Compressed Sensing
Image Recovery

Luis Pereda Amaya
ECE 580: Introduction to Machine Learning
Spring 2022
Link to Jupyter Notebook

Table of Contents

01

Introduction

Introduction of nature of project and its importance

02

Mathematical Formulation

Explanation of mathematical background technical approaches to this problem.

03

Experimental Results

Summary of the results found by running simulations of the system

04

Conclusions

Takeaways learned from completing this project.

01

Overview

Big Picture

In this project we are simulating that the camera which has taken our pictures has been corrupted, and we have only a subset of the pixels of the original image. From here we attempt to reconstruct the original image using the sensed pixels we have at hand. The slides immediately following give a brief introduction to the project, and those thereafter dive into the details and approaches taken to reconstruct images.

Original Uncorrupted Images

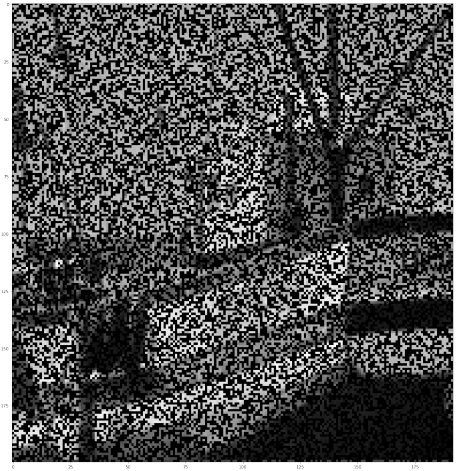
Fishing Boat

Fishing boat is a 200×192 pixel image representing our test case for a small image, which we divide into 8×8 blocks.

Nature

Nature is a 512×640 pixel image representing our test case for a large image, which we divide into 16×16 blocks.

“Corrupted” Images



Fishing Boat

30 pixels of every 8×8 block in the fishing boat image sampled, the rest are lost or “corrupted”.

Nature

100 pixels of every 16×16 nature image samples, the rest are lost or “corrupted”.

Recovered Images

Fishing Boat

The recovered fishing boat image without median filtering for the case where only 30/64 pixels were sensed per block.

Nature

The recovered nature image without median filtering for the case where only 100/256 pixels were sensed per block.

Synopsis

From the couple cases seen in the previous few slides, we can see that even with less than half of the pixels of the original image available in our corrupted images, we are able to recover the missing pixels to reconstruct images that accurately semble the original images. This presentation dives into the methods and approach to this problem, the mathematical formulations considered in tackling this problem, as well as the experimental results and the conclusions drawn from these results.

Methodology: Simulating Image Corruption

We are initially given two images, one of a fishing boat and one of a nature landscape, and to simulate the corruption of these images we split the image into NxN blocks and randomly select S pixels of each of these NXN blocks in the image, discarding the remaining N^2-S pixels by setting their values to zero. To not confuse these corrupted pixels with pixels of the original image that have value zero, we store the indices of the corrupted pixels in the image array so we know which ones to predict and which ones not to overwrite.

Methodology: Estimating DCT Coefficients

We use the 2D Discrete Cosine Transform (DCT) to reconstruct each of the NXN blocks, where the basis functions for the DCT are known, but we must predict the values of the DCT coefficients. This we do as an optimization problem, where from a training set of sensed pixels we use regression with L1 regularization (LASSO) to estimate values for the DCT coefficients that yield a minimized error on a testing set of the sensed pixels. The lambda parameter used in LASSO regression is found using cross-validation, described in detail in the following slide. With the known set of basis functions and predicted DCT coefficients, we can predict values for the corrupted pixels in each block and then from here we have effectively reconstructed an image.

Methodology: Cross-Validation to find Lambda Parameter

Before using LASSO regression to predict DCT coefficients that will be used to recover the image pixel values, cross-validation with random subsets is used to estimate the ideal lambda value, where larger lambda values penalize the complexity of the model, driving more of the DCT coefficients to 0. From a logspace of 1E-6 to 1E6, 60 lambda candidates are tested to estimate the best candidate. For each lambda, the S sampled pixels in the block are split into m=floor(S/6) testing pixels and S-m training pixels. Using LASSO with this lambda the DCT coefficients are predicted using the training pixels, and the MSE is calculated on the LASSO predictions and the testing pixels. This process is repeated 20 times for each unique lambda, and lambda with the lowest average MSE is chosen as the lambda that will be used in LASSO to calculate the DCT coefficients for that block.

Methodology: Evaluating Image Recovery Quality

Once we have estimated our lambda for each block and used LASSO to predict DCT coefficients to recover each block in the image, we can reconstruct the image using these blocks. To evaluate the quality of image recovery, the mean squared error is taken between the recovered image and the original. Then the image is passed through a median filter which replaces each pixel with the median of itself and the 8 other pixels immediately surrounding. The mean squared error is again taken between the filtered image and the original to measure the quality of image recovery after filtering. The equation below taken from the project description[1] defines the equation used to calculate the error between a recovered image and the original.

$$\frac{1}{W \times H} \sum_{\substack{1 \le x \le W \\ 1 \le y \le H}} [\hat{g}(x,y) - g(x,y)]^2$$

Project Goals

- We are looking to see how well we are able to recover the missing pixels of an image. This is our most tangible and applicable goal, as it is a measure of how well our system can accomplish its function of reconstructing a corrupted image.
- We are looking to better understand the key variables and parameters that play into the success of our system in recovering missing pixels. These parameters such as the fraction of pixels corrupted, the block size, the presence of median filtering, as well as the size of the image and its pixel density all contribute to how well we are able to achieve our goal of recovering these missing pixels, and developing a deep understanding of these factors at play is essential in fundamentally understanding the problem space.[1]
- We are also looking to gain familiarity and understanding with applications of a machine learning system rooted in ideas and methods studied in lecture now being used to tackle a tangible problem.

Key Results

All in all, we saw that with this system using the Discrete Cosine Transform to recover corrupted images, we could be quite successful in recovering the missing pixels of images. As we expected, as more pixels were sensed per NxN block in either image, the more successfully we were able to recover the image. Qualitatively, the larger nature appeared to respond better to this form of image recovery, as past 100 sampled pixels per 16×16 block the recovery of the image was excellent, highly resembling the original image, while with the smaller boat image, the same sort of success was only really seen when 50 pixels were sensed of each 8×8 block. While we expected median filtering to increase the quality of image recovery, this was only true in the case where 10 pixels were sampled per block in both the 8×8 and 16×16 case, as the median filter tended to blur the images in a way that lost some detail of the recovered images. Additionally, in using the entire image in the MSE calculations, we know that median filtering will hurt this error score, especially for higher values of S, as we know that any sensed pixel will contribute an error of 0, however after median filtering this will not necessarily be the case. While the median filter provides an element of smoothing to the image, in the simulations where S was a higher value, this smoothing appeared to detract from, rather than add to, the image recovery.

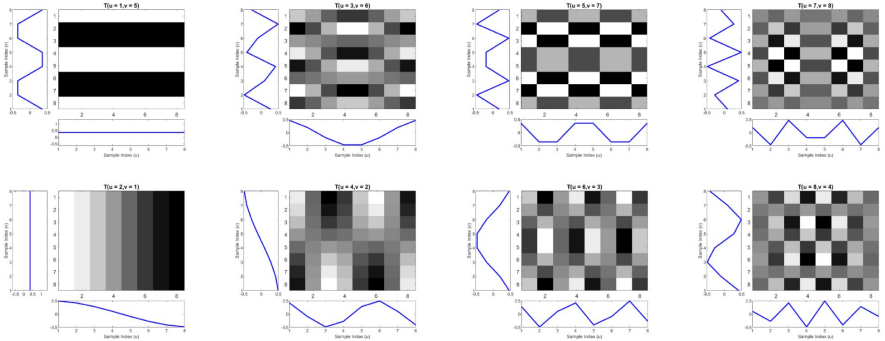
02

Mathematical Formulation

Background on the DCT

The Discrete Cosine Transform is similar to the Fourier Transform, in that both take a signal from the spatial domain into the frequency domain.[2] While the Fourier Transform expresses a function as a sum of both cosines and sines, the Discrete Cosine Transform represents a function as a sum of only cosines, which represent basis functions for this representation, where visuals of some of these 2D basis functions for spatial frequency pairs (u,v) are provided below[1], with the blue lines representing the 1D basis functions.

