Compressed Sensing Image Recovery

Ian Hanus

Background

- Missing/corrupted pixels in an image can lead to loss of information as well as interrupt the attention of the viewer, drawing their focus away
- Images are expensive to store, taking up memory that could be used elsewhere
- Pixels are expensive to acquire in an image
  - What if instead of acquiring and compressing, we could only take the values needed to store the image?
- Solution: compressed sensing!
  - Relies on the sparsity of images in the discrete cosine transform (DCT) domain

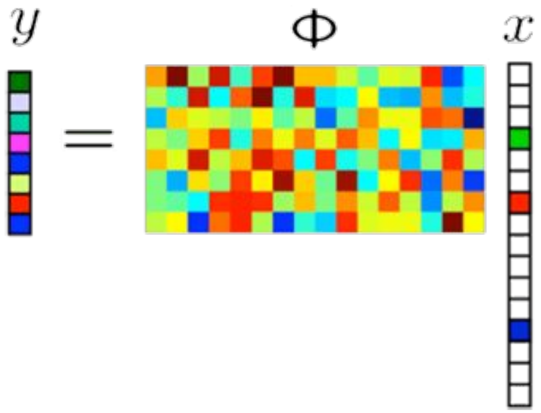
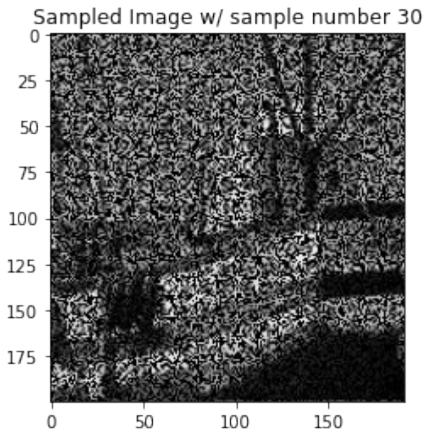


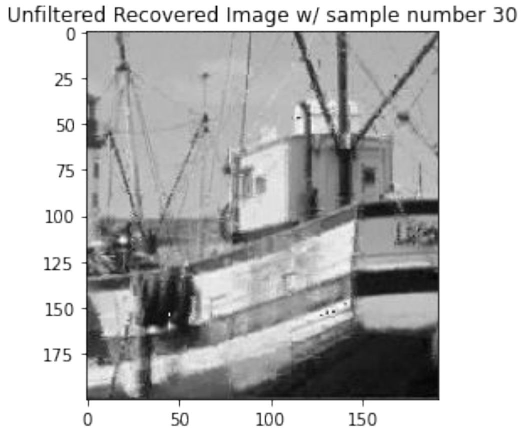
Fig. 1: Visualization of Compressive Sensing, with y=reconstructed pixels, x=DCT coefficients, and ϕ=Transformation matrix http://dsp.rice.edu/2018/08/22/compressive-sensing-resources-are-back/

Introduction

- Project goals
  - Reconstruct a corrupted image using compressed sensing image recovery
  - Be able to explain the mathematical foundation of compressed sensing
  - Evaluate the effectiveness of reconstruction attempt
  - Understand the effect of a median filter applied post reconstruction
- Results
  - Images were able to be effectively reconstructed for higher sample rates (number of valid pixels in the corrupted image), but low sample rates led to high error
  - Median filters were helpful for lower sample rates, but less effective and sometime harmful for higher sample rates



Corrupted boat image w/ S=30 pixels per 8x8 block



Reconstruction of the above corrupted image using compressed sensing image recovery

Compressed Sensing

- Applying 2-D discrete cosine transform separates image into sparse, spectral sub-bands
- Image pixels can be represented as a transformation matrix * DCT coefficient

$$\underset{\textbf{Pixel}}{\underline{g(x,y)}} = \sum_{u=1}^{P}\sum_{v=1}^{Q} \underset{\textbf{Transform Matrix}}{\underline{\alpha_u \cdot \beta_v \cdot \cos\frac{\pi(2x-1)(u-1)}{2p} \cdot \cos\frac{\pi(2y-1)(v-1)}{2Q}}} \cdot \underset{\textbf{DCT Coefficient}}{\underline{G(u,v)}}$$

$$\underset{\textbf{Pixels}}{\begin{bmatrix} c_1 \\ \cdot \\ \cdot \\ \cdot \\ c_n \end{bmatrix}} = \underset{\textbf{Transform Matrix}}{\begin{bmatrix} T_{11} & \cdot & \cdot & \cdot & T_{1N} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ T_{N1} & \cdot & \cdot & \cdot & T_{NN} \end{bmatrix}} \cdot \underset{\substack{\textbf{DCT} \\ \textbf{Coefficients}}}{\begin{bmatrix} \alpha_1 \\ \cdot \\ \cdot \\ \cdot \\ \alpha_n \end{bmatrix}}$$

Compressed Sensing (cont.)

