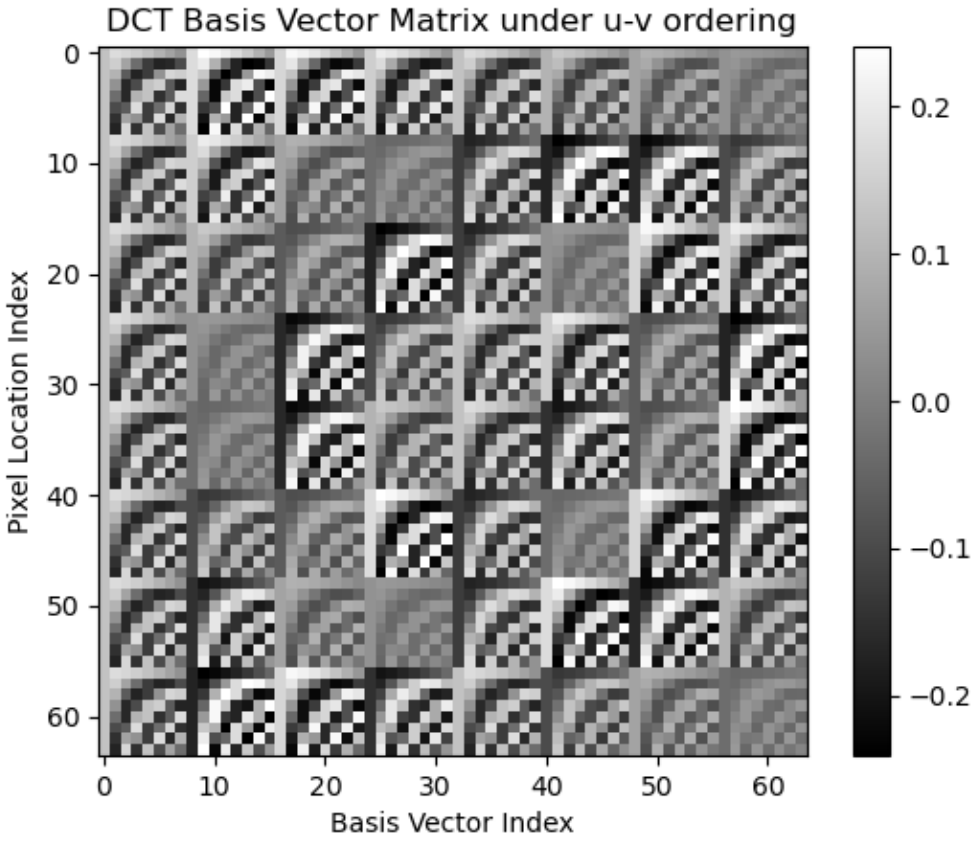


Duke

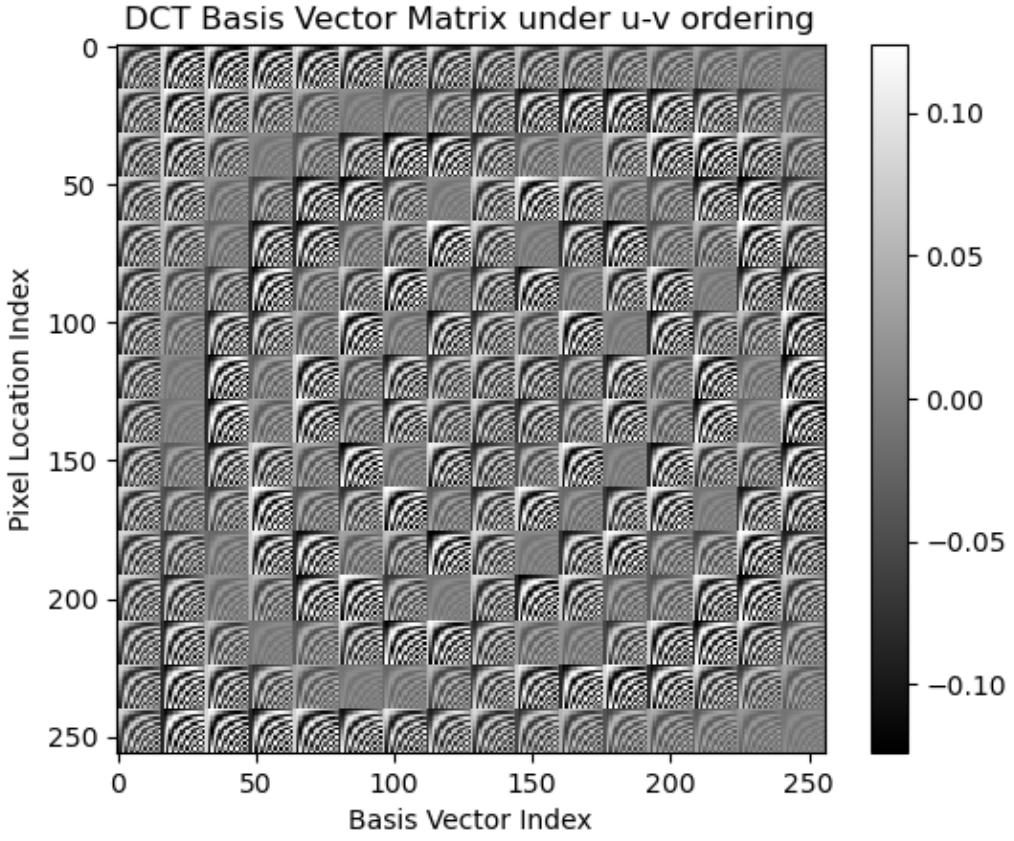
Mathematical Formulation – Discrete Cosine Transform

- Below are the two DCT basis vector matrices that were used for the reconstruction of the two subject images, which were based on 8x8 and 16x16 chips. Each column of the matrix represents a basis vector, which in turn is the vectorization of one of the N^2 basis chips.

8x8 Basis Chip



16x16 Basis Chip



Duke

Mathematical Formulation – Regression Set Up

- For a standard image, if we take a vector **b** (K x 1) which corresponds to the intensities of all of the pixels, we can get the weights **w*** (K x 1) for the BVT coefficients by using the DCT basis vector matrix **A** (K x K) as shown below:

$$\boldsymbol{Aw} = \boldsymbol{b}$$

- This implies that to solve for the appropriate weights of the BVT coefficients for a specific image we would need to solve

$$\boldsymbol{w}^* = \boldsymbol{A}^{-1}\boldsymbol{b}$$

- This works well for uncorrupted images, but since we are missing pixels, the dimension of vector **b** is decreased to only the number of known pixels. This can be referred to as a new vector **d** (S x 1). Likewise, we can remove the rows of matrix **A** corresponding to the corrupted pixels, resulting in a new matrix **B** (S x K). This means if we substitute in **d** for **b**, and **B** for **A** we can no longer solve the previously stated equation due to the system being underdetermined.
- We can however rewrite the problem as the following:

$$\min_{\boldsymbol{w}} \boldsymbol{Bw} - \boldsymbol{d}$$

- This theoretically would give us a solution w to our problem which approximates w*. Generally, to make this a convex optimization problem and thus more easily solvable, it would be turned into a least squares problem of the form:

$$\min_{\boldsymbol{w}} \|\boldsymbol{Bw} - \boldsymbol{d}\|_2^{\,2}$$

Duke

Mathematical Formulation – LASSO[4]

- To enhance our solution, we use regularized regression, namely LASSO (Least Absolute Shrinkage and Selection Operator). This method involves solving a least squares problem but includes a term to minimize the sum of all the magnitudes of the components of the vector by using the L-1 norm.
- The LASSO problem can be set up as follows: Matrix **A** represents the total basis vector matrix solved for using the DCT BVM. Then, vector **d** represents the intensities of all the non-corrupted pixels. We can create a new matrix **B** which contains the rows which corresponds to only the sensed pixels. Then we can solve for **w**, the coefficients in the DCT basis using the following minimization formula:

$$\min_{\boldsymbol{w}} \|\boldsymbol{Bw} - \boldsymbol{d}\|_2^{\,2} + \lambda\|\boldsymbol{w}\|_1$$

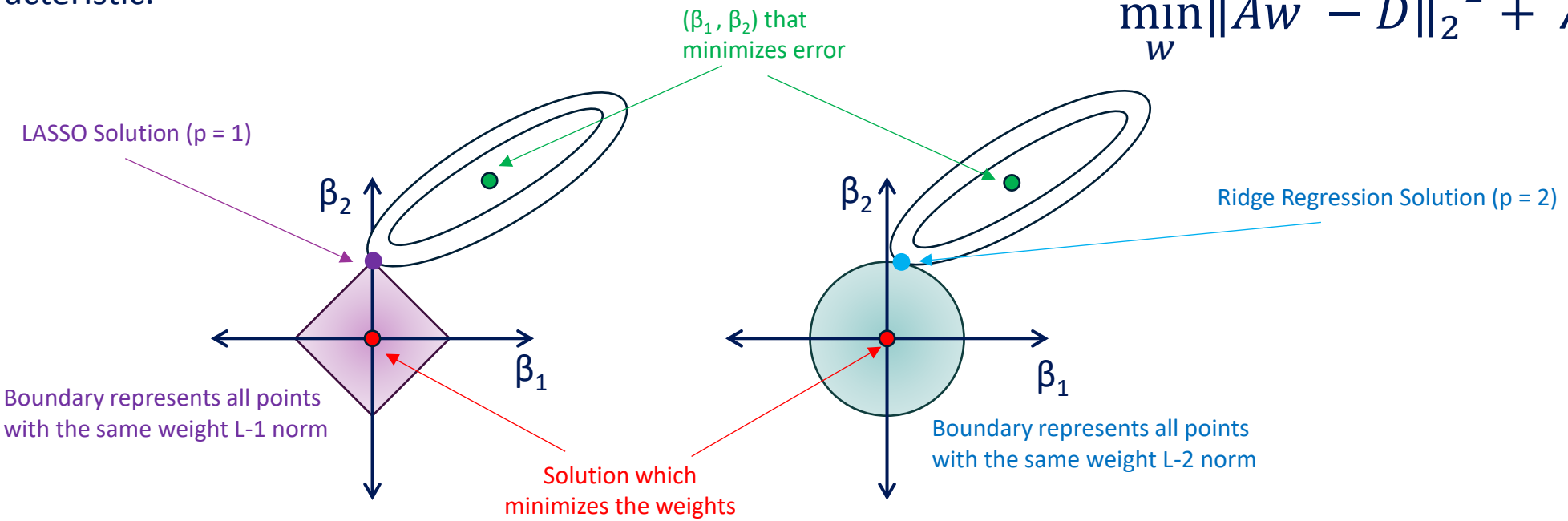
- Here the L-1 norm is used along with the regularization parameter λ to drive many of the coefficients of **w** towards zero. This is ideal since we know that the real image is sparce in the DCT domain, so the reconstruction should be as well If the L-2 norm was used instead here, it would likely make many of the **w** coefficients small, but non-zero. Notice if λ = 0, this is simply least squares regression. The determined **w** vector is then used to determine the missing pixel values.
- One note here is that the basis vector matrix (**B**)'s first column is already a constant not necessarily equal to 1. LASSO (using scikit.learn) expects the constant value to be 1 since for linear regression for example, w_0 would equal to 1 as 1 + ax would be the expected model to fit. This is not the case for our model however, so after solving for the weights, we must manually set w_0 to be equal to the solved model intercept divided by the constant value in the BVM to eliminate bias, effectively normalizing the first entry of the weights.

Duke

Mathematical Formulation – Why L-1 Norm?

- On the last slide it was mentioned that the regularization parameter is tied to the L-1 norm since it drives more of the coefficients to zero compared to the L-2 norm. This slide acts as a 2D visualization of this characteristic.

$$\min_{w} \|Aw - D\|_2^2 + \lambda \|w\|_p$$

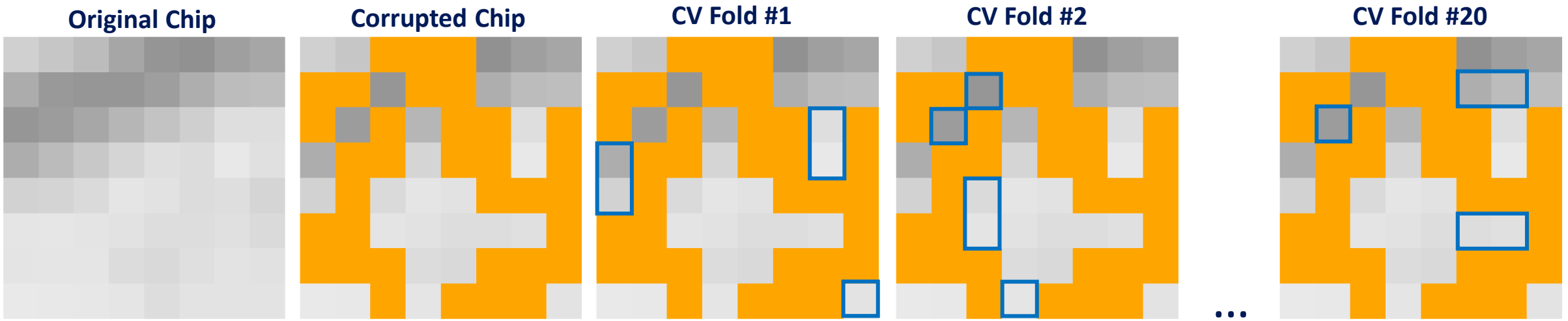


- As we can see from this example, using LASSO (regularization with L-1 norm), you are more likely to find a solution that pushes more of the weights to zero compared to something like ridge regression (regularization with L-2 norm), where you are likely to minimize values, but not force them towards zero.

Duke

Mathematical Formulation – Random Subsets Cross Validation

- To determine what the appropriate regularization parameter (λ) should be for each chip, 20-fold cross validation is performed for candidate values of λ: 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 0.1, 1, 10, 100, 1000, 10000.
- The way a general K-fold cross validation works for this problem is that out of all the non-corrupted pixels, a certain amount are reserved as a test subset, and the model is trained on only the remaining training subset. Then the error is measured between the true test set pixel values and the estimated values for all λ values. The lambda that results in the lowest average mean squared error amongst all 20 folds is chosen for that chip's regression.
- As shown below, each of the 20 different folds uses a unique test subset, where the number of pixels in the subset is equal to floor(S/6) where S is the number of non-corrupted pixels.