The DCT is widely used in signal and image processing, especially as a useful tool in image compression. DCT is useful for image compression, as the DCT representation of an image can be used to discard image information that may not be noticed by the human eye. A commonly used application of the DCT is in JPEG compression, whereby a process known as JPEG quantization discards high frequency information that is generally less important to the image, allowing for the image to lose some information without impacting its viewing quality.[3]



More on the DCT

As seen from the equation below [1], a rasterized image can be represented as the sum of the basis functions of the transformation matrix, scaled by the DCT coefficients. The transformation matrix is always known as long as the dimensions of the image are known, while the DCT coefficients are calculated from the product of the inverse transformation matrix with the rasterized image when the complete image is available. From here we see the usefulness of the DCT for this project. Since not all of the pixels of the complete image are sensed, we must use the known pixels to estimate DCT coefficients. With these estimated DCT coefficients, we will be able to estimate the unknown pixels by taking the superposition of the corresponding rows of the transformation matrix scaled by the coefficients.

$$g(x,y) = \sum_{u=1}^{P}\sum_{v=1}^{Q} \alpha_u \cdot \beta_v \cdot \cos\frac{\pi(2x-1)(u-1)}{2\cdot P} \cdot \cos\frac{\pi(2y-1)(v-1)}{2\cdot Q} \cdot G(u,v)$$

Image pixel **Transformation** **DCT coefficient**

Underdetermined Linear System

In this problem, by trying to reconstruct the corrupted pixels of an image using the 2-D Discrete Cosine Transform, we approached the problem as an underdetermined linear system as we were dealing with a problem characterized by having fewer equations than unknowns. In order to perfectly reconstruct the image, this would require a fully determined system where all of the DCT coefficients would be known, however as we use the pixel values to estimate the coefficients in our approach, this would require us to know all of the pixel values preemptively to satisfy a fully determined system, which would defeat the point of this exploration. Instead, we our left with an underdetermined system which naturally leads to an optimization problem: using the known, or sensed, pixel values to predict the unknown DCT coefficients in such a way that the recovered image yields the least error in comparison to the original image.

Why Regularization?

As described in the methodology, our approach to recovering the missing pixels in an image is centered on predicting DCT coefficients. As in any machine learning or pattern-recognition problem, we wish to avoid overfitting our predictions to the training set, so we must enforce some sort of regularization to limit the complexity of our model. In terms of the bias-variance trade-off, we want to restrict the variance of our model so that it does not create a model that performs excellently on the training data (known pixels), however predicts DCT coefficients very poorly for the unknown pixels. Instead we want to allow for a higher bias in training to enforce a model with lower variance that is predictive for the unknown pixels in the image. By penalizing a more complex model with more weights (DCT coefficients) via regularization, we can allow our model to generalize more effectively. In the context of this problem as a linear system, rather than allowing our model to be of any order or complexity, regularization restricts our solution space to a class of potential models that are less complex, and this additionally allows us to more easily converge on a potential solution via numerical methods, since we have less potential solutions to consider.

But Why L1 Regularization?

The reason for using L1 regularization for our DCT coefficients as opposed to L2 regularization comes from domain knowledge in image processing, where it is known that images tend to be sparse in the DCT domain.[4] We use this information to guide us in a direction of choosing a regularization that gives us many DCT coefficients of zero, rather than simply small values. L1 regularization will tend to impose sparsity in the DCT coefficients, as seen in the geometric representation in two dimensions. The LASSO solution is represented by the first place these shaded constraint regions touch the contours of the loss function, and for L1 regularization, the rotated square, this solution often falls on a corner, and if this is generalized to n-dimensions we can see how such a regularization would result in a solution where many weights are zero. On the other hand, the L2 regularization constraint has no corners, and would thus rarely result in DCT coefficients of zero, which is inconvenient for our particular problem space.

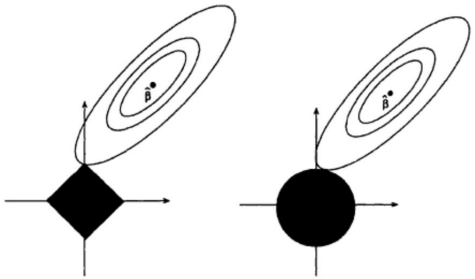
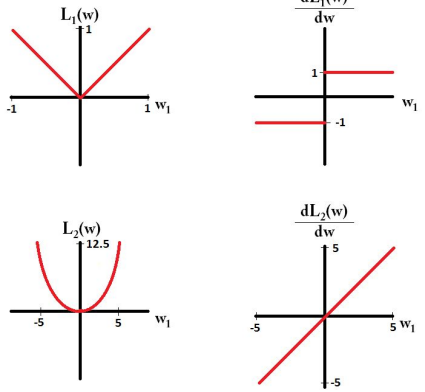


Image taken from Tibshirani[4]



Another analysis of the L1 loss more often leading to DCT coefficients equal to zero studies the gradients of the $L_1(w)$ and $L_2(w)$ loss functions. $L_1(w)$ is a linear function, thus the gradient with respect to the weights is a constant, while $L_2(w)$ is a quadratic function where the gradient is linear. Thus, a descent algorithm used to optimize a model will take smaller and smaller steps towards zero for L2 regularization as the weights approach 0, whereas with L1 regularization the step size will be the same towards zero regardless of the value of the weights. From this we can interpret that small weights will be driven more rapidly to zero by L1 regularization, achieving the result we desire since images are sparse in the DCT domain.

$$\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 \quad \text{subject to} \quad \|\beta\|_1 \leq t$$

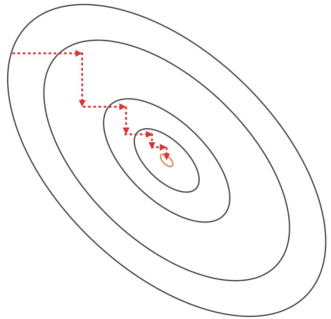
https://www.stat.cmu.edu/~ryantibs/statml/lectures/sparsity.pdf

LASSO Regression

$$\min_{w} \frac{1}{2n_{\text{samples}}} ||Xw - y||_2^2 + \alpha ||w||_1$$

https://scikit-learn.org/stable/modules/linear_model.html#lasso

We utilized the Scikit Learn implementation of LASSO which to minimize the loss function shown above to return the optimal DCT coefficients. This function attempts to reduce the bias of the predictions indiscriminate of their sign by squaring the Euclidean norm of Xw-y, while constraining the weights by penalizing the L1 norm of these weights, imposing sparsity as we noted before. Increasing alpha further reduces the variance of our model by restricting our model to a lower complexity. This package minimizes this loss function via coordinate descent, meaning the loss function is minimized along coordinate directions, with one weight being modified at a time as opposed to gradient descent which would modify the entire weight vector in the opposite direction of the gradient at each step of the algorithm, and these steps are repeated until the algorithm converges on a local minimum for the loss function.



https://www.cs.ubc.ca/~schmidtm/Documents/2015_OptBigData_GaussSouthwell.pdf

Why use a median filter?

Recovering the images in the pixel by the DCT with predicted values of the DCT coefficients provides us with accurate results overall, in particular when we have a relatively large subset of sensed pixels. However, we cannot expect our model to predict each and every pixel well, and we can see in the reconstructed images there tend to be specks or small areas in the image where there are pixels that are seemingly entirely out of place. Median filtering is an effective tool to remove these discrepancies, as these outlier pixels will be replaced by a more conservative estimate of what this value should be. We use a 3×3 window of pixels, in which each pixel is replaced by the median of this neighborhood of 3×3 pixels. The result of such filtering is a smoothing effect that helps reduce the blocky nature of some of the images with fewer sensed pixels, and the removal of these strange outlier pixels that speckle the images. The reason for using a median filter rather than a frequency selective filter such as a high-pass or a low-pass filter is that our motive in using a filter is to try and smooth our image signal to remove some erratic predictions that may arise from our system, not select for only a specific range of signal frequencies. Using such a frequency selective filter would remove information from the image that may be important for the clarity of the image, one example being that edges are high-frequency components of images, so a low-pass filter would cause the system to lose information defining clear edges, hurting the accuracy of our image reconstruction.

03
Experimental Results

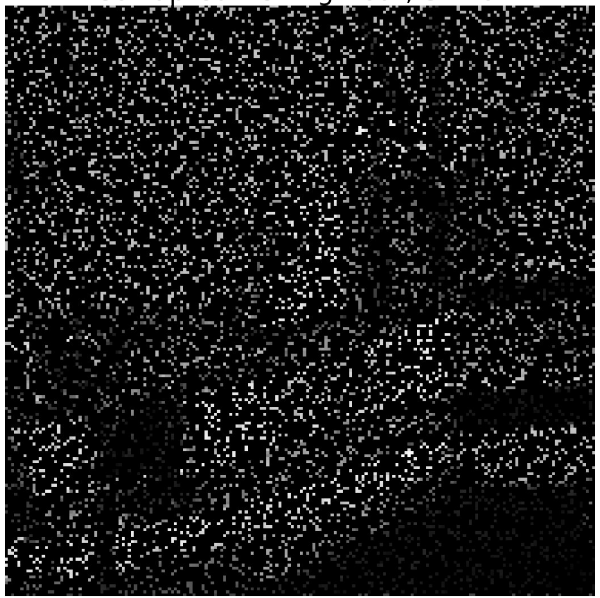
Corrupted Images vs Recovered

Fishing Boat: S=10

MSE=844.21

At a glance, this does not appear to be a great reconstruction of the boat image, where the recovered image is very blocky and when looked at closely appears to resemble the individual components or 2-D bases of the transformation matrices for the DCT. While the recreated image is not too accurate by inspection and has high MSE, we expect this as only 10 samples are taken from each 8×8 block and from these we have to predict values for the remaining 54 pixels. It is notable that even for this worst case of our results, the scaled-down image below of this screen seems to approximate the original image quite well.