- Each column of the transform matrix is a basis function of length P*Q in the DCT domain, calculated from the equation on the previous slide
- Removing the missing pixels from the flattened block array, as well as the rows corresponding to the indices of the missing pixels in the transformation matrix, allows us to define a linear system between the basis in the DCT domain and the pixel values

$$\begin{bmatrix}\times\\ \times\\ ?\\ ?\\ \times\end{bmatrix} = \begin{bmatrix}\times & \times & \times & \times & \times\\ \times & \times & \times & \times & \times\\ \times & \times & \times & \times & \times\\ \times & \times & \times & \times & \times\\ \times & \times & \times & \times & \times\end{bmatrix} \cdot \begin{bmatrix}\times\\ \times\\ \times\\ \times\\ \times\end{bmatrix} \longrightarrow \begin{bmatrix}\times\\ \times\\ \times\end{bmatrix} = \begin{bmatrix}\times & \times & \times & \times & \times\\ \times & \times & \times & \times & \times\\ \times & \times & \times & \times & \times\end{bmatrix} \cdot \begin{bmatrix}\times\\ \times\\ \times\\ \times\\ \times\end{bmatrix}$$

- Linear system is underdetermined!
  - This is due to the missing pixels: formulated as linear system but information is missing
  - Need additional constraint to solve
  - In the DCT domain, natural images tend to be sparse
  - Use Lasso L_1 norm regularization to drive DCT coefficients towards zero

Lasso Regression

- Lasso = Least Absolute Shrinkage and Selection Operator
  - Regularization technique
  - Similar to linear regression, but calculated coefficients are given a cost in optimization (l1 penalty)
  - Cost to be minimized is the RSS plus the sum of the magnitude of the weights times a regularization constant (λ)
- For compressed sensing, this means that the cost of the DCT coefficients is minimized
  - Try to keep DCT coefficients at zero, in line with natural images being sparse in the DCT domain!
  - Now a convex optimization problem to find the optimal DCT coefficients
- Once the optimal DCT coefficients are found, apply the inverse DCT transform to recover the predicted pixels!
- The L_1 norm is used so that the cost of outliers is linear instead of the L_2 norm, which would have exponential cost
  - L_1 used so that less important coefficients are driven to zero, whereas Ridge would simply drive the highest values lower

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Median Filtering

- Especially for lower sample corrupted images, there can be rather jarring differences between neighboring pixels
- To solve this we apply median filtering
- Median filter is applied by averaging a pixels value w/ it’s neighbors
  - This is done to provide a smoothness to the image in the visual domain
  - Nearby pixels will be less different than before, so less jarring contrast between neighboring pixels
  - Makes sense because outside of explicit object boundaries, nearby pixels are similar colors in images