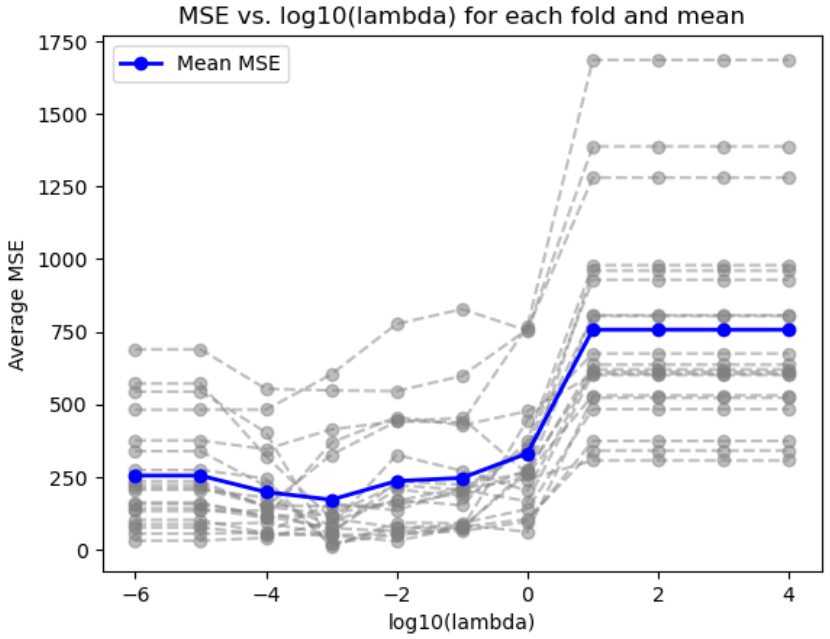
Orange pixels represent corrupted pixels. Pixels highlighted blue indicate pixels that are used for testing, whereas all the remaining non-corrupted pixels are used for training.

Duke

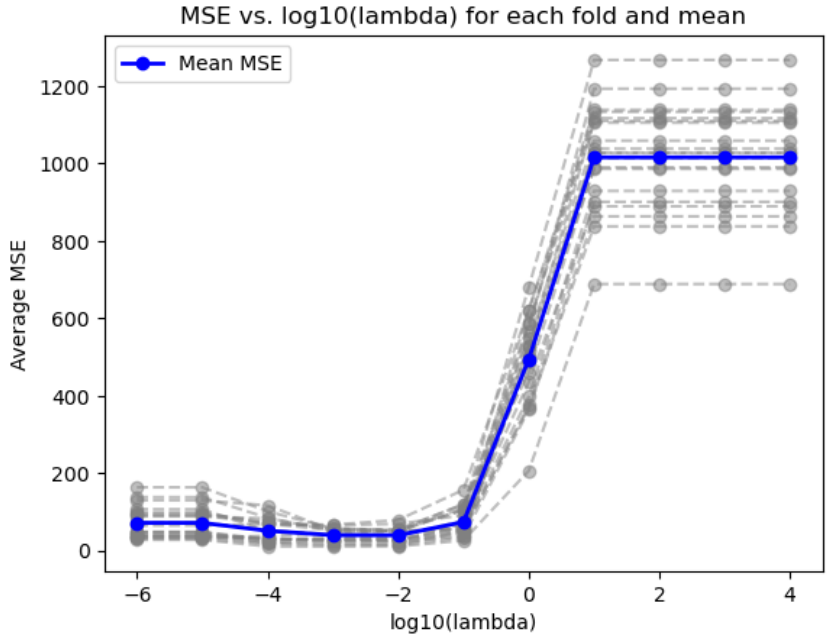
Mathematical Formulation – Random Subsets Cross Validation

- A sample from each of the two subject images of the 20-fold random subsets cross validation process for selecting a regularization parameter for a particular chip is shown below. The fishing_boat.bmp chip is 8x8 whereas the nature.bmp chip is 16x16.
- As seem in the plots, there is a large variance between all the individual folds, so by using the average amongst 20 random folds, we can get a better picture as to what is most likely to be the best regularization parameter to minimize the error in the reconstructed chip. The minimizer of the average mean squared error is selected to be the parameter value for the reconstruction of the entire chip. This process is repeated for every chip.

fishing_boat.bmp



nature.bmp

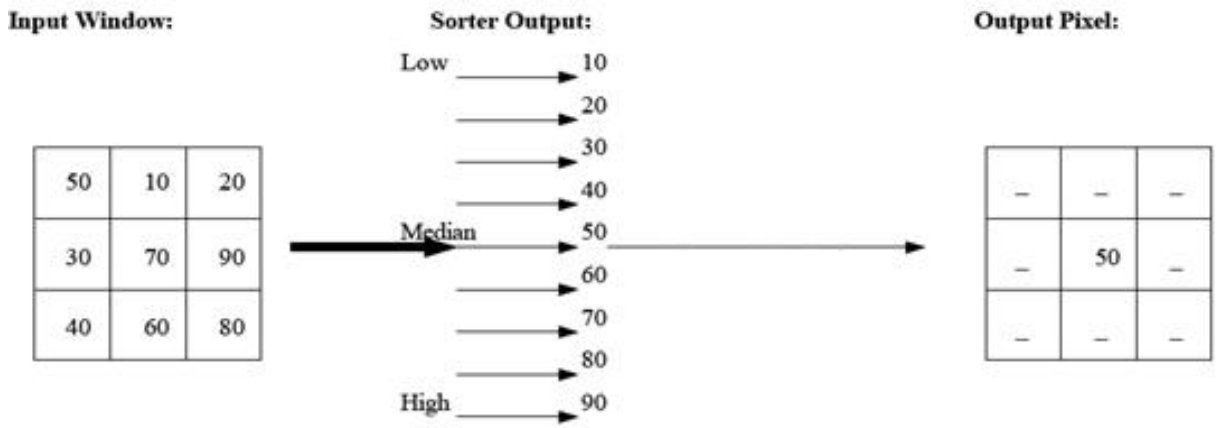


Duke

The points on the blue line represent the average mean squared error across all 20 folds for a given regularization parameter λ. Each of the faded grey lines represents one of the twenty cross validation folds.

Mathematical Formulation – Median Filtering

- When an image is reconstructed chip by chip, there is a likelihood of impulsive noise due to poor estimation of a pixel value, which could be due to which pixels were corrupted. A way to help fix this issue would be to apply a filter over the reconstructed image to try to remove some of that noise.
- An ideal filter for this issue would be a median filter, which as the name suggests takes the median over all pixel values in a given range around a center pixel and assigns that center pixel a new value which is the median value. Essentially this filter makes sure that pixel intensities are not too far away from any of its neighboring pixel values.
- In this instance, a median filter would be a better choice than a frequency dependent filter. For example, a low pass filter would likely not preserve edges and smooth the entire image rather than just the impulsive noise.[5] On the other hand, a high pass filter would not work either since it would sharpen the edges and make the noise more pronounced than it already is, thus a median filter is the ideal filter for this task.[5]



Example diagram showing how a median filter works for a specific pixel.[6]

Duke

Simulations – fishing_boat.bmp – Corrupted Images

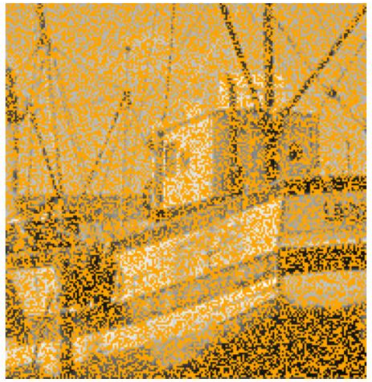
- To create a corrupted image, a specific number of pixels were removed from the original fishing_boat.bmp image. Since the image is broken down into 8 x 8 chips for reconstruction, the number of pixels removed is also per 8 x 8 chip. For these simulations, there are five different levels of corruption : 14, 24, 34, 44, and 54 corrupted pixels per chip. In this report, rather than referring to the number of corrupted pixels, we refer to the number of sensed pixels (S) which is just N * N - # of corrupted pixels. These pixels are randomly selected for each of the chips in the image for each simulation. Examples of these corrupted images are shown below.

Original

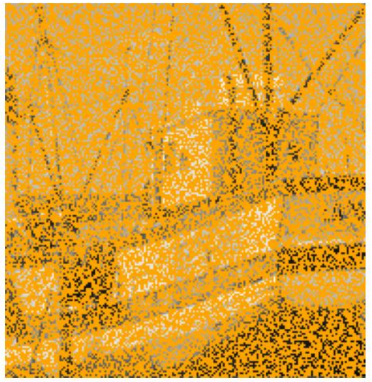
Corrupted (S = 50)

Corrupted (S = 40)

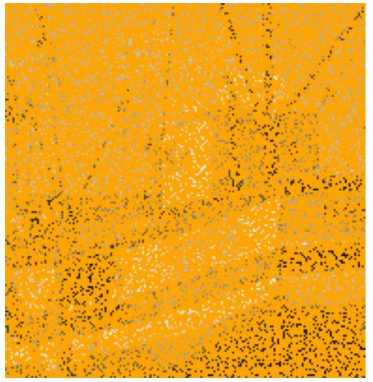
Corrupted (S = 30)



Corrupted (S = 20)



Corrupted (S = 10)



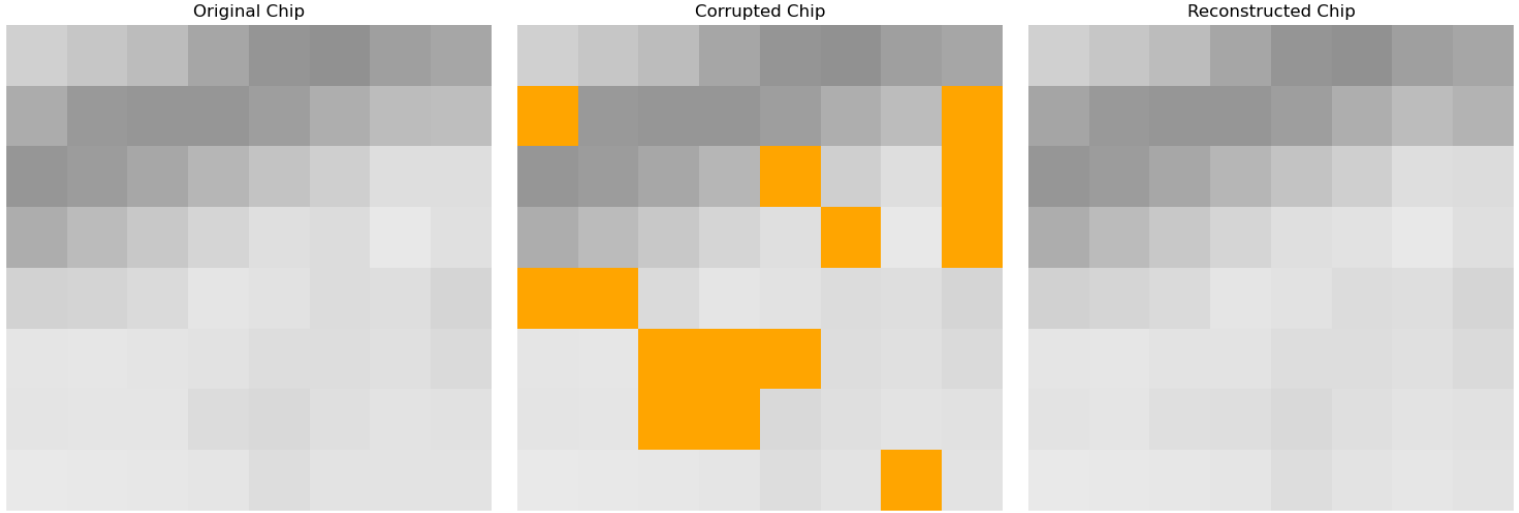
Duke

Orange pixels represented corrupted/missing pixels

Simulations – fishing_boat.bmp – Single 8x8 Pixel Chip

- The following slides show examples of the reconstruction process for a single 8 x 8 chip (this chip's top left corner is located at (64,88). The process is shown for 5 different amounts of corrupted pixels, indicated by the number of sensed pixels (non-corrupted).
- As described earlier, each chip goes through a DCT transformation, a 20-Fold random subsets cross validation to determine the best regularization constant, then LASSO regression to determine the missing pixel intensities. There is no median filtering being done to each individual chip. Each time the pixels which are corrupted are randomized.
- While it is hard to tell with an individual chip, it is sufficient to say that as the number of sensed pixels decreases, the quality of the reconstructed chip decreases, which makes sense because there is less data for the model to train on.