Corrupted Fishing Boat, S=10



Recovered Fishing Boat, S=10

Original Fishing Boat

Fishing Boat: S=20

MSE=526.91

In this case, where 20 pixels are sensed of each of each 8×8 block in the fishing boat image, we see a sharp increase in the image quality with more sharply defined edges and the blocks in the image appearing less drastically like the basis functions of DCT for spatial frequency pairs. From the MSE values we see a 37.6% in accuracy from the image recovery here where S=20 compared to when S=10. The small scale image reflects this as we are able to scale the size of the image up from the last case and still see a sharper representation of the image.

Corrupted Fishing Boat, S=20

Recovered Fishing Boat, S=20

Original Fishing Boat

Fishing Boat: S=30

MSE=345.21

A similar trend follows here where 30 pixels are sampled of each 8×8 block of the fishing boat image. We can see that the recovered image resembles the original image much more so than when 10 or 20 pixels were sensed per block, with a 34.5% improvement compared to the MSE where S=20. The most notable difference is the edges of the 8×8 blocks are far less noticeable, with the image having an overall smoother and more natural characterization.

Corrupted Fishing Boat, S=30

Recovered Fishing Boat, S=30

Original Fishing Boat

Fishing Boat: S=40

MSE=202.60

Continuing on the trend we have seen, this case where 40 pixels of every 8×8 block are sensed improves on the MSE of when S=30 by 42.3%, and we can note that there appears to be a linear relationship between this change in sample size and MSE. Qualitatively, this recovered image mostly reduces the number of erratic pixel predictions compared to the previous case, and finer details such as the masts and ropes are captured far better.

Corrupted Fishing Boat, S=40

Recovered Fishing Boat, S=40

Original Fishing Boat

Fishing Boat: S=50

MSE=92.64

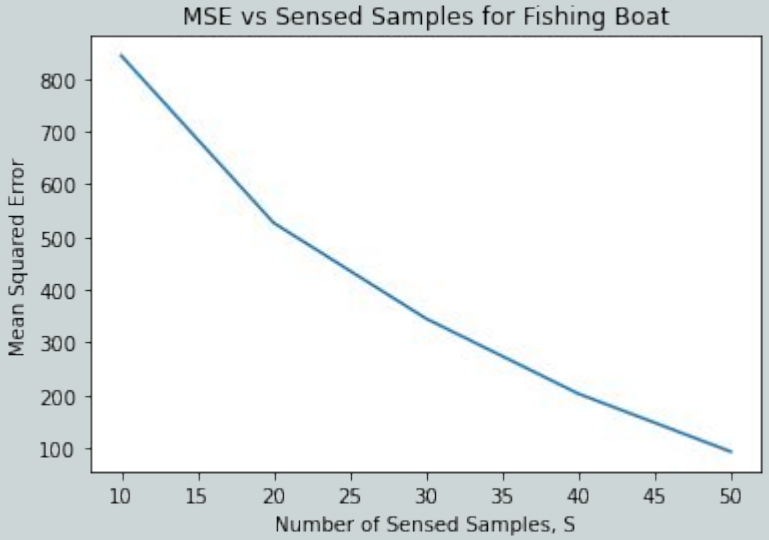
This recovered image, with 50 samples per 8×8 block, qualitatively appears to be an almost near-perfect replica of the original image. The major discrepancies are speckled erratic pixel predictions, but overall the image is recovered in a manner very similar to the original. The improvement on the MSE in this case is 54.3% compared to S=40, showing the largest improvement so far for an increase in 10 samples per block.

Corrupted Fishing Boat, S=50

Recovered Fishing Boat, S=50

Original Fishing Boat

Results of Image Recovery for Fishing Boat



The relationship between the number of sensed samples in the fishing boat image and the mean squared error as compared to the original image is shown to be what we would expect. A greater number of sensed pixels provides more data from which to formulate a prediction, and our calculations for the MSE are thus more aptly able to predict the unknown DCT coefficients.

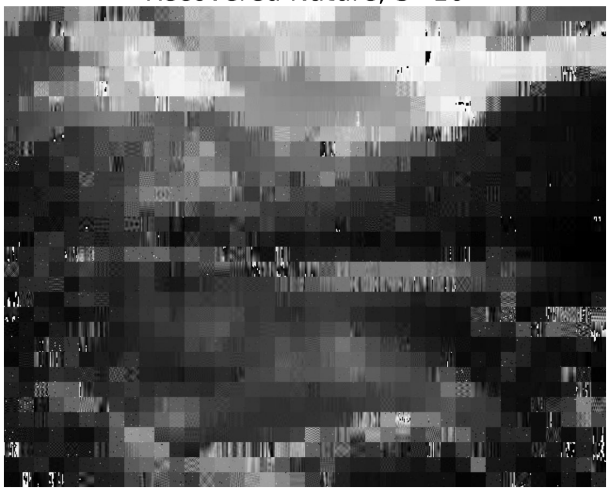
Nature: S=10

MSE=892.80

Similarly to how the fishing boat with 10 pixels sensed per block did not appear to resemble the original image at all, and the blocks resembled the 2D basis function of the transformation matrix of the DCT, this image of the reconstructed nature image with 10 pixels sampled for every 16×16 block hardly resembles the original image. It is somewhat surprising that the MSE for this image is similar to the fishing boat image with S=10, since in this case only 10/256 pixels in each block are sampled where before 10/64 pixels in each block are sampled. This being said, although the details of this image are lost almost entirely, the general patterns of brightness are captured by the pixel predictions, which is more clear when we scale this picture down to a very small size.

Corrupted Nature, S=10

Recovered Nature, S=10



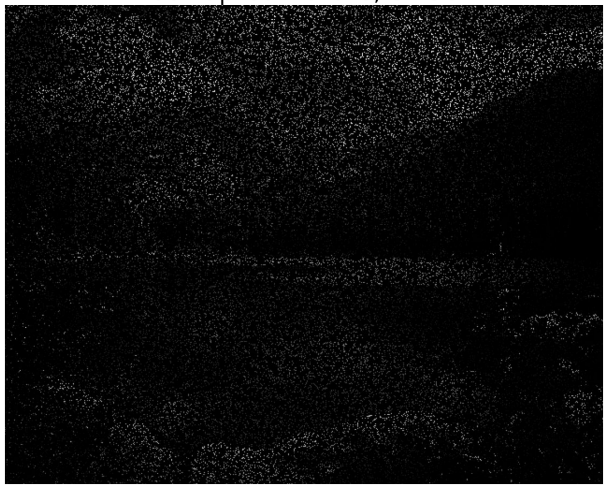
Original Nature Image

Nature: S=30

MSE=574.69

Qualitatively, we can see that although the recovered nature image with 30 samples is still very blocky with certain blocks appearing more like DCT basis functions rather than components of the image, the recovery quality has improved significantly. From 10 sensed pixels per block to 30, there is a 41.1% improvement in the MSE, showing a high, but predictable, increase in accuracy of recovery as we are able to formulate our predictions on a larger set of data. From the minimized image we can again appreciate that the patterns of the image are captured by the predictions, albeit the blockiness of the image is apparent and somewhat distracting.

Corrupted Nature, S=30



Recovered Nature, S=30

Original Nature Image

Nature: S=50

MSE=454.85

With a 20.9% improvement in the MSE for S=50 to S=30, we can see a quantitative improvement in our recovery of the image, and qualitatively this can be seen by some of the blocks that appeared more like DCT bases than part of the image in S=30 becoming more reflective of the image content in that area in this case. By inspection, the improvement in this image is not nearly as stark in comparison to when S=30 as opposed to the jump from S=10 to S=30. There are some improvements in the image quality overall, but judging from the smaller jump in MSE improvement and not-as-noticeable improvements qualitatively, a large portion of this greater accuracy may potentially be attributed to more pixels being sensed rather than improved predictions.