$$F[x, y] = (\sum_{i=-1}^{1} \sum_{j=-1}^{1} I(x-i, y-j))/9$$

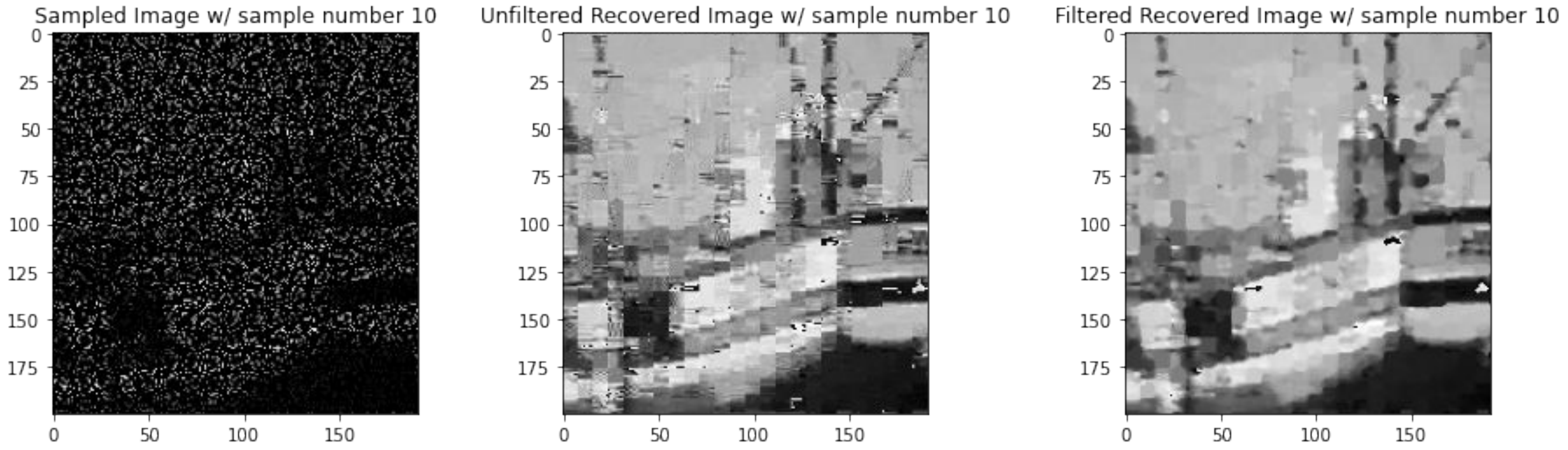
Results

- The images were broken down into small blocks (8x8 for the fishing boat, 16x16 for the Lena woman). The boat images were sampled such that 10, 20, 30, 40, and 50 pixels were not "corrupted" and the Lena images were sampled such that 10, 30, 50, 100, and 150 pixels were not "corrupted"
- The linear system was constructed as stated in the mathematical formulation for each of these blocks, and the sklearn.linear_model.Lasso tool was used for convex optimization. 50 α values from 10^{-6} to 10^{6} were considered, using GridSearchCV to find the optimal α value for each block and cross-validation done using 20 random subsets, with training data making up 5/6ths of the data and testing making up 1/6th
- The predicted DCT coefficients were estimated using the chosen α, and the inverse transform was applied to each block. The blocks were then reconstructed into the recovered images that will be shown.
- Recovery error was stated to be the mean squared error between the recovered image and the original image

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

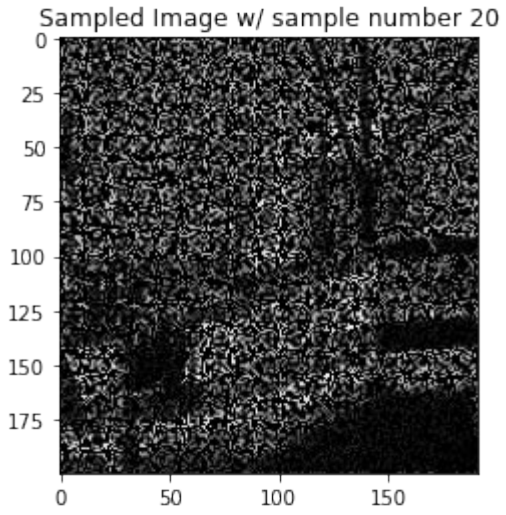
Fishing Boat (Sample 10)

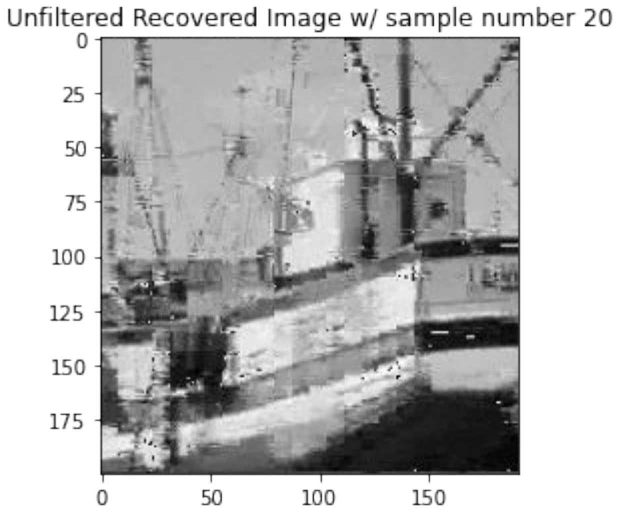
- All of the following examples will follow the format: sampled image, unfiltered reconstruction, and filtered reconstruction
- Here we see the fishing boat with 10 samples. This performed the worst of all of the sampled images, as it had the least information, with MSE_unfiltered = 898.57 and MSE_filtered = 692.72

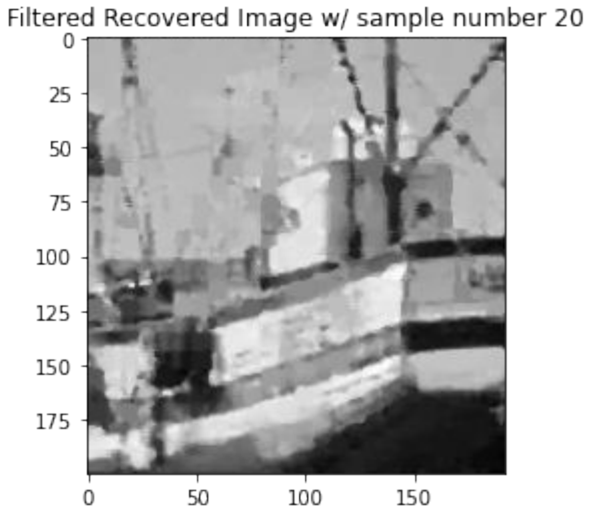

Sampled Image w/ sample number 10
Unfiltered Recovered Image w/ sample number 10
Filtered Recovered Image w/ sample number 10
0
25
50
75
100
125
150
175
0
50
100
150

Fishing Boat (Sample 20)

- There was a great improvement in MSE with doubling of the sample pixels, with the MSE_unfiltered = 494.72 and MSE_filtered = 394.87