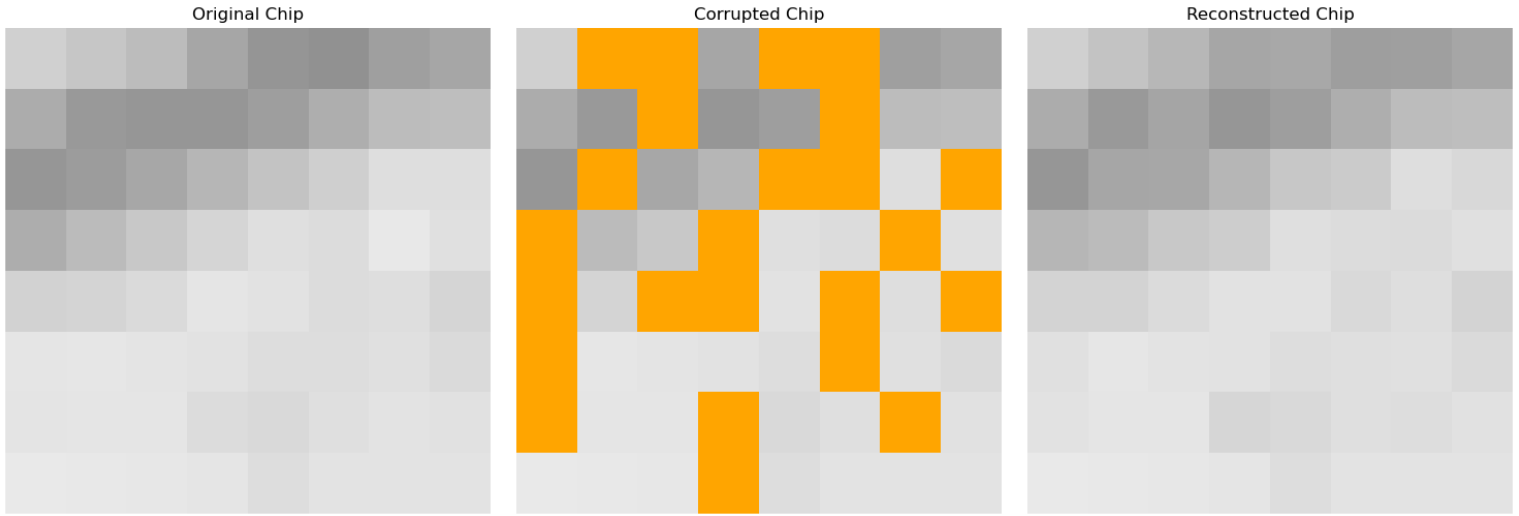
50 Sensed Pixels



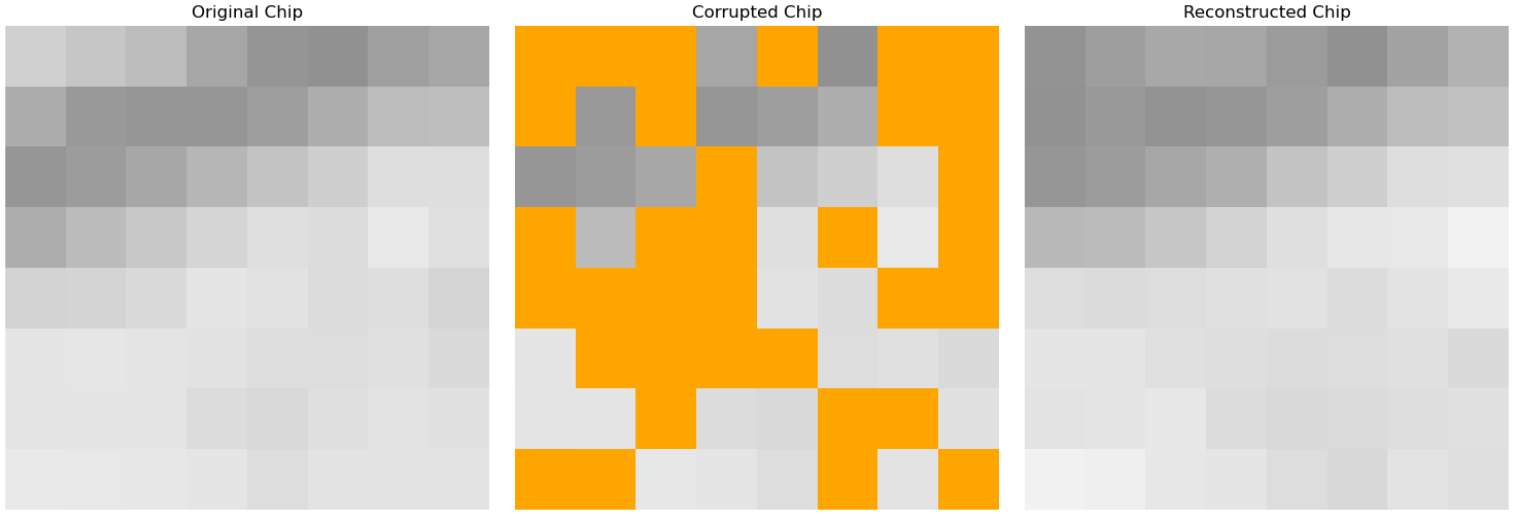
Duke

Simulations – fishing_boat.bmp – Single 8x8 Pixel Chip

40 Sensed Pixels



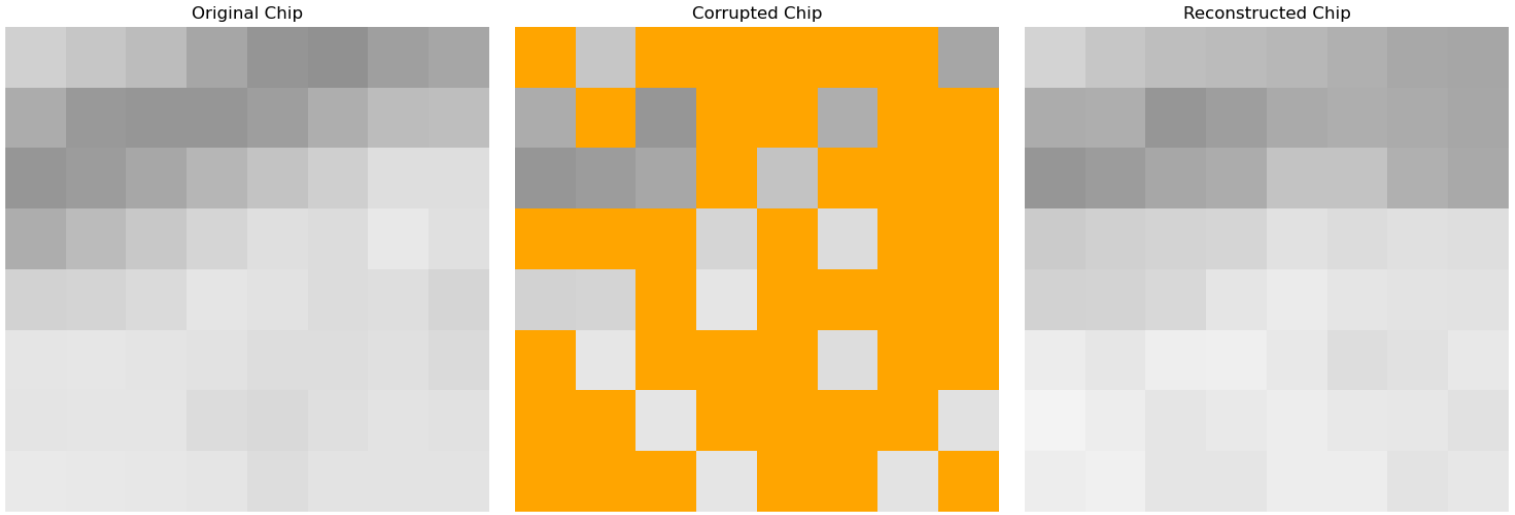
30 Sensed Pixels



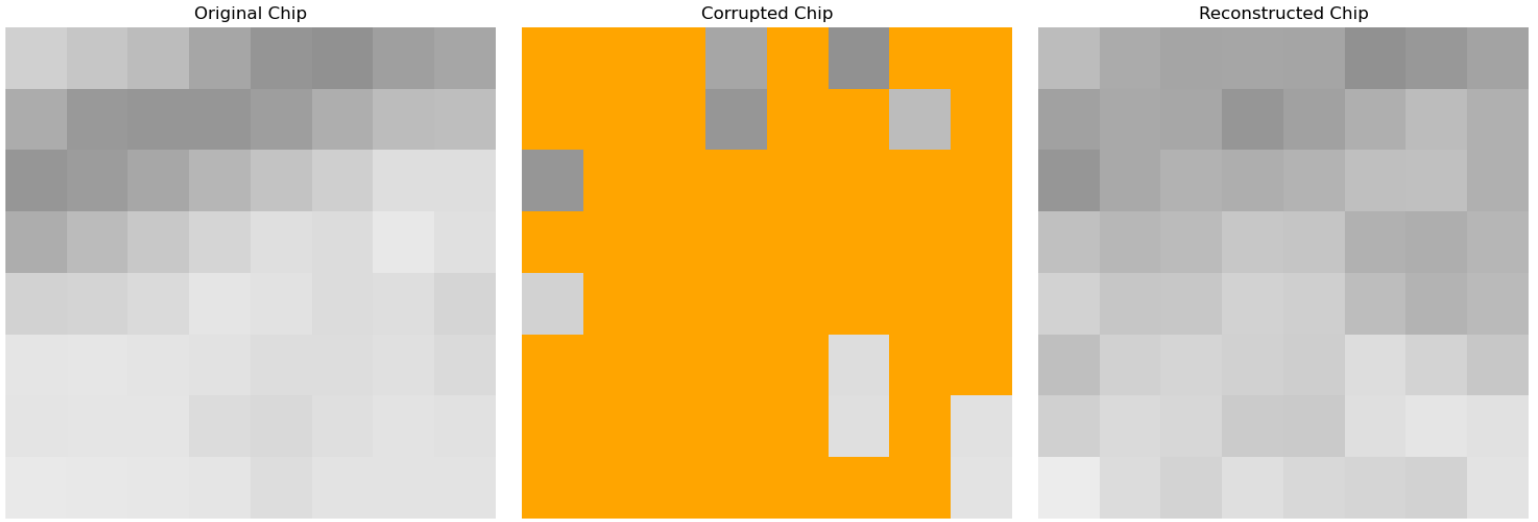
Duke

Simulations – fishing_boat.bmp – Single 8x8 Pixel Chip

20 Sensed Pixels



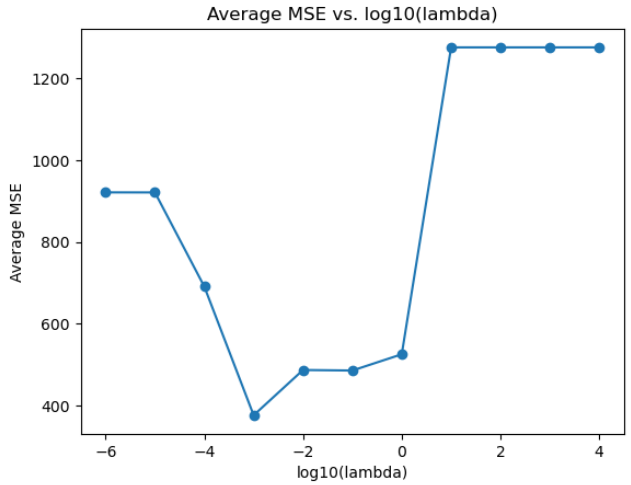
10 Sensed Pixels



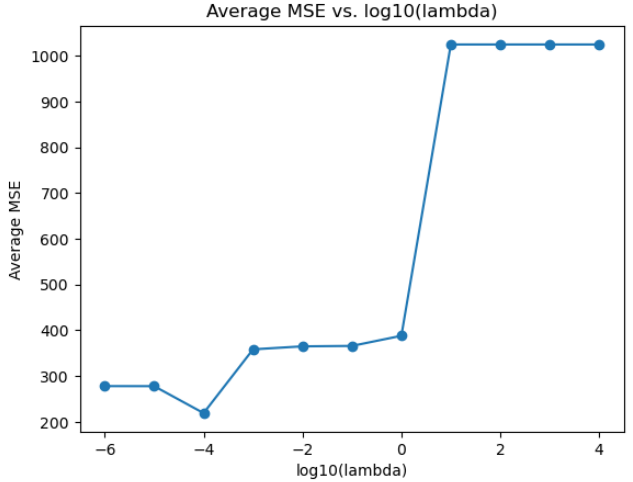
Duke

Simulations – fishing_boat.bmp – Single Chip Regularization Parameters

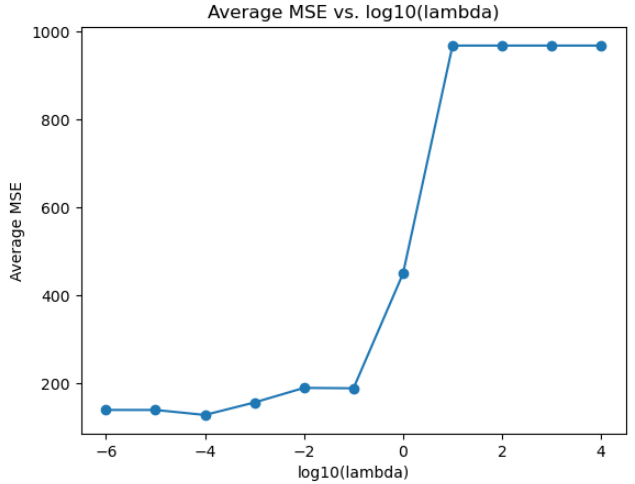
10 Sensed Pixels



20 Sensed Pixels

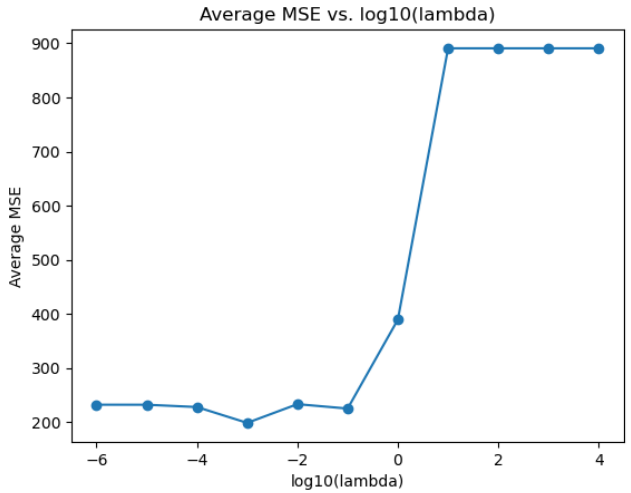


30 Sensed Pixels

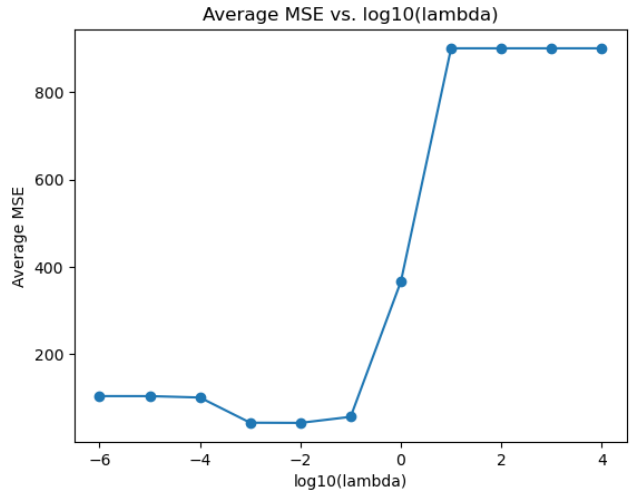


These plots show the average mean squared error across all folds as a function of the regularization parameter λ on a log scale. The selected λ is the minimizer of this function. These plots change with number of sensed pixels, which shows its sensitivity to the training data, particularly we see a more jagged plot for a lower number of sensed pixels. We also see the minimizer alternating between 10^{-3} and 10^{-4} which shows the variance depending on which pixels are being sensed.