Corrupted Nature, S=50

Recovered Nature, S=50

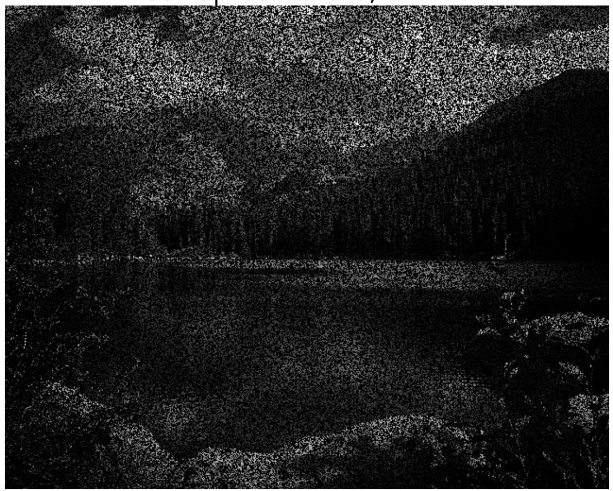
Original Nature Image

Nature: S=100

MSE=292.33

In this case of S=100, we have 50 more pixels out of the 256 pixels in each block sensed for this case as opposed to the last case where S=50, and we measure a 35.7% increase in accuracy in the MSE between these two simulations. The improvements in this recovered image are drastic, with much finer details being captured, particularly in the clouds, and areas of steady brightness are much smoother while in the previous case these areas were far more blocky. The details in the ripples of the water as well as in the shape of the trees are remarkable improvements to the previous recovered image where S=50.

Corrupted Nature, S=100



Recovered Nature, S=100

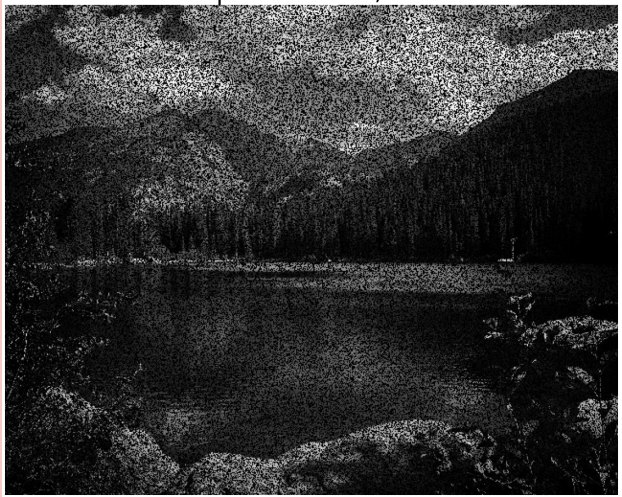
Original Nature Image

Nature: S=150

MSE=172.80

Showing a 40.9% increase from the the S=100 simulation, the recovered image nature when S=150 is near perfect. The remnants of any sort of blockiness prevalent in the the S=100 reconstruction is gone, and nearly all of the details present in the original image are captured. The most prevalent issue in this simulation is the presence of some erratic pixel predictions, especially in the lower right of the image, but overall this is an exceedingly accurate compared to the original image.

Corrupted Nature, S=150



Recovered Nature, S=150

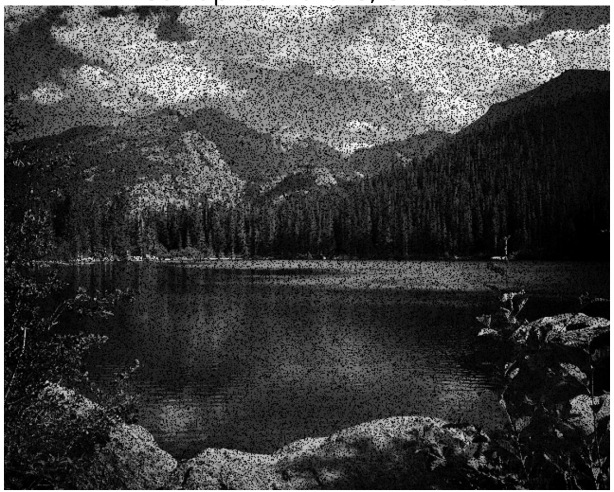
Original Nature Image

Nature: S=200

MSE= 80.40

With a 53.5% improvement from the simulation where S=150, this simulation is nearly identical to the original image. There are some pixels that are visibly poorly predicted but they are very sparse in the recreated image, and overall this is a highly accurate reconstruction of the nature image.

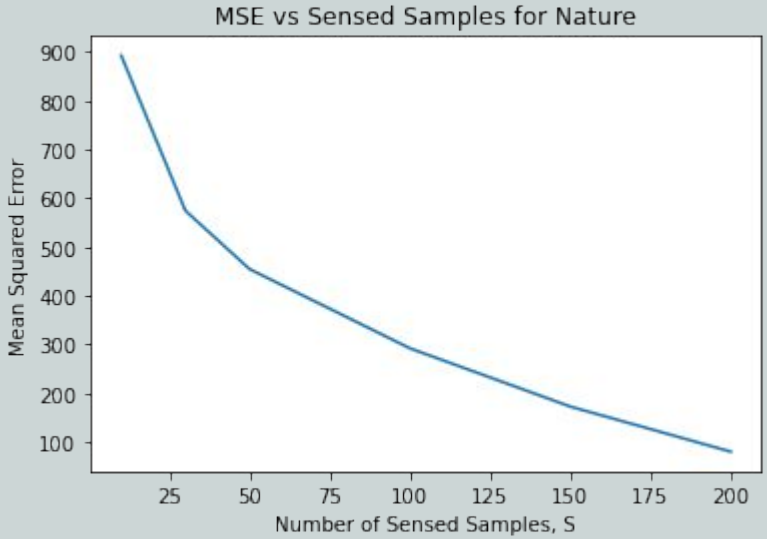
Corrupted Nature, S=200



Recovered Nature, S=200

Original Nature Image

Results of Image Recovery for Nature



Similarly to the results from the fishing boat image, we see the same inverse relationship between number of sensed pixels per block and MSE, which is again what we would expect being given more information to use in predicting the unknown pixels. One notable difference, is that the plot of MSE vs sensed samples per 16×16 block in this case decreases at a faster rate initially, then tapers off to a more consistent linear relationship, while the fishing boat showed a linear relationship between MSE and sensed samples per 8×8 block throughout the range of 10 samples to 50 samples per block.

Results of Median Filtering

Fishing Boat: S=10

It is a bit difficult to notice the improvements between the filtered and unfiltered images of the fishing boat with S=10 even though the filtered image has a lower MSE, since both are far off from being accurately representing the original image. The filtered image smooths over the pixel values yielding blocks that look less like the bases of the DCT, and captures the overall patterns of the original image, but very little of the detail.

MSE=791.13

Recovered and Filtered Fishing Boat, S=10

Original Fishing Boat

MSE=844.21

Recovered Fishing Boat, S=10

Fishing Boat: S=20

In this case the MSE of the filtered and recovered image is higher than that of the unfiltered image. Qualitatively, it seems that the filter goes beyond smoothing the image to blurring it excessively, so that no fine edges or details are captured in the filtered image, and this likely contributes to the lower MSE.

MSE=594.82

Recovered and Filtered Fishing Boat, S=20

Original Fishing Boat

MSE=526.913

Recovered Fishing Boat, S=20

Fishing Boat: S=30

We begin to see a relationship here that as the number of sensed pixels per block increases, the MSE of the filtered images begins increases relative to the MSE of the unfiltered images. We can reason that this is because the accuracy of the recovered images increases as the number of samples per block increases, and once there is a threshold of accuracy reached, filtering blurs the recovered image hurting the accuracy rather than helping it

MSE=421.78

Recovered and Filtered Fishing Boat, S=30

Original Fishing Boat

MSE=345.21

Recovered Fishing Boat, S=30

Fishing Boat: S=40

The same observations can be made here for S=40. While the median filter smooths the image removing some of the erratic pixel predictions, some of the finer details captured in the unfiltered image are blurred in the filtered image. Additionally in the calculation of MSE, median filtering will change values of the sensed pixels which would have a loss of 0 in the unfiltered case, so this is another factor driving the MSE up.

MSE=276.59

Recovered and Filtered Fishing Boat, S=40

Original Fishing Boat

MSE=202.60

Recovered Fishing Boat, S=40

Fishing Boat: S=50

The effect of median filtering changing values from the original image that exist in the recovered image can again be seen here. The filtered image appears to be a hazy and slightly blurry representation of the original image, while the unfiltered recovered image is much more crisp but with some erratic pixel predictions interspersed, overall giving the unfiltered image an appearance more similar to that of the original.