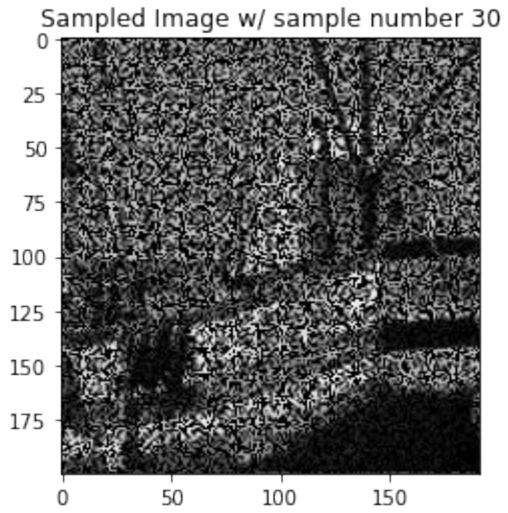


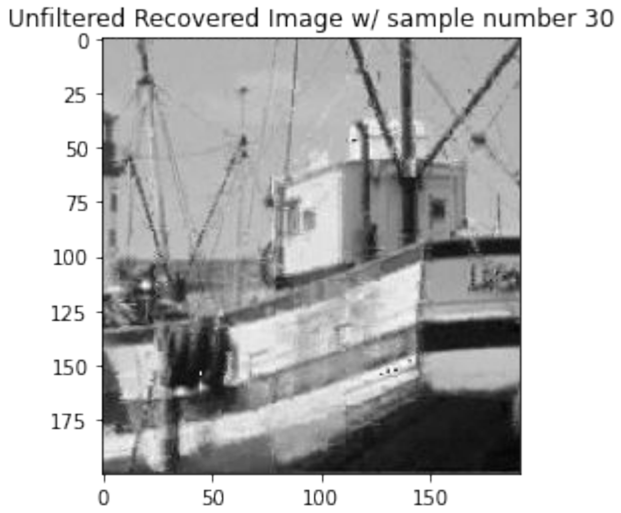


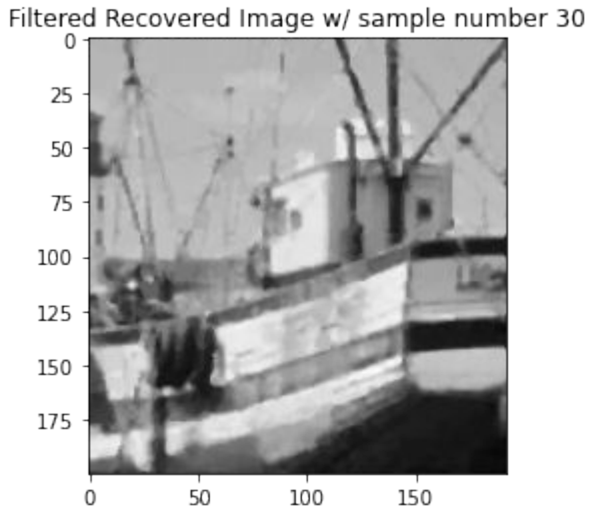


Fishing Boat (Sample 30)

- There was a slight improvement in MSE from the 20 sample experiment, with the MSE_unfiltered = 270.73 and MSE_filtered = 263.71

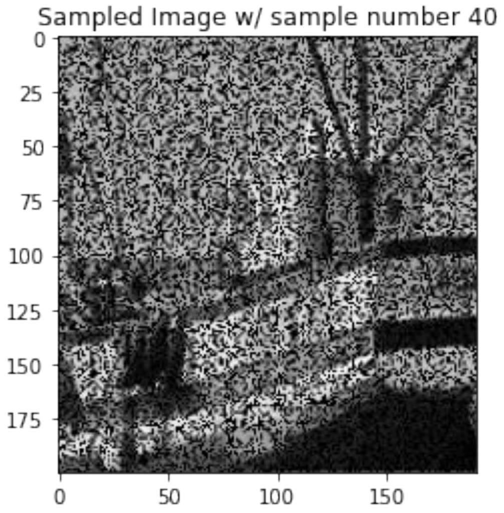


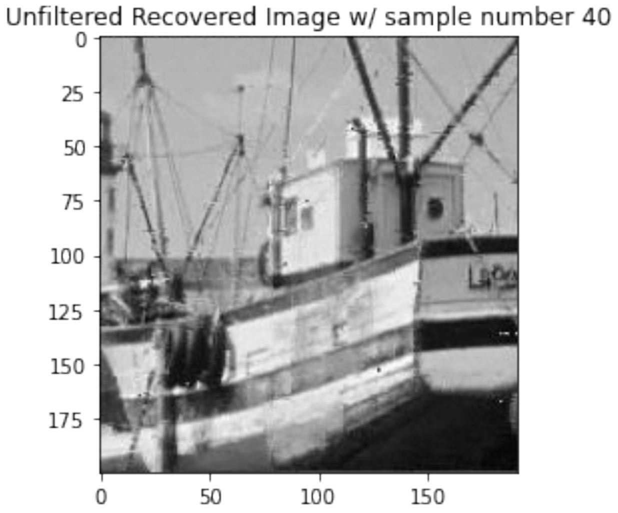


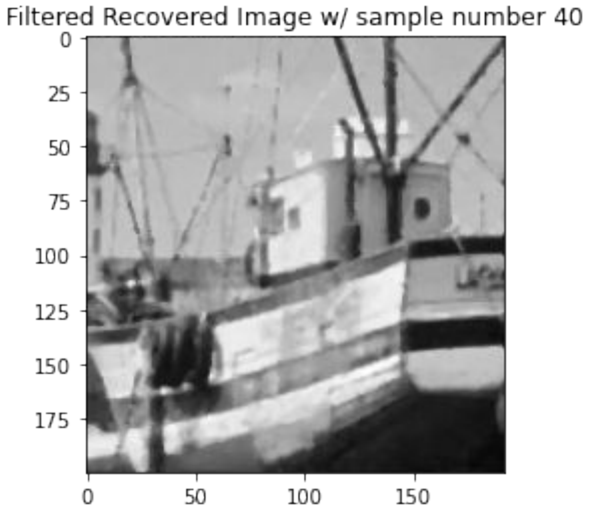


Fishing Boat (Sample 40)