40 Sensed Pixels



50 Sensed Pixels

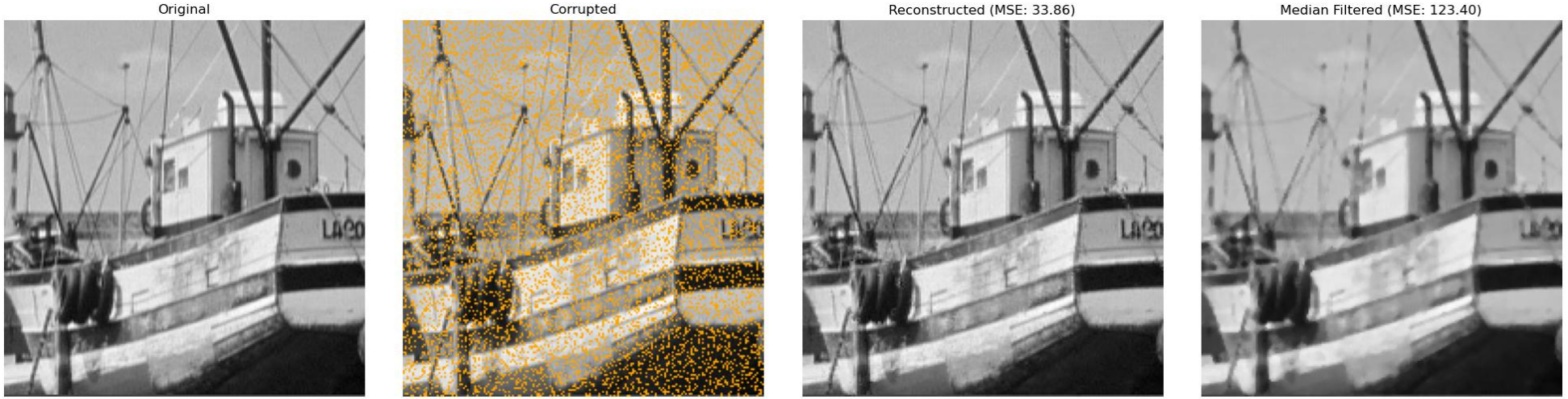


Duke

Simulations – fishing_boat.bmp – Full Image Recovery

- The following slides show examples of the reconstruction process for the entire fishing_boat.bmp image. The process is shown for 5 different amounts of corrupted pixels, indicated by the number of sensed pixels (non-corrupted).
- Here, each chip goes through a DCT transformation, a 20-Fold random subsets cross validation to determine the best regularization constant, then LASSO regression to determine the missing pixel intensities. Then after all chips are restored, the median filter is applied across the entire image. The mean squared error of the pixel intensities for the initial reconstruction, and reconstruction after median filtering compared to the original are shown in the plot titles. Note that the MSE is taken over ALL pixels (even ones that are not corrupted).
- The more corrupted pixels there are, the poorer the reconstruction quality is, which is particularly noticeable at lower numbers of sensed pixels. Additionally, the median filter helps to smooth out these lower quality images by removing some of the staticky salt and pepper noise that occurs in the original reconstructed images. This is less noticeable in the higher quality reconstructed images, where the median filtering seems to blur parts of the image.

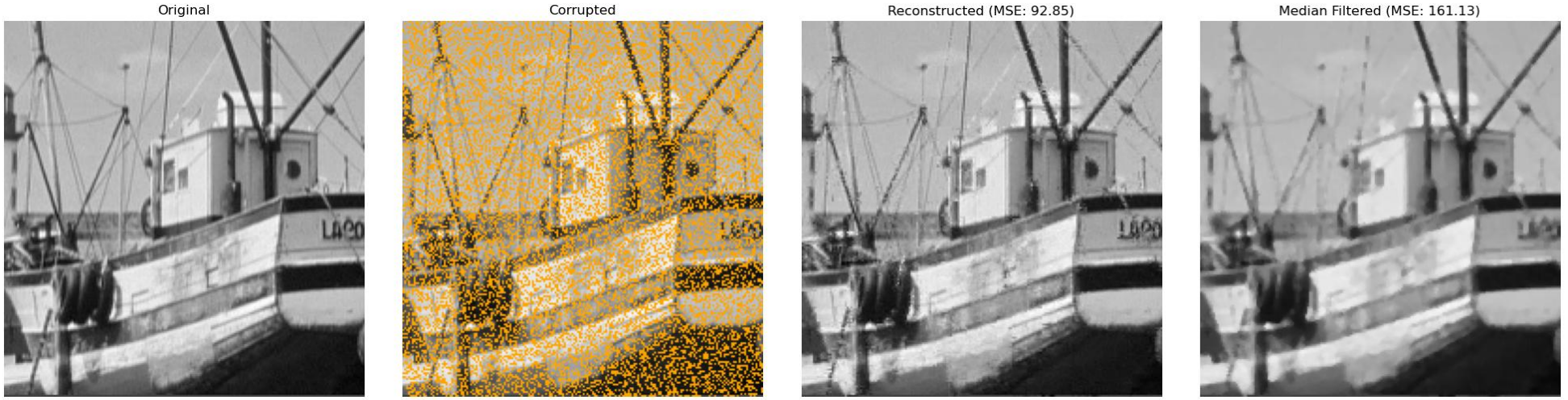
50 Sensed Pixels Per 8x8 Chip



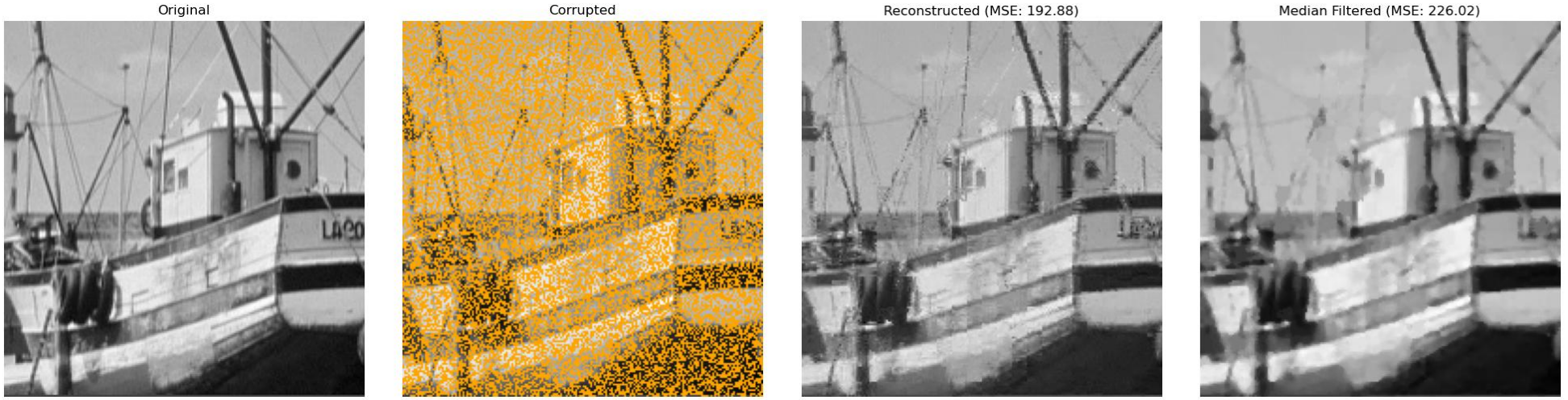
Duke

Simulations – fishing_boat.bmp – Full Image Recovery

40 Sensed Pixels Per 8x8 Chip



30 Sensed Pixels Per 8x8 Chip

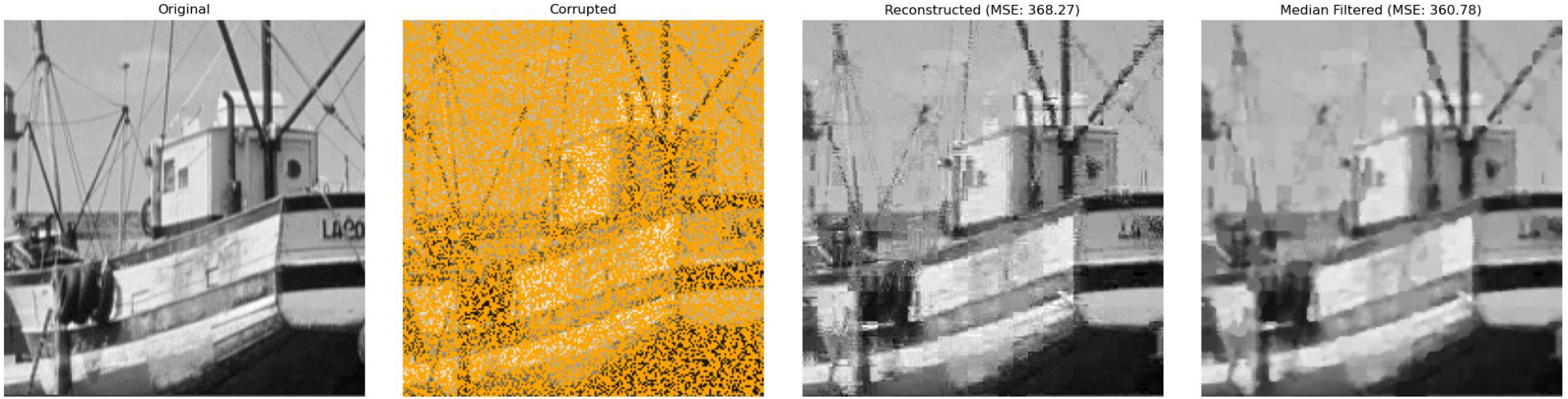


Duke

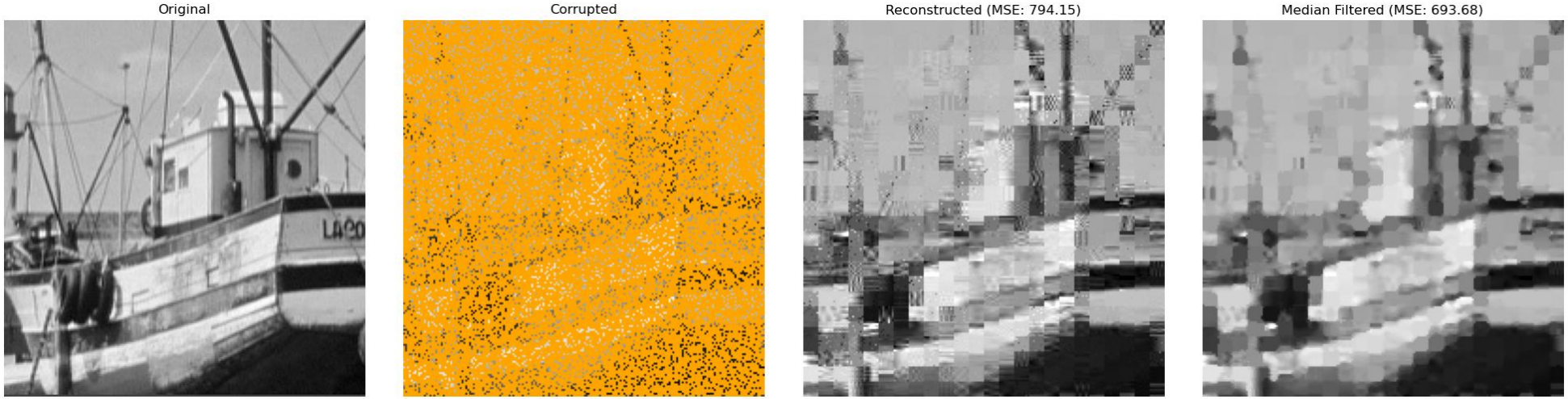
These two reconstructions generally look good, but we start to lose definition on areas such as the ropes and the word on the boat, which are more complex than the body of the boat or the background.

Simulations – fishing_boat.bmp – Full Image Recovery

20 Sensed Pixels Per 8x8 Chip



10 Sensed Pixels Per 8x8 Chip

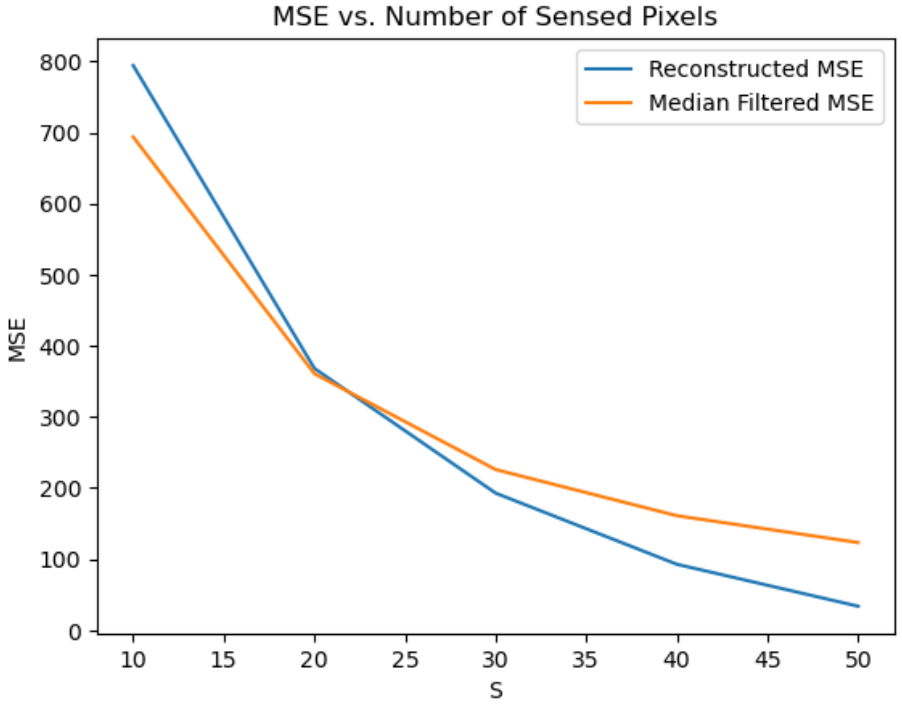


Duke

These two reconstructions do not look as good, and we can start seeing more distinct blocks, even though the median filtering helps this a bit. This is due to the number of corrupted pixels per chip being too high.