MSE=215.60

Recovered and Filtered Fishing Boat, S=50

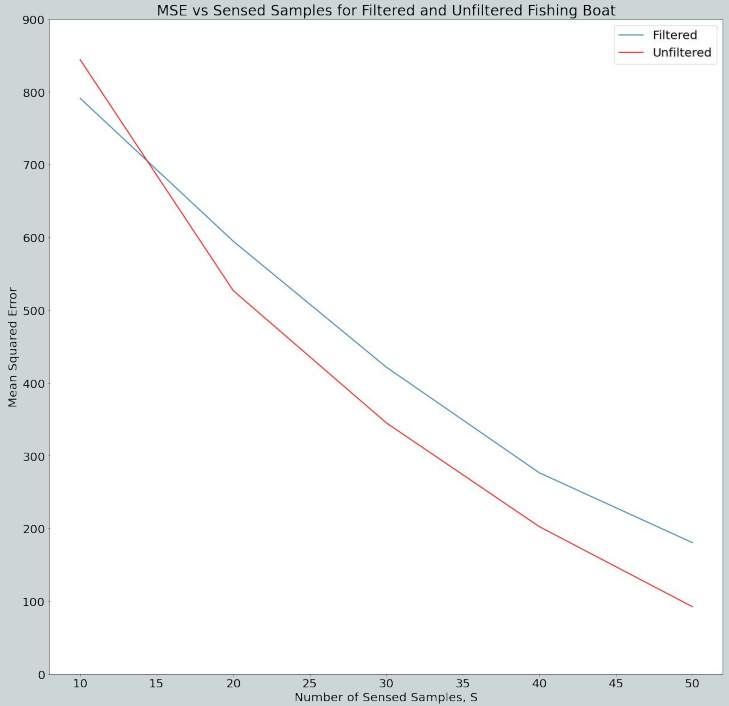
Original Fishing Boat

MSE=92.64

Recovered Fishing Boat, S=50

Results of Filtering on MSE for Fishing Boat

From the results of the plot we can see that median filtering helped reduce the MSE for the recovered image when the number of sensed pixels for every 8×8 block was low, with S=10 being the only instance where median filtering improved the MSE. Following this, as more sensed pixels form part of the recovered image, median filtering begin to hurt the MSE scores because original values of the image are replaced by the median filter, which will be penalized in the calculation of the MSE. From a qualitative standpoint, we can see that as S increases the recovered image becomes more and more clear, resembling the original image more closely. As this occurs, the blurriness of the filtered image hurts the quality of the image recovery rather than improving it.



Nature: S=10

For this case where 10 pixels are sensed out of every 256 in the 16×16 block, median filtering improves the MSE. Visually, it is difficult to make out the benefits that this filtering provides as both the unfiltered and filtered images do not capture any of the details of the original nature image. With predictions based on only 10/256 of the pixels in each block, we do not expect high accuracy, so the median filter helps replace each pixel with what may be a more conservative, and in this case likely estimate for this pixel.

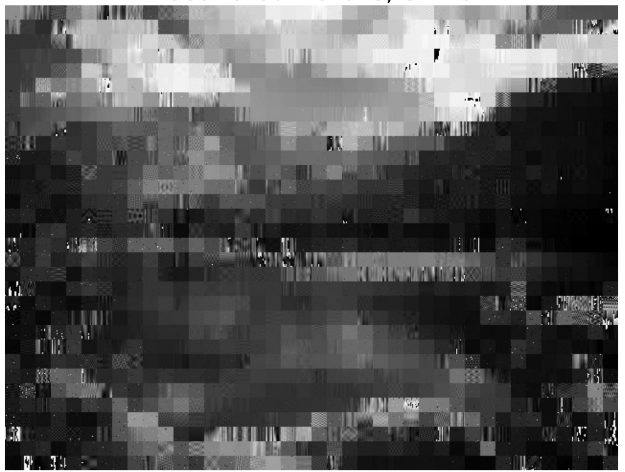
MSE=816.00

Recovered and Filtered Nature, S=10

Original Nature Image

MSE=892.80

Recovered Nature, S=10



Nature: S=30

For S=30, the MSE is no longer lower for the filtered nature image than for the unfiltered image. Qualitatively, the images appear rather similar, the filtered version simply appearing a more blurred out version of the unfiltered image. With few noticeable qualitative improvements, and knowing that the MSE for the pixels that are sensed and not predicted in the unfiltered image will be 0 whereas they may not be after median filtering, a higher MSE can be expected.

MSE=587.88

Recovered and Filtered Nature, S=30

Original Nature Image

MSE=574.69

Recovered Nature, S=30

Nature: S=50

For the same reasons as described in the fishing boat filtered images as well as in the S=30 case for nature in the previous slide, we can expect a higher MSE for the filtered images as we increase in number of sensed pixels for the recovered image. While the filtering smooths the blocks of the image and eliminates some of the graininess inside of the blocks, overall the unfiltered image appears to preserve more detail.

MSE=514.92

Recovered and Filtered Nature, S=50

Original Nature Image

MSE=454.85

Recovered Nature, S=50

Nature: S=100

There is a much larger difference in MSE in the filtered and unfiltered recovered nature images here, which makes sense qualitatively. By now, our system is able to predict the 156 pixels in each block that are unknown in this case with a high degree of accuracy, so the blurriness induced by the median filter only hurts the MSE.

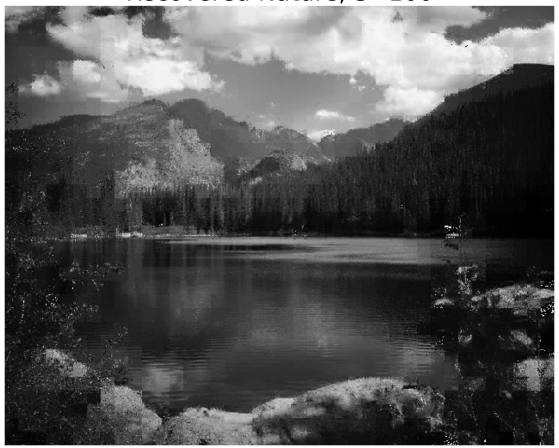
MSE=404.28

Recovered and Filtered Nature, S=100

Original Nature Image

MSE=292.33

Recovered Nature, S=100



Nature: S=150

While the filtered image in this case provides a relatively accurate semblance of the original nature image, the recovered image simply captures much more details and nuances of the original image. Looking at the filtered nature image we can see it appears rather dull in comparison to the crisp unfiltered image. Another consideration is that the median filter uses zero padding to calculate the median at the boundaries, so at the corners the median will tend to be much lower than what the pixel of the original image at that location might actually be.

MSE=310.54

Recovered and Filtered Nature, S=150

Original Nature Image

MSE=172.80

Recovered Nature, S=150

Nature: S=200

The same explanation for the past two slides holds here. The system is able to calculate the missing pixel values with a high degree of accuracy, so the blurring of the filtered image makes the image lose details of the unfiltered image that exist in the original image. Additionally, since 200/256 pixels of every block will have a MSE of 0 for the unfiltered image, we expect the filtered image to have a worse MSE.

MSE= 240.09

Recovered and Filtered Nature, S=200

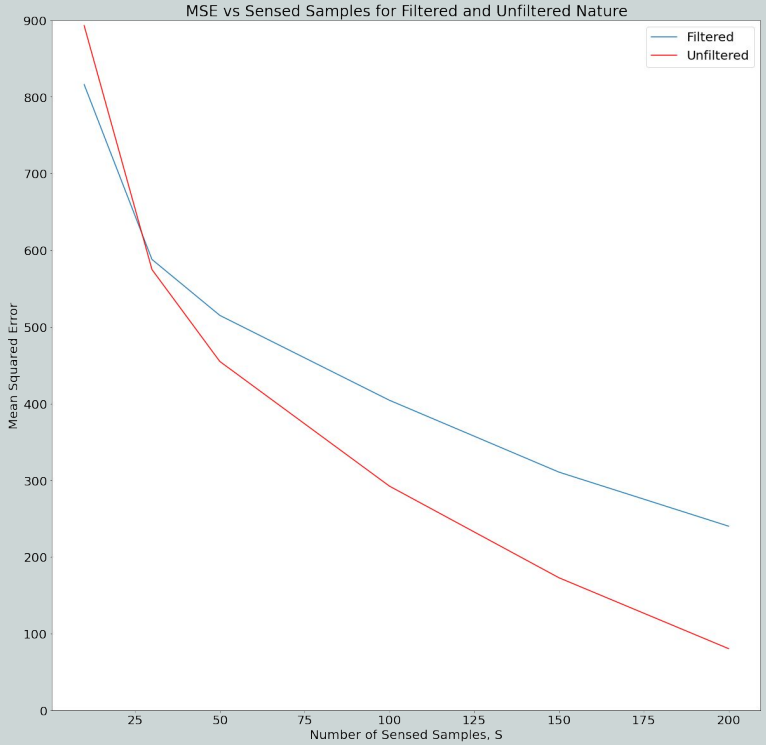
Original Nature Image

MSE= 80.40

Recovered Nature, S=200

Results of Filtering on MSE for Nature

Similarly to the fishing boat image, for S=10 median filtering helps bring down the MSE below that of the unfiltered image. Beyond this the MSE becomes worse for the filtered image compared to the unfiltered image at an increasing rate, which we expect as the image recovery system becomes better at predicting the corrupted pixels when there are more sensed pixels available from which to formulate a prediction. Additionally, when calculating MSE for the entire filtered and unfiltered image relative to the original, the unfiltered image will have an error of 0 for every sensed pixel, which may not be the case for the filtered case. This issue gets increasingly worse as more pixels are sensed and may explain the increasing difference in MSE as S increases.