- Another slight improvement, with MSE_unfiltered = 123.95 and MSE_filtered = 179.13

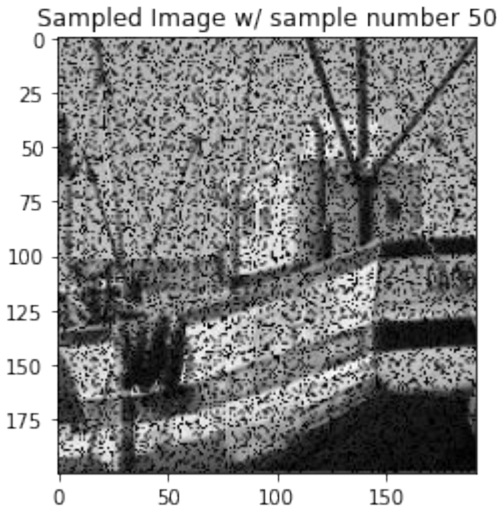


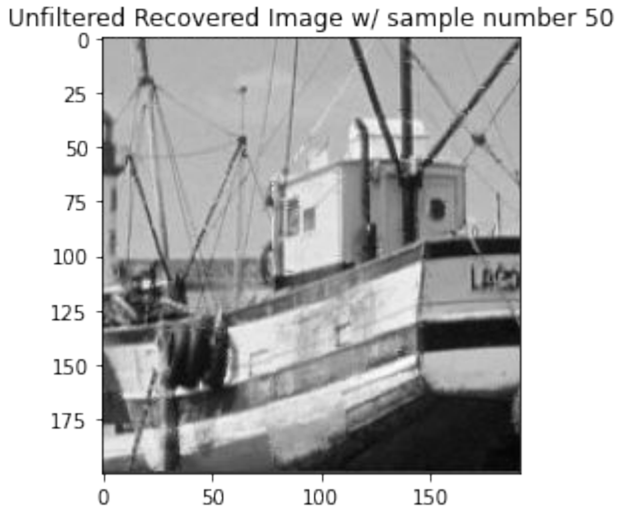


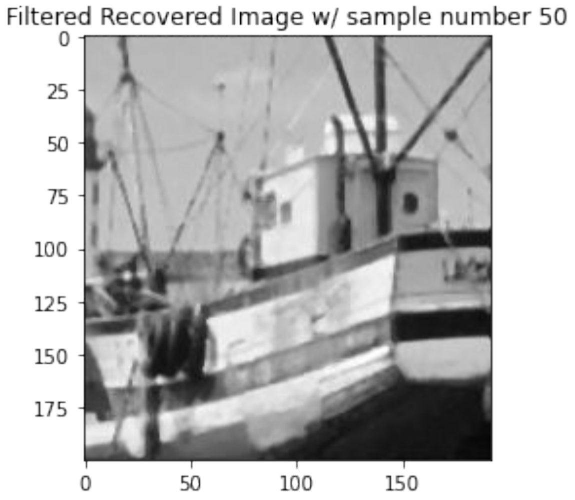


Fishing Boat (Sample 50)

- Here we the unfiltered MSE drop a decent amount, but the filtered MSE only drops slightly
- MSE_unfiltered = 48.15 and MSE_filtered = 140.93