Simulations – fishing_boat.bmp – Full Image Recovery

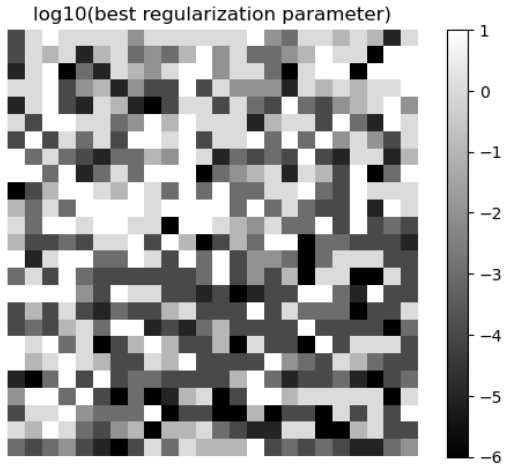
- By plotting the mean squared error of the image versus the number of sensed pixels can see that the mean squared error tends to decrease as the number of corrupter pixels decreases (S increases). This makes sense since with less uncorrupted pixels we have more data to train our regularized regression model, so it is more likely that our model is correct. Also, we are taking the MSE over ALL pixels, not just the initially corrupted ones, thus we include more pixels with zero error when we have less uncorrupted pixels, since we are not replicated the non-corrupted pixels when we apple the reconstruction.
- Additionally, we can see that for a lower number of sensed pixels the median filtered image has a lower error. However, if the number sensed pixels per chip are greater than ~22, the non-filtered reconstructed image has a lower error. This makes sense since the non-filtered reconstructed image does not alter the sensed pixels, only estimates the missing ones. The median filter somewhat blurs the image, which is useful for when we only have a few sensed pixels, however when we have a larger number of sensed pixels this ends up changing the intensities of pixels which we originally had the true of and thus increasing the error for chips with a higher proportion of sensed pixels.



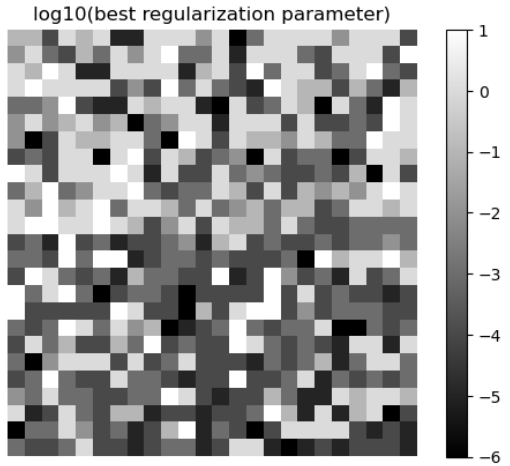
Duke

Simulations – fishing_boat.bmp – Regularization Parameter

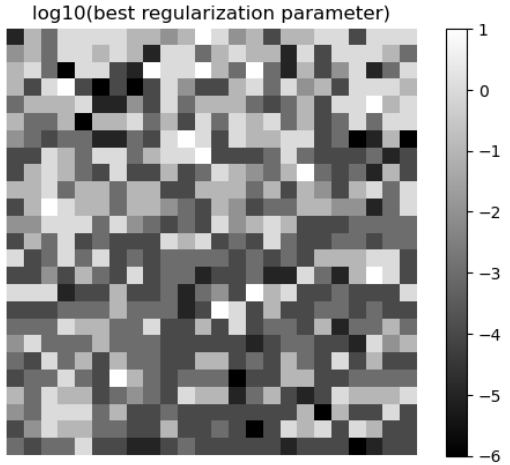
10 Sensed Pixels



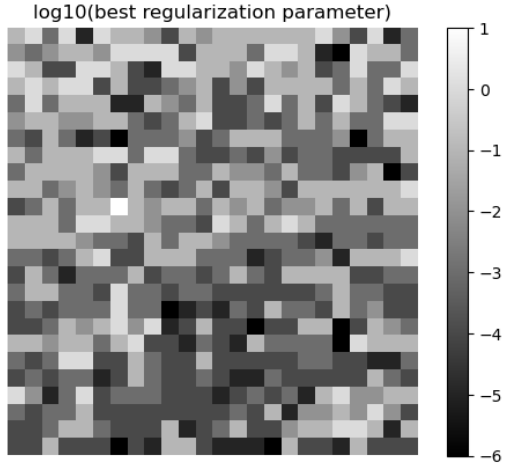
20 Sensed Pixels



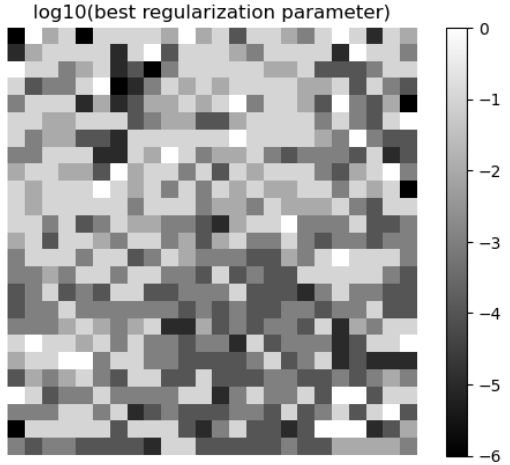
30 Sensed Pixels



40 Sensed Pixels



50 Sensed Pixels



These plots show the best parameter chosen for each chip for all 5 reconstructed images. It appears that as the number of sensed pixels increases the variance in regularization parameters decreases. Additionally, we can see this plot mirrors the original image, where darker sections represent where there is an object, and lighter sections represent where there is a plain background. It can be seen here that the main object is in the bottom right portion of the image.

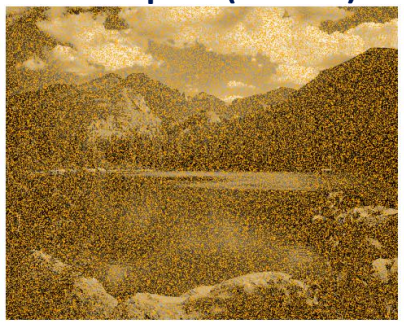
Duke

Simulations – nature.bmp – Corrupted Images

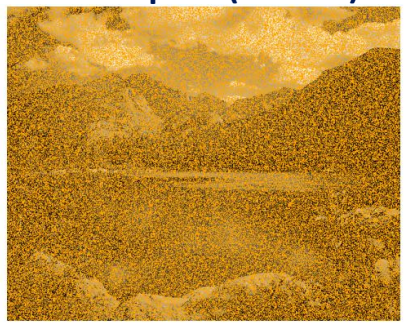
- To create a corrupted image, a specific number of pixels were removed from the original nature.bmp image. Since the image is broken down into 16 x 16 chips for reconstruction, the number of pixels removed is also per 16 x 16chip. For these simulations, there are five different levels of corruption : 106, 156, 206, 226, and 246 corrupted pixels per chip. Just as was done for fishing_boat.bmp, rather than referring to the number of corrupted pixels, we refer to the number of sensed pixels (S) which is just N * N - # of corrupted pixels. These pixels are randomly selected for each of the chips in the image for each simulation. Examples of these corrupted images are shown below.

Original

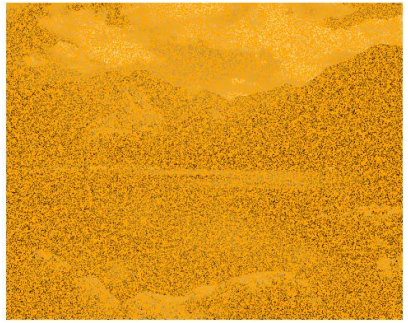
Corrupted (S = 150)



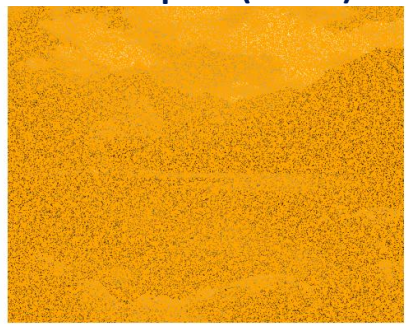
Corrupted (S = 100)



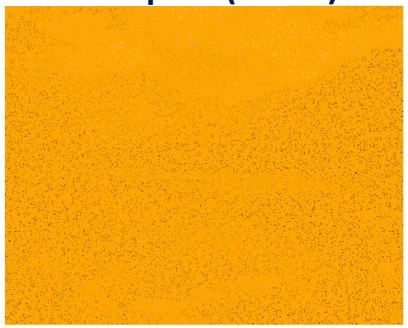
Corrupted (S = 50)



Corrupted (S = 30)



Corrupted (S = 10)



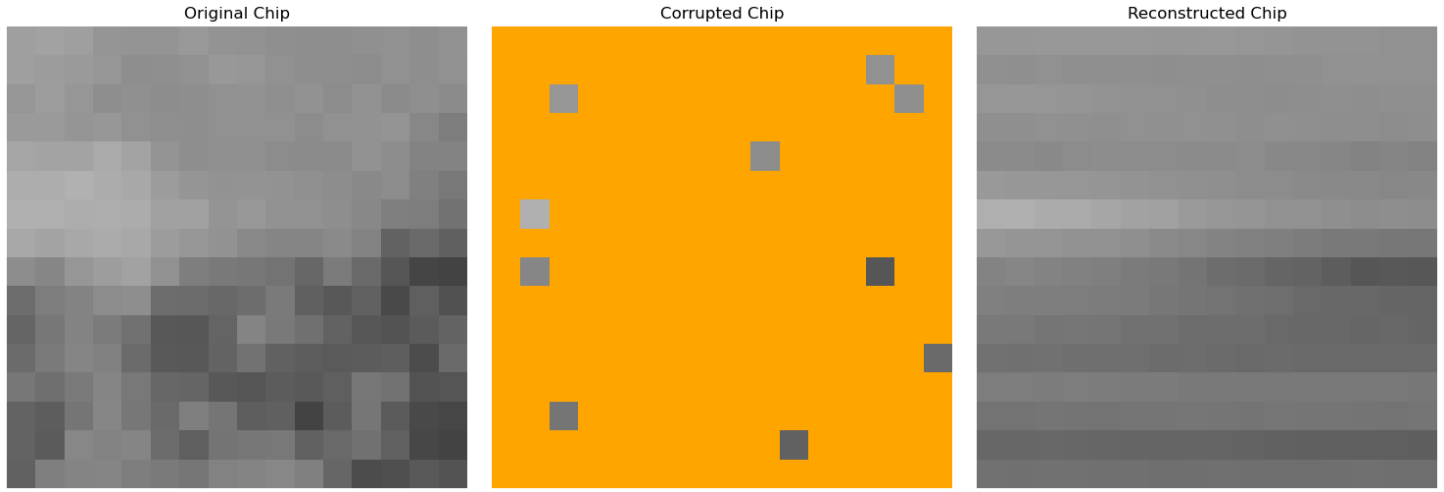
Duke

Orange pixels represented corrupted/missing pixels

Simulations – nature.bmp – Single 16x16 Chip

- The following slides show examples of the reconstruction process for a single 16 x 16 chip (this chip's top left corner is located at (0,16). The process is shown for 5 different amounts of corrupted pixels, indicated by the number of sensed pixels (non-corrupted).
- As described earlier for fishing_boat.bmp, each chip goes through a DCT transformation, a 20-Fold random subsets cross validation to determine the best regularization constant, then LASSO regression to determine the missing pixel intensities. There is no median filtering being done to each individual chip. Each time the pixels which are corrupted are randomized.
- While it is hard to tell with an individual chip, it is sufficient to say that as the number of sensed pixels decreases, the quality of the reconstructed chip decreases, which makes sense because there is less data for the model to train on. It is particularly poor with 10 sensed pixels, since this is only ~4% of the total pixels.