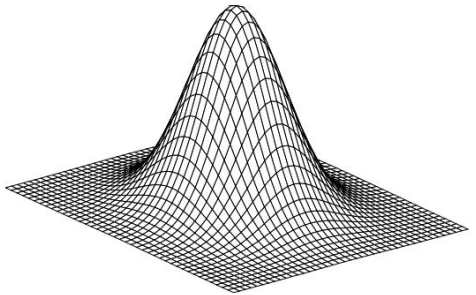


Explorations: Gaussian Filtering

Gaussian Filters

The Gaussian kernel is the most commonly used smoothing function in computer vision, and since our aim in using a median filter is to smooth the signal to remove erratic pixel predictions, we explore filtering the image using a sequence of 1-D convolutions with a Gaussian kernel.[6] This implementation from scipy.ndimage is acceptable because convolution is a linear operation, and the 2-D Gaussian kernel is separable, so convolving an image in the row direction, then convolving the result in the column direction with a 1-D Gaussian kernel is equivalent to convolving the image with a 2-D Gaussian kernel. The shape of the Gaussian allows for smoothing of the image where the pixel being replaced contributes the most to the value that will replace it, along with those nearest to it, while further pixels will contribute less. The sigma value, or the standard deviation of the Gaussian can be chosen to determine how much smoothing will occur in the image, with higher values of sigma corresponding to wider Gaussians and thus more smoothing in the image.

$$G(v,u) \stackrel{\text{def}}{=} e^{-\frac{1}{2}\frac{u^2+v^2}{\sigma^2}} \quad .$$



Images cited in [6]

Gaussian Filter: Fishing Boat, S=10

The Gaussian filtered image score better than the median filtered image, with a 7.9% improvement from the median filter when sigma=0.5, and a 15.6% improvement from the median filter when sigma=1. Qualitatively, the Gaussian filtered image when sigma=0.5 retains some of the DCT basis function appearance in the blocks, but smoothed over slightly, while when sigma=1 the figure is much more smoothed over.

Original Fishing Boat

MSE=844.21

Recovered Fishing Boat, S=10

MSE=791.13

Recovered and Filtered Fishing Boat, S=10

MSE=728.96

Gaussian Filter, S=10, Sigma=0.5

MSE=667.66

Gaussian Filter, S=10, Sigma=1

Gaussian Filter: Fishing Boat, S=30

The Gaussian filtered images here again have lower MSE than the median filter, with a 32.0% improvement for sigma=0.5 and a 17.6% improvement for sigma=1, however we see the Gaussian filtered image for sigma=1 perform just worse than the unfiltered image here. Qualitatively, the Gaussian filters appear to smooth the images in a cleaner way compared to the median filter while preserving more detail.

Original Fishing Boat

MSE=345.21

Recovered Fishing Boat, S=30

MSE=421.78

Recovered and Filtered Fishing Boat, S=30

MSE=286.70

Gaussian Filter, S=30, Sigma=0.5

MSE=347.59

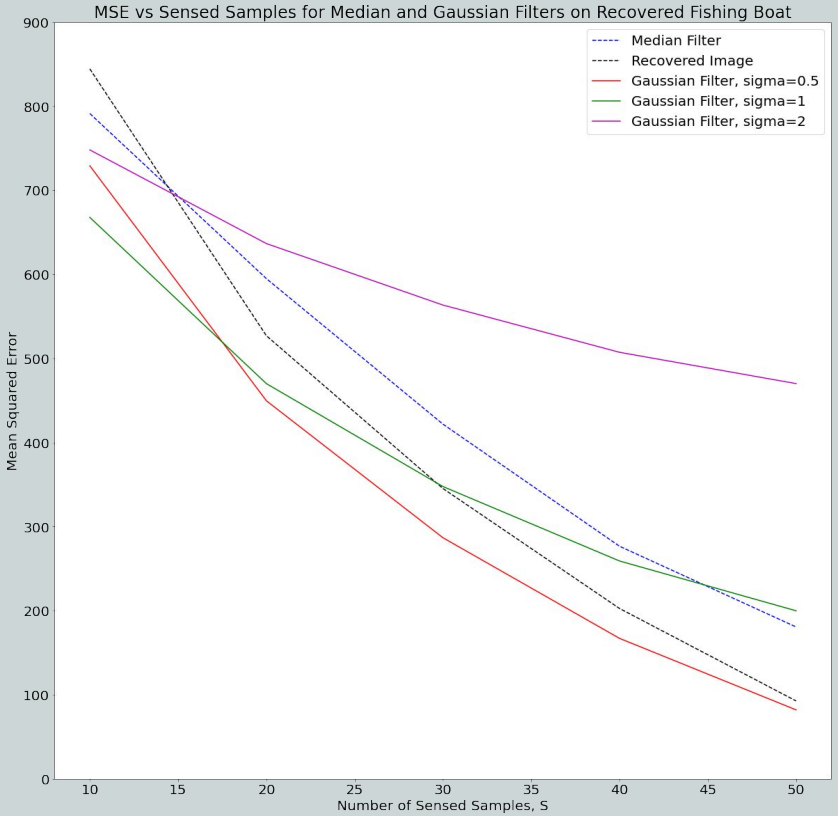
Gaussian Filter, S=30, Sigma=1

Gaussian Filter: Fishing Boat, S=50

Here we see the Gaussian filters again outperform the median filter, with sigma=0.5 having a 62.0% lower MSE with sigma=1.0 having a 7.4% lower MSE. The Gaussian filtered image with sigma=0.5 slightly outperforms the MSE of the unfiltered image, with fewer erratic pixel predictions and an overall smoother feel to the image.

Results of Gaussian Filtering for Fishing Boat

From this plot and the previous results, we can see strong improvements on the MSE from the Gaussian filtered images compared to the median filtered images, in particular for the case where sigma=0.5. This parameter seems low enough to be able to preserve localized information at each pixel in the image which in turn also preserves some finer details that get lost in median filtering as well as in Gaussian filtering with higher sigma values. The filter is still able to remove some of the erroneous pixel predictions however, and these qualities of smoothing while preserving detail help improve the quality of the image, even beyond that of the recovered image when sigma=0.5.



Gaussian Filter: Nature, S=10

We can see from the MSE scores that the Gaussian filter is 0.7% worse when sigma=0.5 and 8.3% better than the median filter for this case. This is somewhat expected, as this only 10/256 pixels known per block, we will want to smooth this case more than others to make more conservative estimates for each pixel. Qualitatively, the differences are difficult to make out among any of the four images shown below, except that the filtered images show some signs of smoothing.

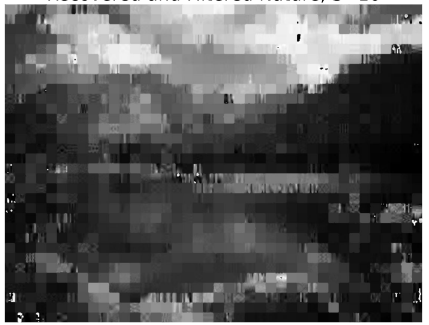
Original Nature Image

MSE=892.80

Recovered Nature, S=10

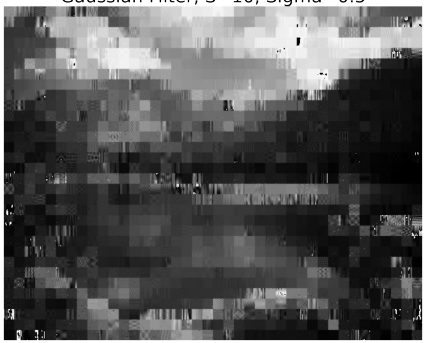
MSE=816.00

Recovered and Filtered Nature, S=10



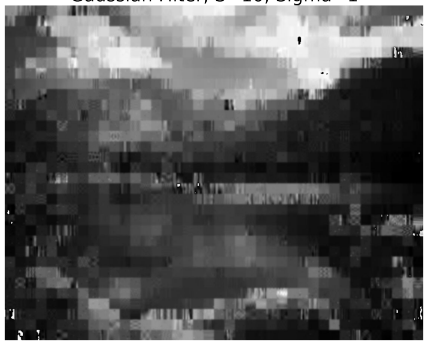
MSE=821.99

Gaussian Filter, S=10, Sigma=0.5



MSE=748.60

Gaussian Filter, S=10, Sigma=1



Gaussian Filter: Nature, S=50

In this case, both the Gaussian filtered images improve the quality of the image reconstruction compared to the unfiltered and median filtered images. With sigma=0.5, the MSE is improved by 17.4% and with sigma=1 the MSE is improved by 14.5% compared to the median filtered image. Qualitatively, we can see that the Gaussian filtered images do capture more details of the original image and resemble the original more than the median filtered image.