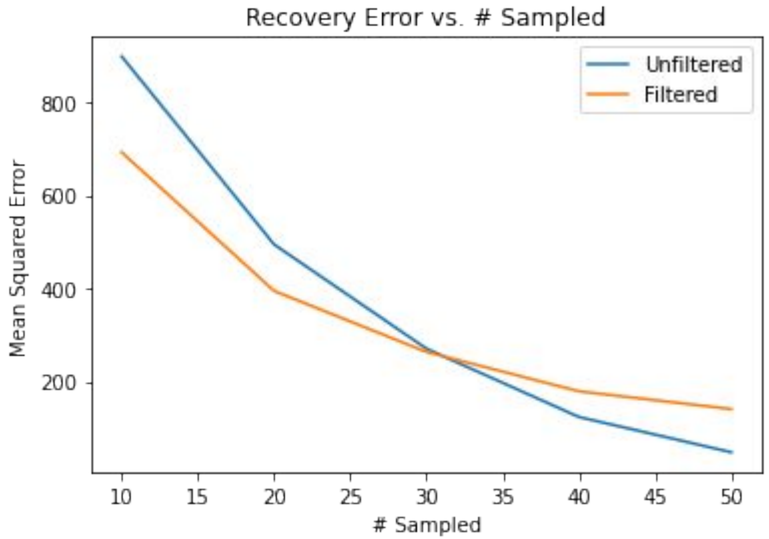






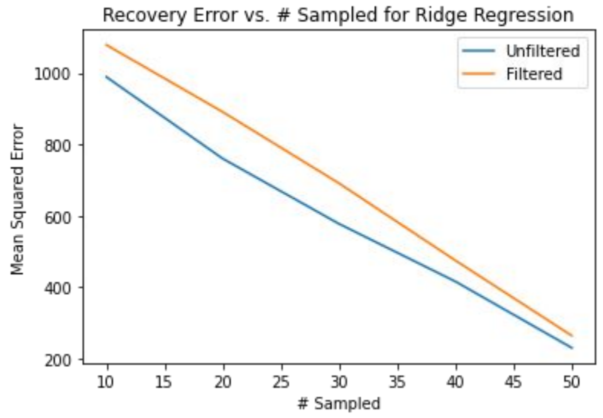
Filtered vs. Unfiltered (Fishing Boat)

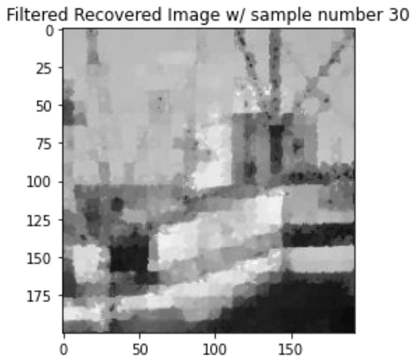
- The plot to the right shows the recovery error of the fishing boat given x pixels sampled for both the filtered and unfiltered image
- The filtered performed better when there were fewer samples, while the unfiltered performed better when there were more samples
    - Unfiltered began to perform better around 30 samples



Using Ridge Regression for Boat

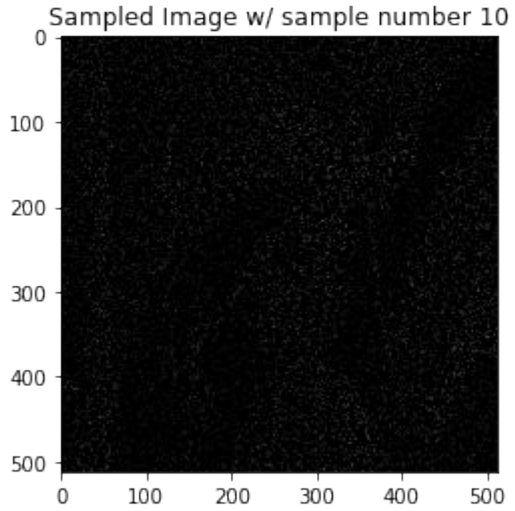
- The same conditions were applied as the previous experiment, but instead of using Lasso to establish the L_1 loss function for the DCT coefficients, Ridge was used to add L_2 normalization
- Higher loss for both filtered and unfiltered at every point compared to the Lasso run

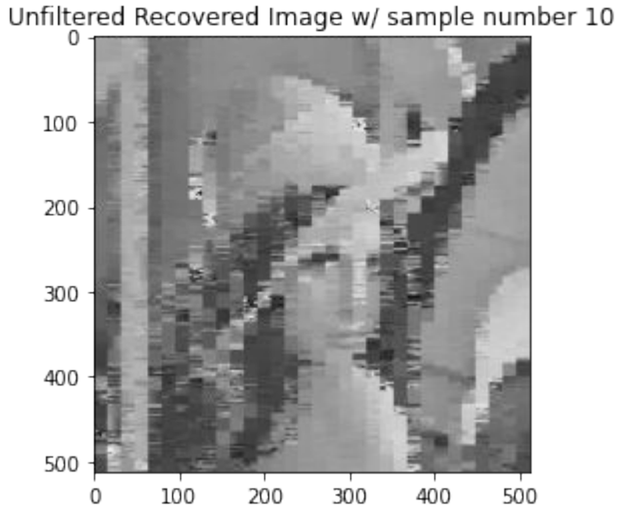


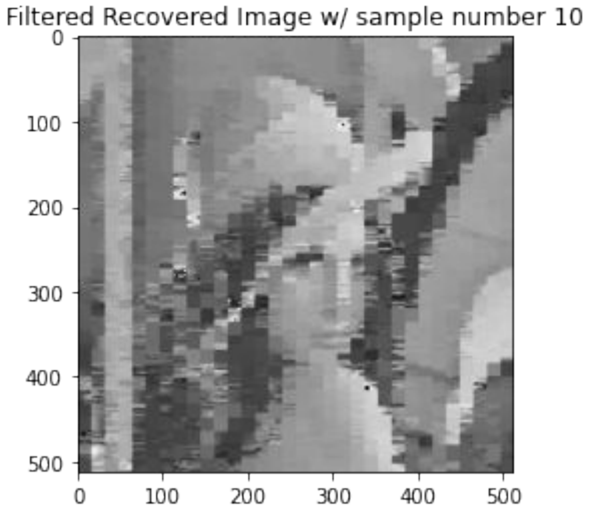


Lena (Sample 10)