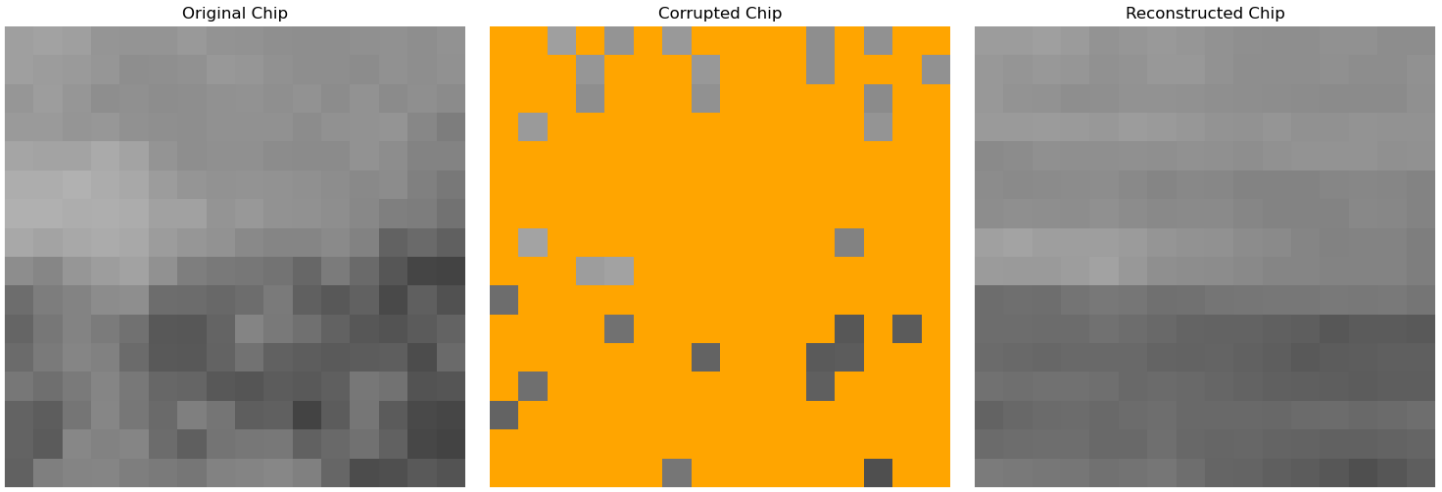
10 Sensed Pixels Per 16x16 Chip



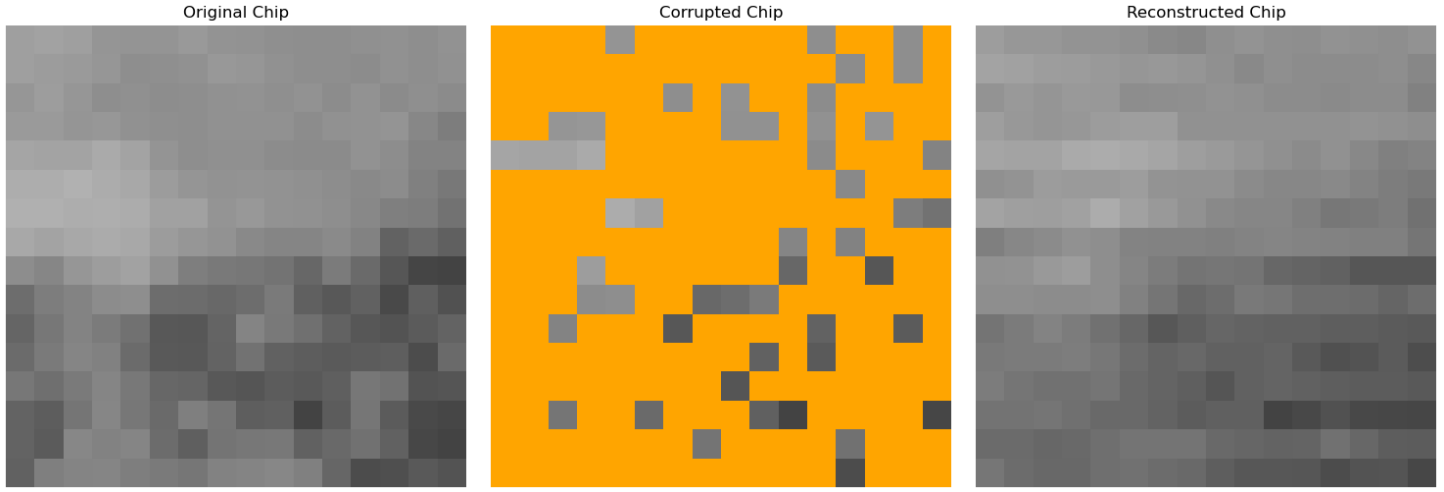
Duke

Simulations – nature.bmp – Single 16x16 Chip

30 Sensed Pixels Per 16x16 Chip



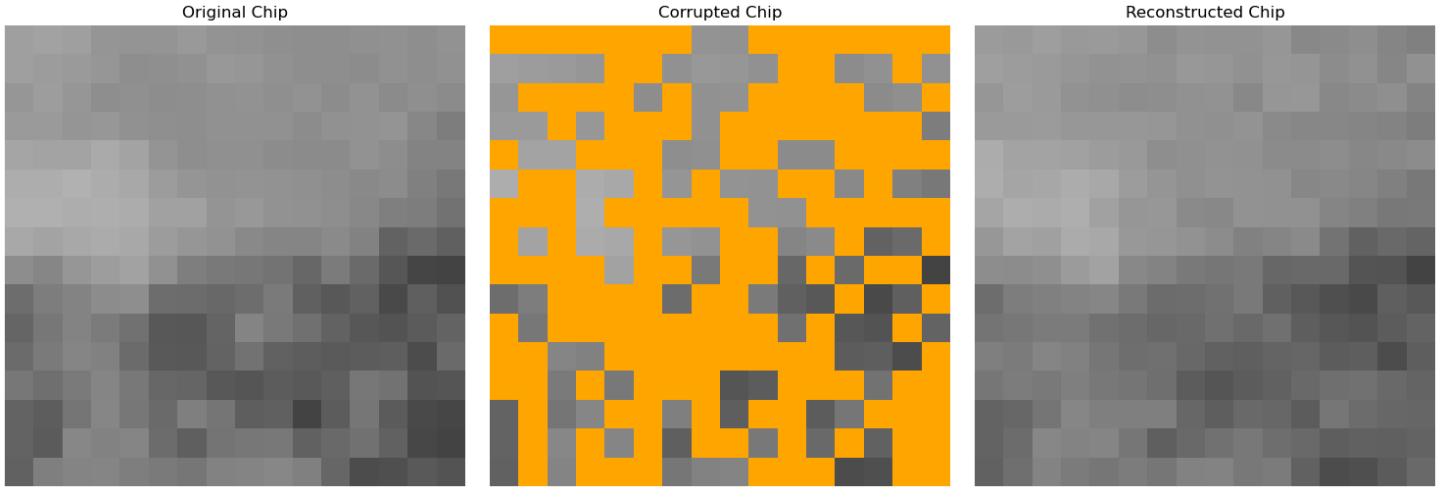
50 Sensed Pixels Per 16x16 Chip



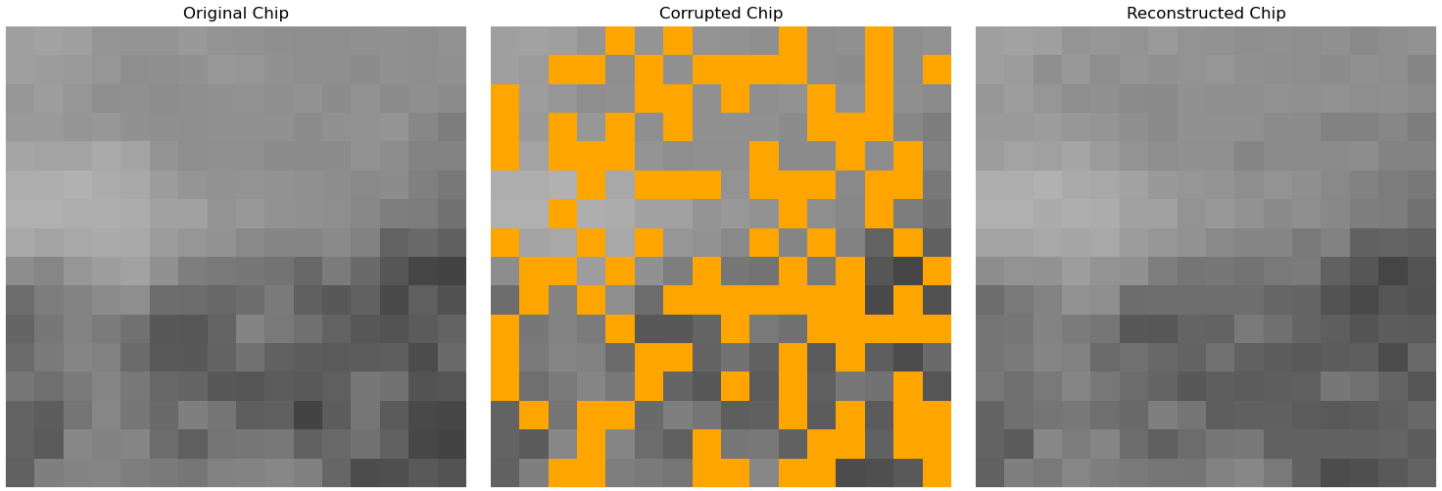
Duke

Simulations – nature.bmp – Single 16x16 Chip

100 Sensed Pixels Per 16x16 Chip



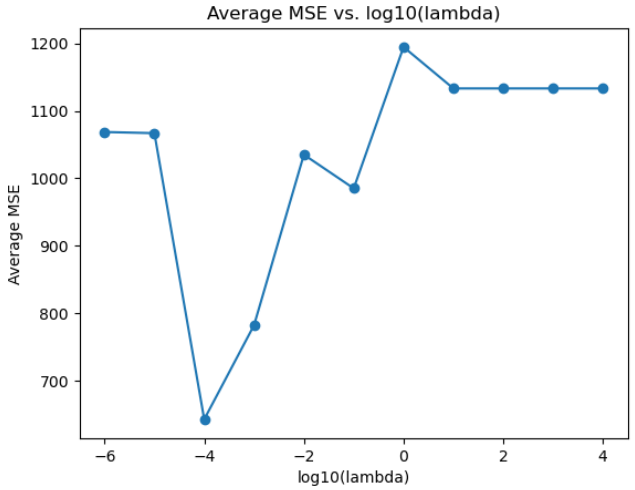
150 Sensed Pixels Per 16x16 Chip



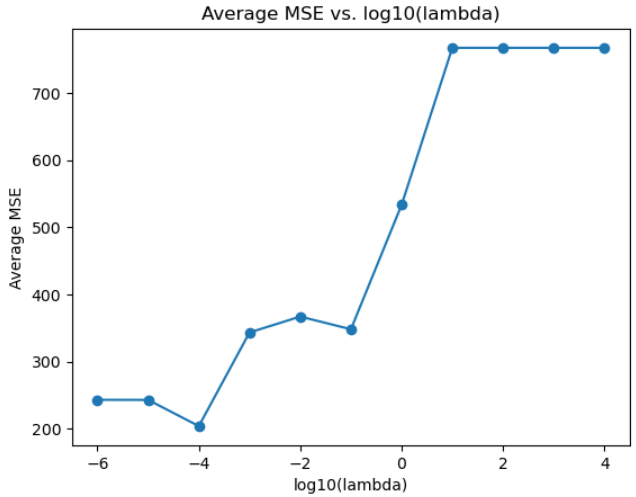
Duke

Simulations – nature.bmp – Single Chip Regularization Parameters

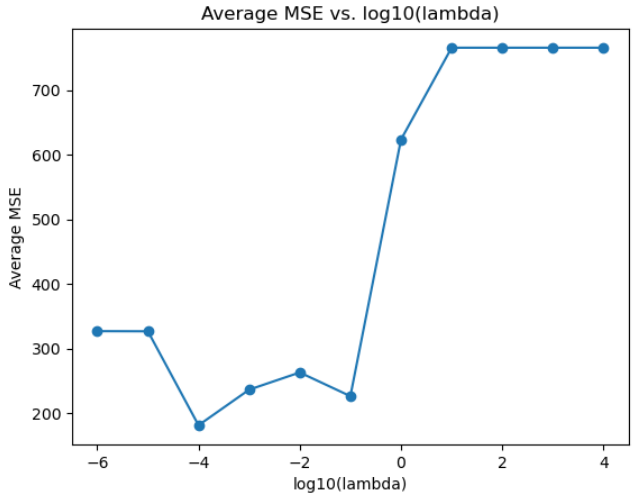
10 Sensed Pixels



30 Sensed Pixels

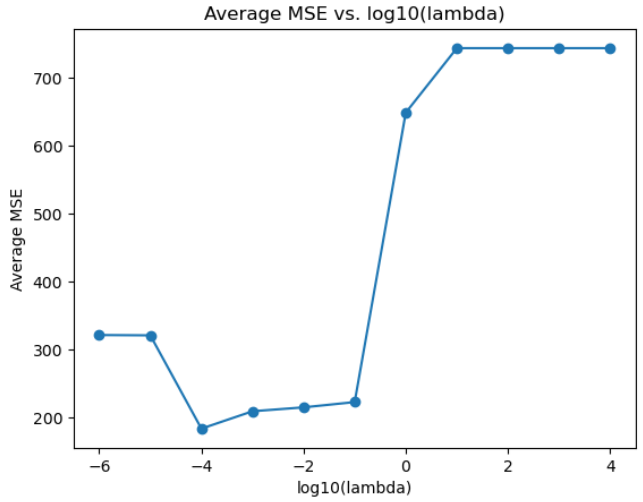


50 Sensed Pixels

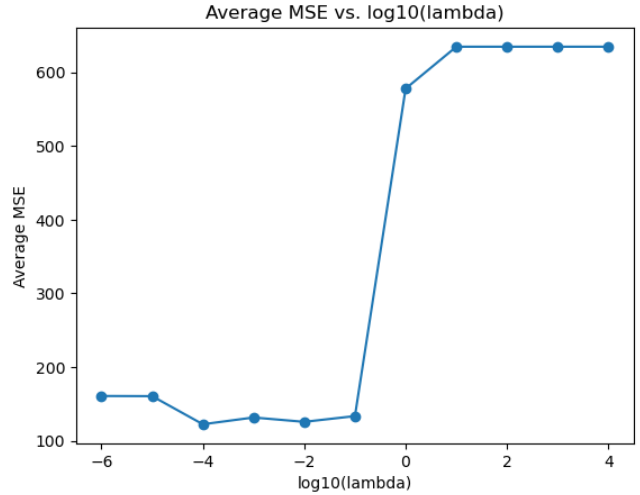


These plots show the average mean squared error across all folds as a function of the regularization parameter λ on a log scale. The selected λ is the minimizer of this function. These plots change with number of sensed pixels, which shows its sensitivity to the training data, particularly we see a more jagged plot for a lower number of sensed pixels.

100 Sensed Pixels



150 Sensed Pixels

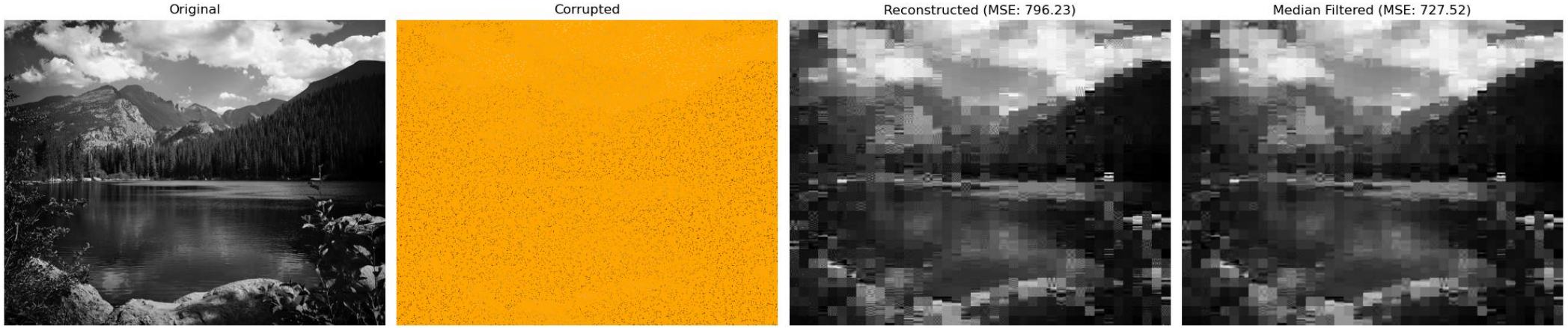


Duke

Simulations – nature.bmp – Full Image Recovery

- The following slides show examples of the reconstruction process for the entire nature.bmp image. The process is shown for 5 different amounts of corrupted pixels, indicated by the number of sensed pixels (non-corrupted).
- The more corrupted pixels there are, the poorer the reconstruction quality is, which is particularly noticeable at lower numbers of sensed pixels. Additionally, the median filter helps to smooth out these lower quality images by removing some of the staticky salt and pepper noise that occurs in the original reconstructed images. This is less noticeable in the higher quality reconstructed images, where the median filtering seems to blur parts of the image.
- We also can see that the image recovery tends to do a better job with the background like the clouds first, whereas the trees and the shrubs tend to be the last component that is appropriated reconstructed due to their complexity.