Original Nature Image

MSE=454.85

Recovered Nature, S=50

MSE=514.92

Recovered and Filtered Nature, S=50

MSE=425.45

Gaussian Filter, S=50, Sigma=0.5

MSE=440.80

Gaussian Filter, S=50, Sigma=1

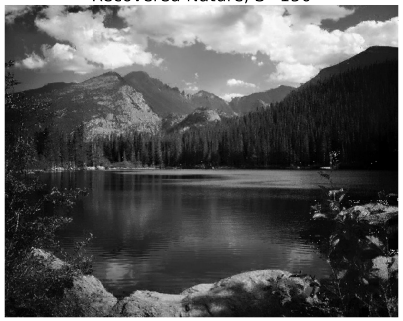
Gaussian Filter: Nature, S=150

The MSE is improved by 45.2% for sigma=0.5 and by 15.4% for sigma=1 compared to the MSE of the median filtered image. With S=150, the Gaussian filtered performs slightly better than the unfiltered image with sigma=0.5, but sigma=1 performs worse. Qualitatively we see the images overall look pretty similar, with some erratic pixels smoothed out by the filtered images.

Original Nature Image

MSE=172.80

Recovered Nature, S=150



MSE=310.54

Recovered and Filtered Nature, S=150

MSE=170.04

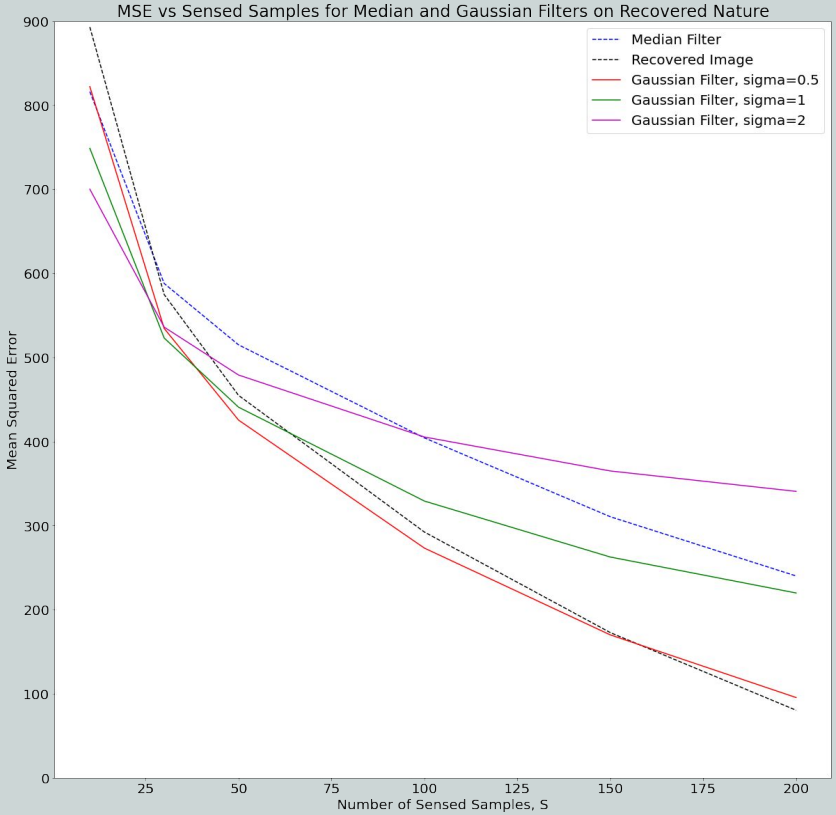
Gaussian Filter, S=150, Sigma=0.5

MSE=262.62

Gaussian Filter, S=150, Sigma=1

Results of Gaussian Filtering for Nature

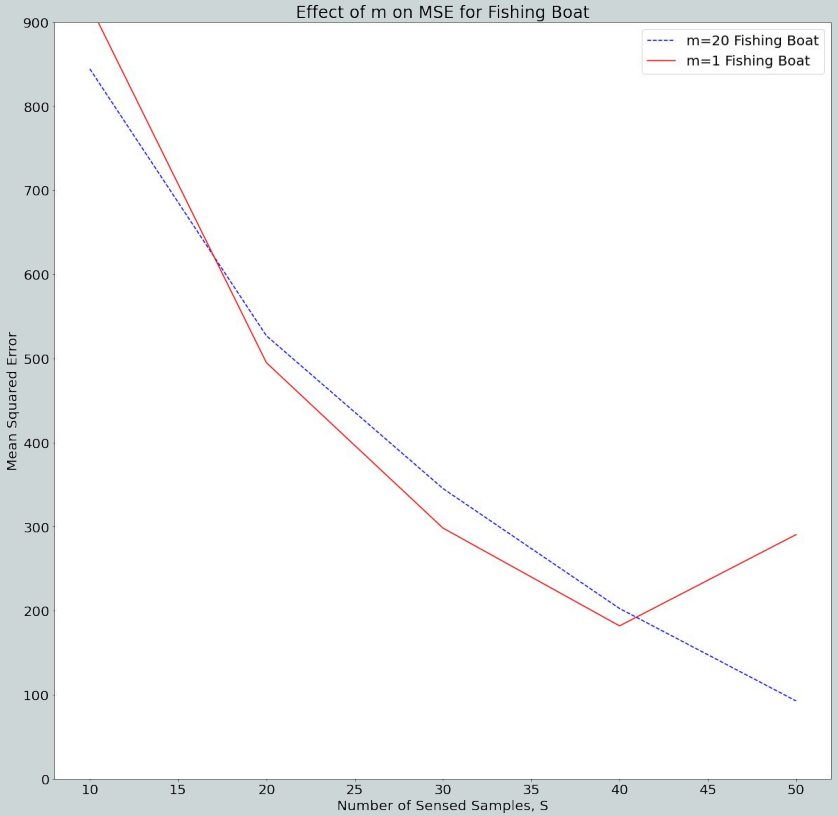
From the plot we see that the Gaussian filter with sigma=0.5 tends to perform slightly better at recovering the original image until when S is about 155 pixels per block. For most values of S, the Gaussian filters with sigma=0.5, 1 perform better than the median filter, likely because the ability to capture the local information of the pixel as well as the information of the surrounding pixels is more useful in pixel prediction than the approach of the median filter. The Gaussian filter with sigma=2 does not perform nearly as well as the other Gaussian filters, except for very low values of S, likely because this image smooths the image too much, which is useful with very poor predictions, but is less useful as our predictions improved.



Explorations: Costs and Benefits of m Lambda Training Iterations

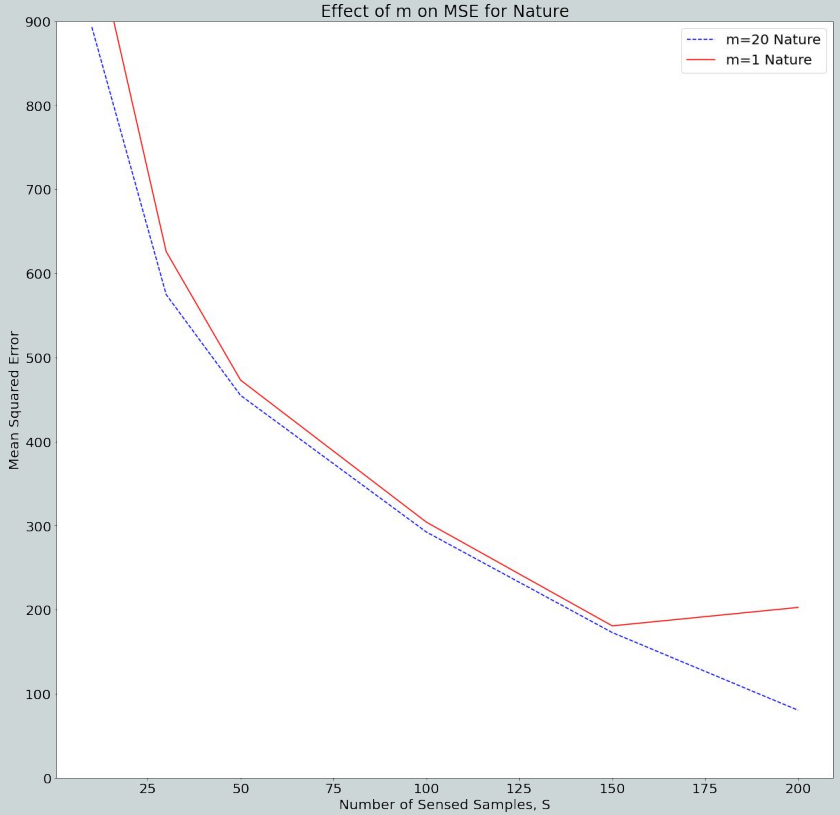
Effects of m on MSE for Fishing Boat

We would expect performing random subset cross validation m=20 times to find a lambda parameter for LASSO would consistently provide us with a lower MSE than finding a lambda value using cross validation where m=1. These results represent only one trial of this experiment, which should ideally be ran several times to determine a true relationship with more confidence. However, this rejects our expectation of the importance that cross validation would play to lower the MSE. While we expect that after many trials, this experiment would reveal that the MSE tends to be lower for m=20 iterations of cross-validation, it may not be by as much as we think. In terms of runtime, the program runs about 20 times slower for m=20 than for m=1, so there is a certain benefit to running simulations with m=1 especially when making changes to the system and wanting to see quick results.