- MSE_unfiltered = 582.58 and MSE_filtered = 476.87
- A poor reconstruction is helped a bit by filtering, but there is simply too much missing information to obtain a realistic reconstruction regardless of filtering. It is still possible to at least tell what the image is supposed to be



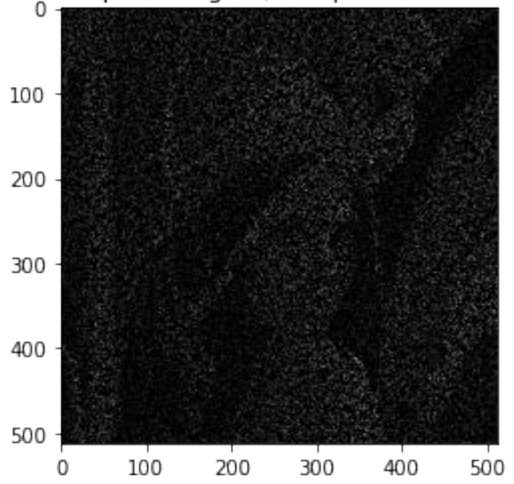




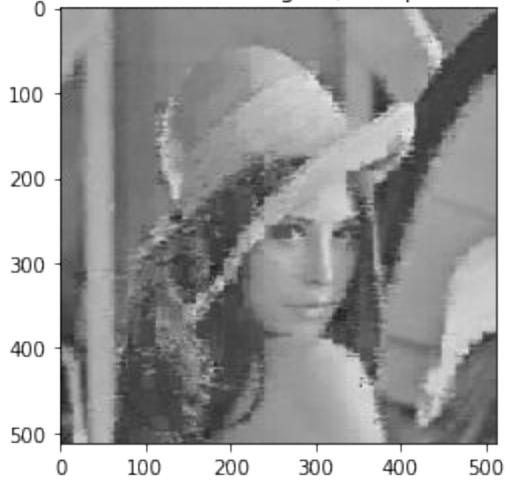
Lena (Sample 30)

- MSE_unfiltered = 324.10 and MSE_filtered = 217.15
- The image begins to take more obvious shape through the compressed sensing image recovery, and median filtering helps create a more reasonably realistic looking image

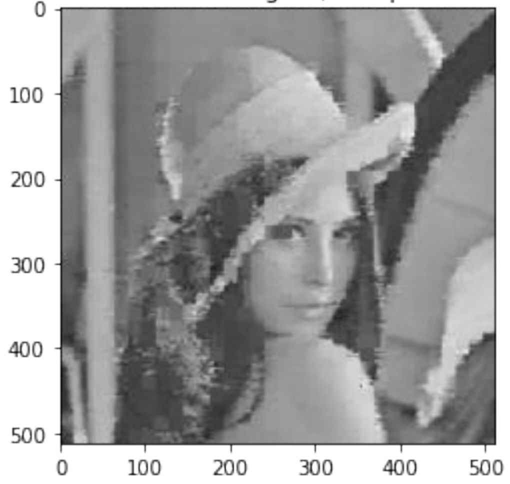
Sampled Image w/ sample number 30



Unfiltered Recovered Image w/ sample number 30

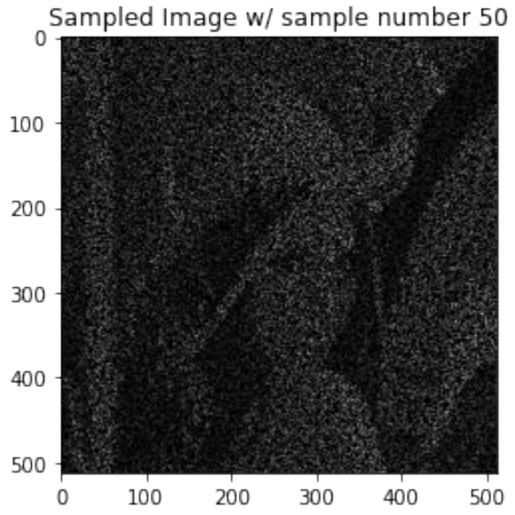


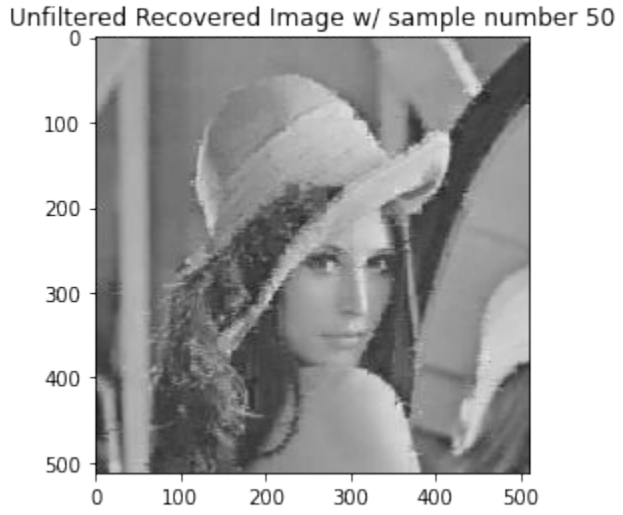
Filtered Recovered Image w/ sample number 30