10 Sensed Pixels Per 16x16 Chip

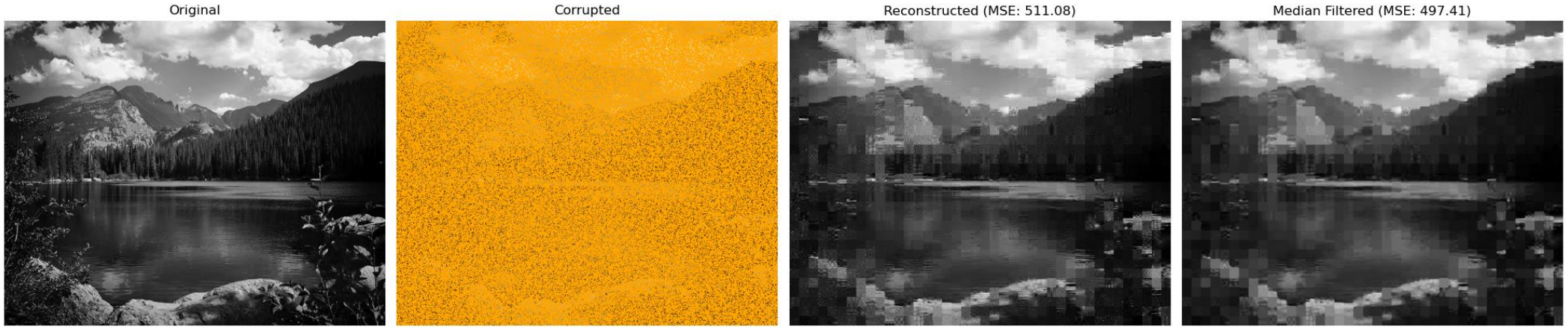


Just like for fishing_boat.bmp, we see that for only 10 sensed pixels per chip, we get a very blocky effect which makes sense due to the low number of non-corrupt pixels, however we can still see the general outline of the image details.

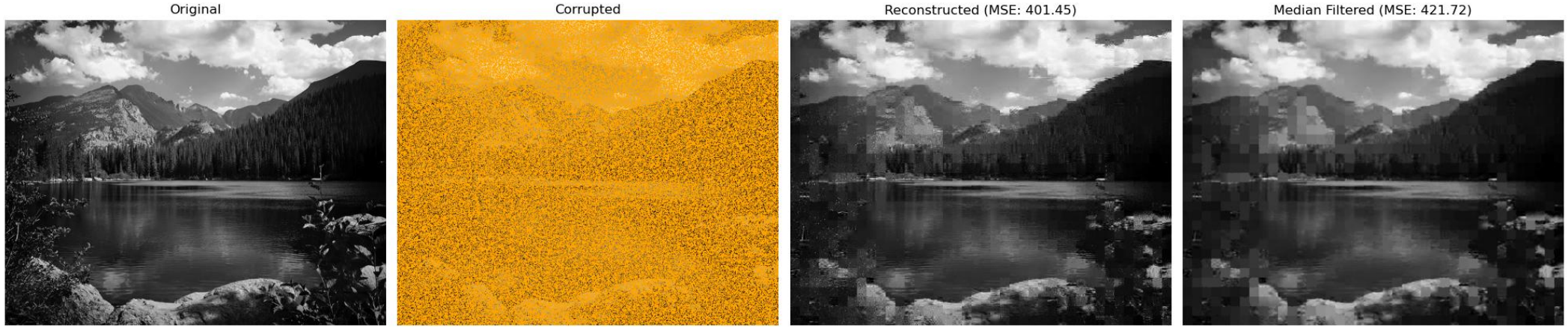
Duke

Simulations – nature.bmp – Full Image Recovery

30 Sensed Pixels Per 16x16 Chip



50 Sensed Pixels Per 16x16 Chip

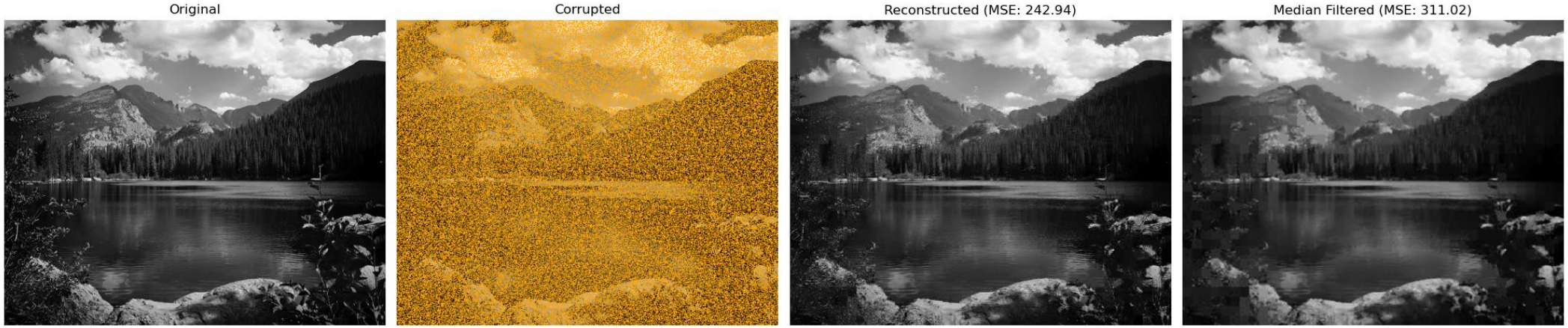


Duke

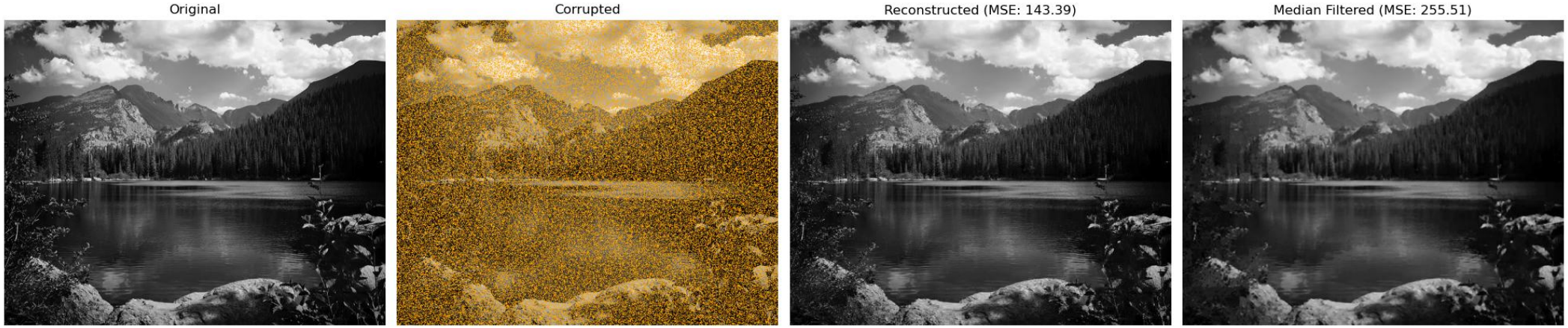
Here we see for both S values the clouds, sky and mountains look passable, but the trees and shrubbery are very choppy, since they are more intricate components of the image.

Simulations – nature.bmp – Full Image Recovery

100 Sensed Pixels Per 16x16 Chip



150 Sensed Pixels Per 16x16 Chip

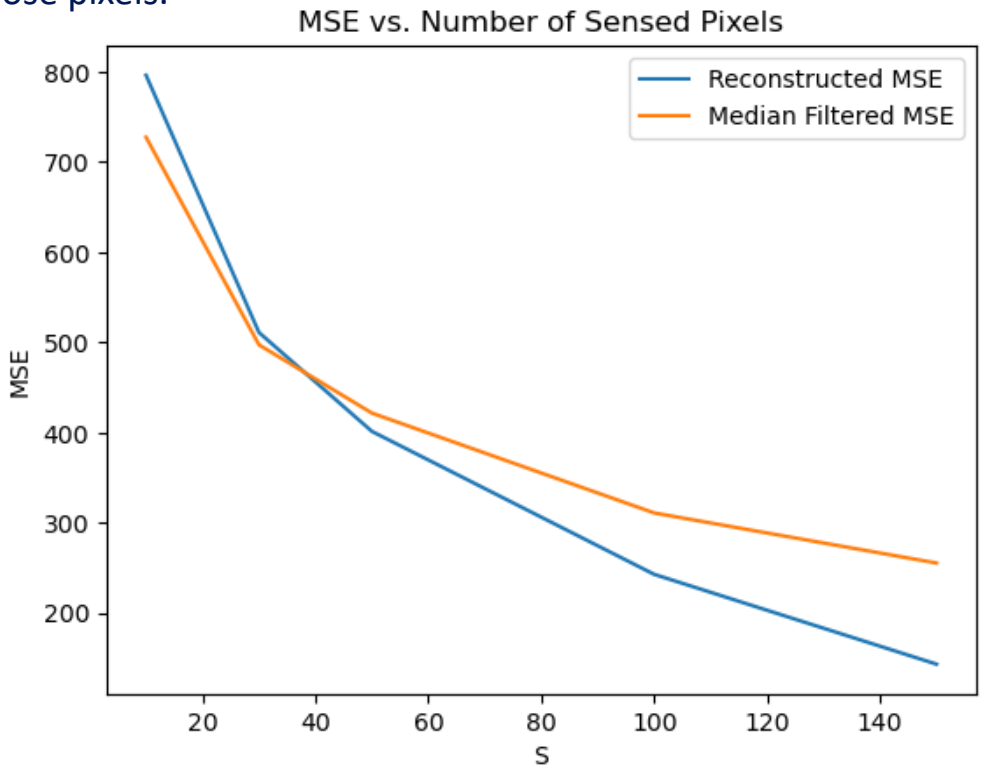


Duke

Both reconstructed images look very similar to the original images, even better than the median filtered images. The only component that seems to be off is the shrub in the front left of the image due to its complexness geometry.

Simulations – nature.bmp – Full Image Recovery

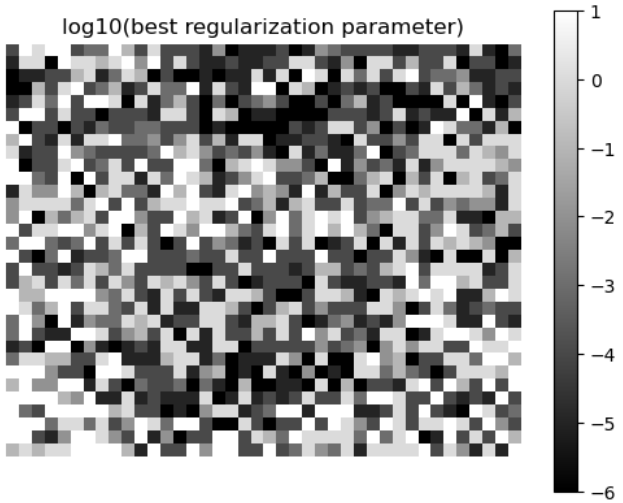
- Like fishing_boat.bmp, by plotting mean squared error vs the number of sensed pixels we can see that the mean squared error tends to decrease as the number of corrupter pixels decreases (S increases). This makes sense since with less uncorrupted pixels we have more data to train our regularized regression model, so it is more likely that our model is correct. Also, we are taking the MSE over ALL pixels, not just the initially corrupted ones, thus we include more pixels with zero error when we have less uncorrupted pixels, since we are not replicated the non-corrupted pixels when we apple the reconstruction.
- Also, we can see that just like for fishing_boat.bmp, for a lower number of sensed pixels the median filtered image has a lower error. However, if the number sensed pixels per chip are greater than ~40, the non-filtered reconstructed image has a lower error. Just as before, this makes sense since the non-filtered reconstructed image does not alter the sensed pixels, only estimates the missing ones. The median filtered image adjusts the intensities of all pixels, thus changing sensed pixels from their real value to some different value, increasing the error of those pixels.