Effects of m on MSE for Nature

For this plot, we can draw many of the same conclusions as in the last slide. Although this experiment requires more trials to be able to draw definite conclusions from these relationships, we are surprised that there is not a larger difference for the MSEs between the case where m=20 and m=1 , where m is the number of times cross validation is repeated to find lambda. This is useful information in particular when changes are being made to the system and we want to view the results of these said changes faster.

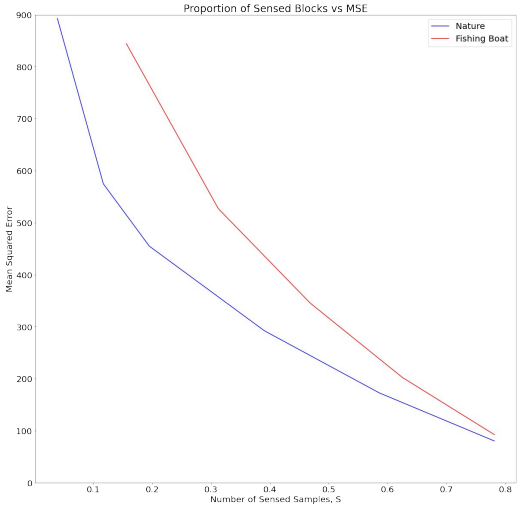


04

Conclusions

Factors Affecting Image Recovery: S

The most important factor affecting the quality of our image recovery was S, the number of sensed pixels for each NxN block. We predicted that this relationship would be dependent on the proportion of S over the total number of pixels per block, but from the figure on the right we can see this only begins to appear true when S is about 0.5 the total number of pixels in the block, and as S approaches 0.8 the total number of pixels in the block only then does the MSE appear dependent on the proportion of pixels sampled rather than just the number of pixels sampled. The reason that for smaller proportions of pixels sampled that image recovery may be more accurate is that this approach of predicting the DCT coefficients using LASSO regression may only need a certain quantity of data to begin making accurate predictions of the DCT coefficients, as a opposed to a quantity of pixels that is a relatively significant proportion of the total number of pixels in the image. If a certain number of pixels per block is sufficient to determine patterns in the block and predict the DCT coefficients, this could explain the shape of this plot for lower proportions of S. All in all, as S, the proportion of sensed samples increased, the accuracy of the predictions increased as well, with a smaller slope for the larger nature image.



Factors Affecting Image Recovery: Filtering

Median filtering appeared to be an effective way of increasing the image recovery accuracy for images with low S, or sensed pixels per block. The reason for this is that with so few pixels available from which to base a prediction, there are many erratic pixel predictions present in the recovered image, which can be smoothed out by filtering. However, as the number of sensed pixels per bock increases, the system is able to make increasingly more accurate predictions on the missing pixels, and median filtering begins to hurt the recovery of the image rather than helping it. In a sense it over-smooths the image, blurring it and losing detail that improves the quality of the reconstructed image. In addition, the formulation of the MSE should be taken into account, as it is found by taking the error of every pixel in the recovered images with those of the original images. Median filtering is penalized strongly in such a formulation, as the sensed pixels in the unfiltered recovered image would each have an error of zero, while in median filtering it is likely these values would be replaced by some other pixel, thus having these pixels now contribute to the error. All in all it appears both quantitatively and qualitatively the median filtering should only be used for the lowest values of S. On the other hand, Gaussian filtering with sigma=0.5 proved highly effective in smoothing the image signal, and yielded improved values of the MSE for nearly every value of S for both images. The reason for these more successful results is that the Gaussian filter provides the benefits of smoothing in a less drastic way, not replacing one pixel by another entirely, but instead by replacing the pixel by a weighted sum of itself and the pixels around it, and this approach allowed the filtering to capture more detail while fixing unwanted pixel values.

Limits and Problems of Approach

We are limited in the scope of this project in a data sense, since we are only given two images with which to work with and this limits the range of conclusions we are able to draw and relationships we are able to establish. Without a broader set of diverse images, it is difficult to know if the conclusions we draw for these two images apply to other images that we have not yet worked with. As our approaches were not aimed at these specific images, and did not change our approach to the problem for each image except for changing the block size, we believe that the conclusions we have made from studying this system on these two images are valid and generalizable, but we would have more confidence in this if we had more images with which we could confirm our existing beliefs and potentially develop new ones.

One problem of our approach was with our use of median filtering to change the entire recovered image, since MSE penalizes the median filtering method of replacing each pixel with the median of its immediate neighborhood. As explained prior, the error of any sensed pixel would be zero for the unfiltered image and after filtering each of these pixels would likely contribute to the error. It would seem to make more sense to use median filtering on a copy of the recovered image, but then only replace the unsensed, or corrupted pixels with the corresponding pixels in the median filtered image. I believe this would reduce erratic pixel estimates and reduce the blurriness associated with the median filtered images.

Potential Improvements for an Ideal System

One way to improve our system to create an ideal system for image recovery would be to research what the ideal block sizes are for certain kinds and sizes of images to be able to select this properly as a parameter for a particular image. Another improvement that could be made is an investment in a computer that could run these simulations more rapidly, so that updates to the design and tinkering with parameters could be done more easily. Programming this project so that many of the operations can be run in parallel and thus be run on a GPU rather than a CPU could help in the development of the system in the future as well. Such improvements could be used to run many more simulations with many more images to finely tune parameters such as the number of times cross validation should be ran to select a lambda for each block, what sigma value is ideal in Gaussian filtering of images, and a more dense set of lambdas through which to iterate for lowest error for each block. Finding this sort of information and storing this in a system could allow for implemented conditionality that can select the right parameters for the right sorts of problems, and thus the system would be optimized for many images that it may encounter.

To improve the current system, the most immediate steps would be to collect more images with which to run simulations on and draw inferences from these results, experiment with filtering where the filtering replaces only the corrupted pixels as opposed to all of the pixels, and continuing to experiment with Gaussian filters and other filtering techniques to find what modeling choices allow for the best image recovery.

Collaborators

Paul Truitt

Shared ideas while working through code and slidedoc

Remy Cross

Worked together to solve coding issues and talked through concepts in slidedoc

Arman Shekarriz

Discussed concepts while working through slidedoc

Thank You

CREDITS: Diese Präsentationsvorlage wurde von **Slidesgo** erstellt, inklusive Icons von **Flaticon** und Infografiken & Bilder von **Freepik**

Bitte lösche diese Folie nicht, es sei denn du bist ein Premium Nutzer

References

- [1] Project description provided by Dr. Tantum
- [2] Roberts, Eric. “Lossy Data Compression: JPEG.” Accessed March 2, 2022. https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossy/jpeg/dct.htm.
- [3] Watson, Andrew B. “Image Compression Using the Discrete Cosine Transform,” 1994, 17.
- [4] C. Zhao, S. Ma and W. Gao, "Image compressive-sensing recovery using structured laplacian sparsity in DCT domain and multi-hypothesis prediction," 2014 IEEE International Conference on Multimedia and Expo (ICME), 2014, pp. 1-6, doi: 10.1109/ICME.2014.6890254.
- [5] Tibshirani, Robert. “Regression Shrinkage and Selection via the Lasso.” Journal of the Royal Statistical Society. Series B (Methodological) 58, no. 1 (1996): 267–88. http://www.jstor.org/stable/2346178.
- [6]Tomasi, Carlo. “Correlation, Convolution, and Filtering,” 2022, 13.
- Numpy: 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. (Publisher link).
- Matplotlib.pyplot: J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.
- Scikit Learn: Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
- Pandas: Data structures for statistical computing in python, McKinney, Proceedings of the 9th Python in Science Conference, Volume 445, 2010.
- Scipy: Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272.