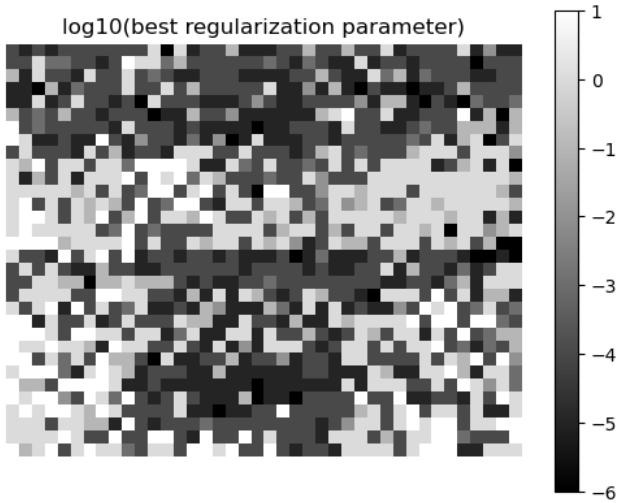
Duke

Simulations – nature.bmp – Regularization Parameter

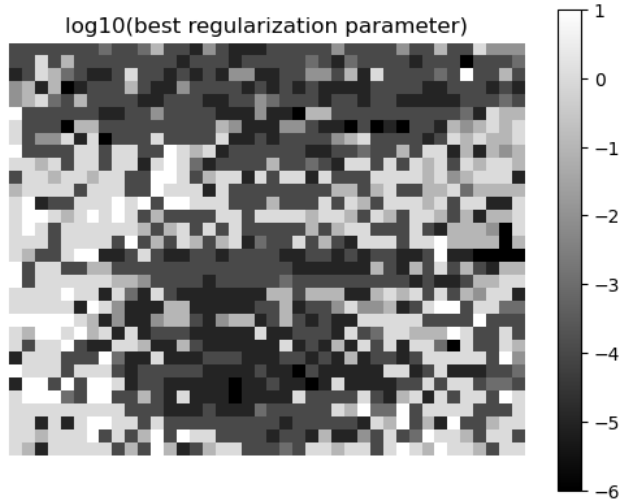
10 Sensed Pixels



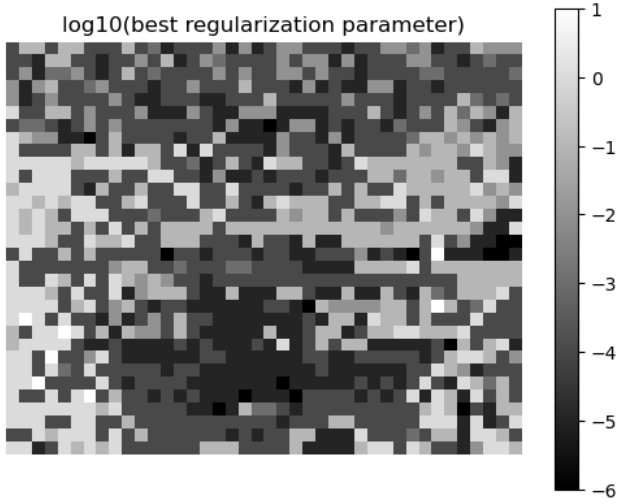
30 Sensed Pixels



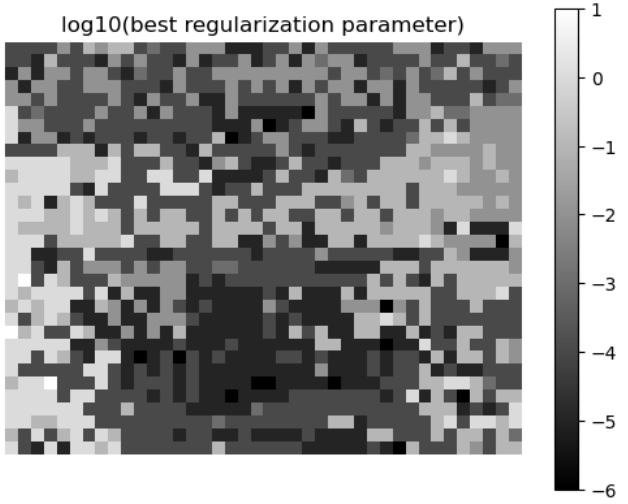
50 Sensed Pixels



100 Sensed Pixels



150 Sensed Pixels

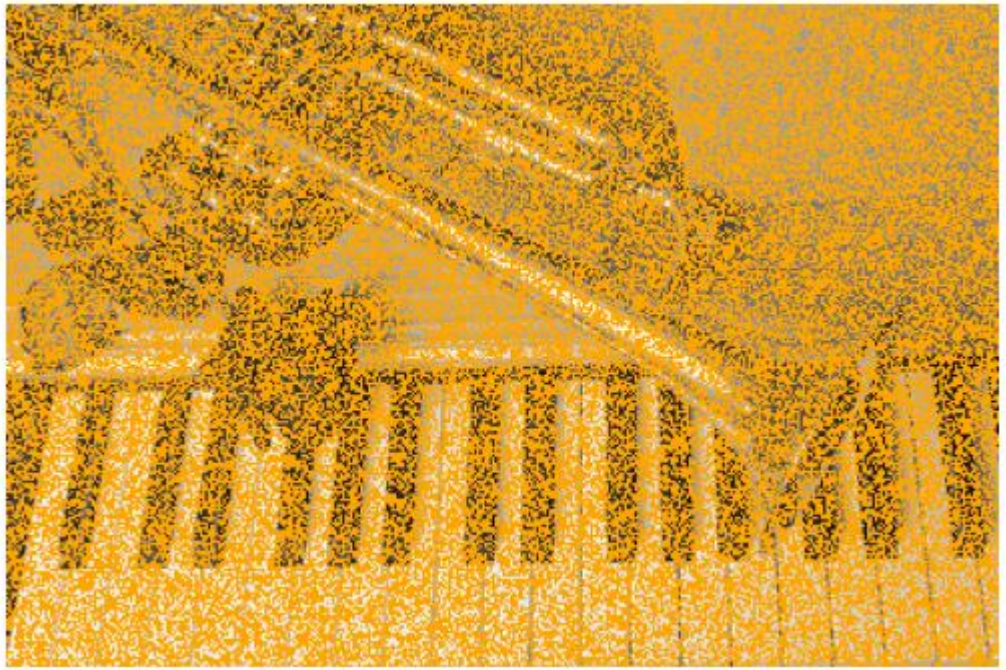


These plots show the best parameter chosen for each chip for all 5 reconstructed images. It appears that as the number of sensed pixels increases the variance in regularization parameters decreases. We notice many more chunks of the same parameter values occurring in the images with more sensed pixels compared to the ones with few sensed pixels. Additionally, we can see this plot mirrors the original image, particularly we can tend to see where the sky separates from the trees, and where the trees separates from the lake, as these have contrasting regularization parameter values.

Duke

Field Test Image

- The field test image shown below is an actual corrupted image, where the orange pixels represent pixels that are missing. The final task of this project is to complete a recovery of this image using our previously created algorithm and determine the parameters (chip size and median filter size) that provide the best reconstructed image.
- For my reconstructions I tried two different chip sizes (8 x 8 and 16 x 16) as well as two different median filter sizes to determine which would yield the best reconstructed image.



Duke

Field Test Image - Reconstructions

8x8 Chips

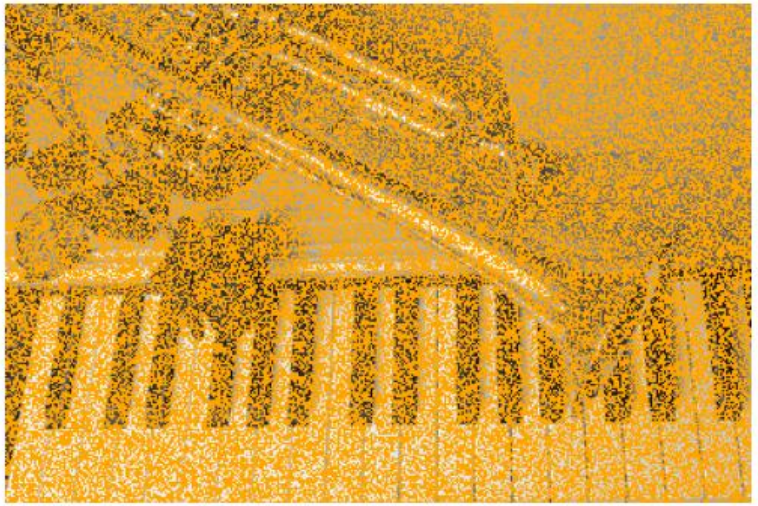
16x16 Chips

Duke

Field Test Image - Reconstructions

- For the final reconstruction, the parameters that yielded the best results were a block size of 16x16 pixels with a median filter size of 3. The images produced with 8x8 pixels seemed to have more noise throughout the image particularly near the black keys and the trumpet. The median filter with size 5 seem to blur the image too much where some of the edges seem to be smoothed too much.
- The best reconstructed image still has some issues, particularly near the horn of the trumpet where it is overlapping with black keys. This is probably the most diverse section of the image, so this makes sense. The reconstruction of the flower seems to be very accurate in this implementation.

Corrupted Field Image



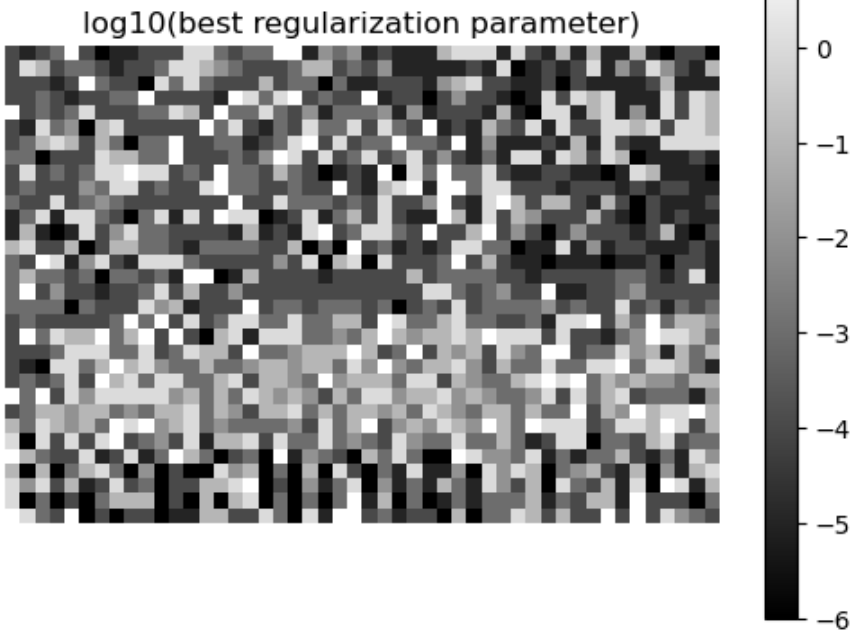
Best Reconstructed Image

Duke

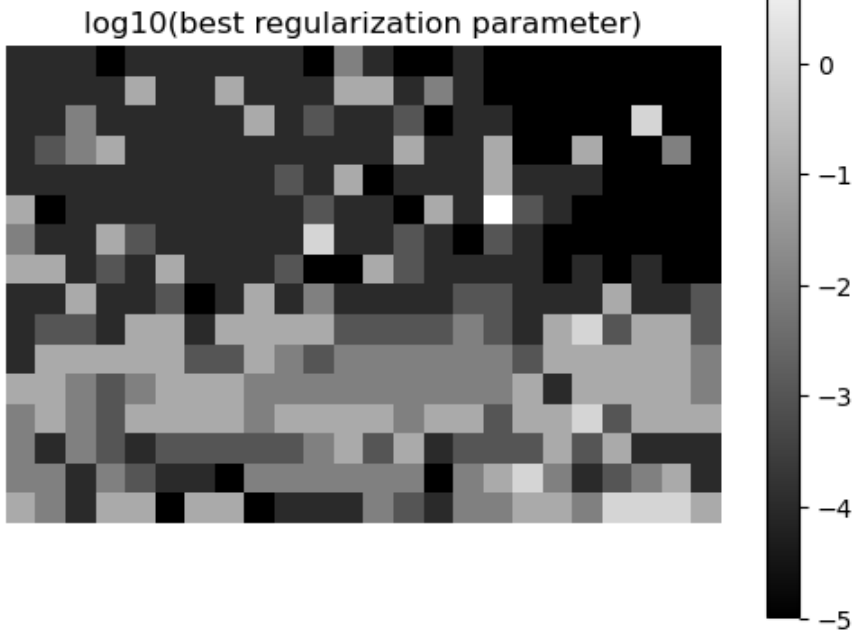
Field Test Image – Regularization Parameter

- These plots show the best parameter chosen for each chip for the images constructed by 8x8 chips and 16x16 chips respectively.
- We see that for the image made up of 16x16 chips there are significantly more clusters of the same regularization parameter compared to that of the 8x8 chips image. This is likely due to there being less of an impact on which pixels are missing if there are larger chip sizes, and thus there is more variability in the image with smaller chip sizes, if one chip contains a lot of the missing pixels for that region.
- Additionally, as we saw for the two sample images, we can somewhat make out the original image in this visualization, as we can see the boundary between the keyboard and the background very clearly, it is somewhat harder to make out the trumpet and flower.

8x8 Chips



16x16 Chips



Duke

Conclusions

- In general, the regularized regression approach works very well for image reconstruction. This definition of “very well” depends on how many pixels are missing in the first place, as with only a small amount of non-corrupted pixels, we can recover the general façade of the original image. If only around half of the pixels are missing, this approach generates a reconstructed image that is quite accurate and only seems to falter towards the edges of some of the unique objects in the images.
- For this implementation of regularized regression, it is important that you choose the appropriate basis and regularization types. For image recovery, DCT is an ideal basis to choose since images represented in the DCT basis are typically sparse, and thus naturally pairs well with LASSO for the regularized regression since it also usually outputs a sparse result. A different set of bases or a different regularization method (case-by-case ridge regression as seen earlier) would likely result in more inaccurate results if they were not similarly complementary.
- In the future I would consider trying different shapes for the chip sizes during the reconstruction of the corrupted image (namely rectangles) to see if this had any improvement on the quality of the reconstructed image. Additionally, I could consider trying a different kind of filtering on a case-by-case basis to see if there is any improvement.
- The problem itself is broken down into parts which makes modular coding very natural to implement here, which lead to efficient code which did not take too long to run these simulations. This worked really well, and I would certainly try to implement modular coding into any algorithms I work on in the future.

References - Readings

1) Van Veen, D., Jalal, A., Soltanolkotabi, M., Price, E., Vishwanath, S., & Dimakis, A. G. (2018). Compressed sensing with deep image prior and learned regularization. arXiv preprint arXiv:1806.06438.
2) Jafarpour, B., Goyal, V. K., McLaughlin, D. B., & Freeman, W. T. (2010). Compressed history matching: exploiting transform-domain sparsity for regularization of nonlinear dynamic data integration problems. Mathematical Geosciences, 42, 1-27.
3) L. Stanković and M. Brajović, "Analysis of the Reconstruction of Sparse Signals in the DCT Domain Applied to Audio Signals," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 26, no. 7, pp. 1220-1235, July 2018
4) Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B: Statistical Methodology, 58(1), 267-288.
5) Gonzalez, R. C. (2009). Digital image processing. Pearson education.
6) Boateng, K. O., Asubam, B. W., & Laar, D. S. (2012). Improving the effectiveness of the median filter.

References – Python Libraries

Python libraries used for the code include opensource libraries:

1) Matplotlib – Plotting graphs and figures
https://matplotlib.org/stable/index.html
2) NumPy – For numerical operations
https://numpy.org/doc/
3) Scikit-learn – ShuffleSplit – Implement K-Folds random subset cross validation
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.ShuffleSplit.html
4) Scikit-learn – Lasso – Perform LASSO regression
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html#sklearn.linear_model.Lasso
5) SciPy – Median Filtering
https://docs.scipy.org/doc/scipy/
6) Joblib – Accelerate run time by activating multiprocessing
https://joblib.readthedocs.io/en/stable/

Collaborations

- For all aspects of this project, from code support to results comparisons to overcoming obstacles, I collaborated with Jesen Tanadi, a fellow member of my lab who is also in the course.
- Specifically, we tended to compare our visual outputs for the reconstructions to make sure they looked relatively similar Additionally, made sure that our mean squared error values were on the same order of magnitude.
- We also spent a decent amount of time together discussing the LASSO approach and how to appropriately take the constant term (w_0) into account while using the scikitlearn implementation of LASSO. We also worked through some issues about what would be a useful package to use for K-Fold random subset cross validation, since the Kfold package from scikitlearn wasn't exactly the type of cross validation which was ideal to use for this specific problem.
- In addition to Jesen Tanadi, I also worked through a few issues with Dr. Stacy Tantum, including topics such as why the Discrete Cosine Transform is particularly useful for image reconstruction, and how that leads to LASSO being a natural choice for regularized regression. first and we discussed the importance during cross validation to compute your folds first and then apply all of your different regularization parameters to the predetermined fold, rather than producing a new set of folds for each regularization parameter.

Duke