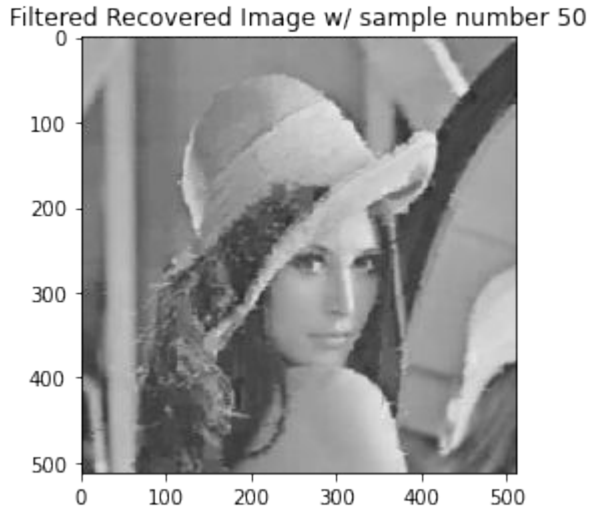


Lena (Sample 50)

- MSE_unfiltered: 189.40 and MSE_filtered = 119.45
- Applied median filtering creates a smoother, more realistic output that seems less granular

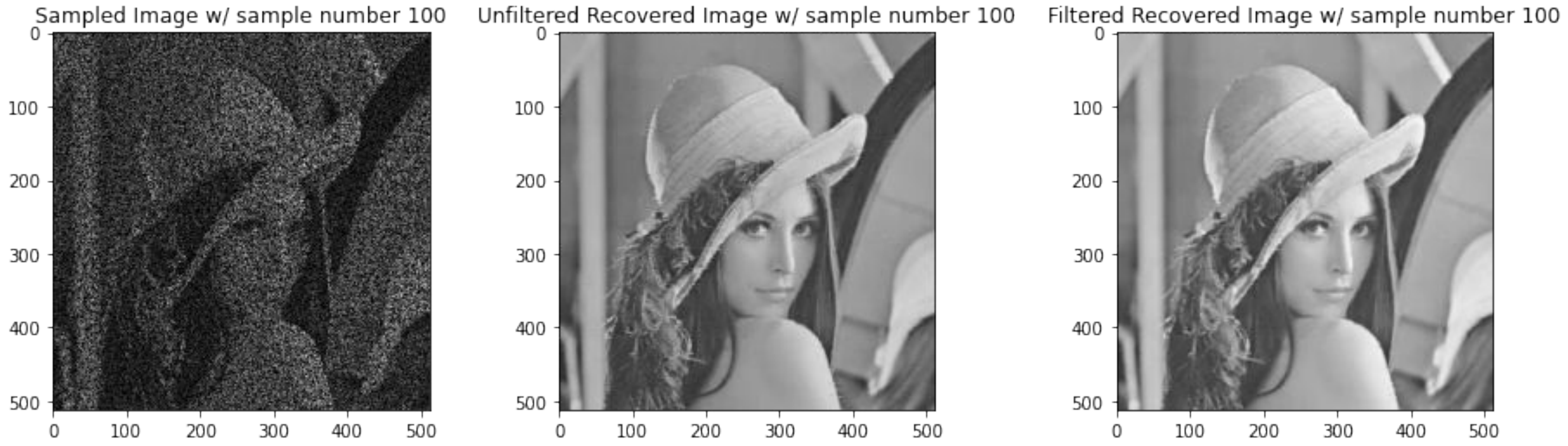






Lena (Sample 100)

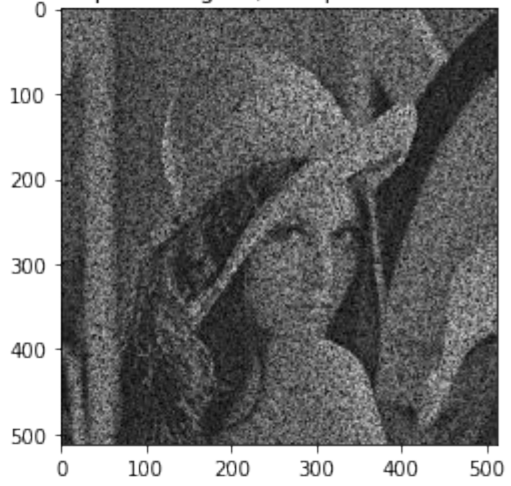
- Sampling 100 of the 256 pixels per block, we can reconstruct the image quite well, and there is only a small visible difference w/ applied median filtering
- MSE_unfiltered = 65.31 and MSE_filtered = 48.93


Sampled Image w/ sample number 100
Unfiltered Recovered Image w/ sample number 100
Filtered Recovered Image w/ sample number 100
0
100
200
300
400
500

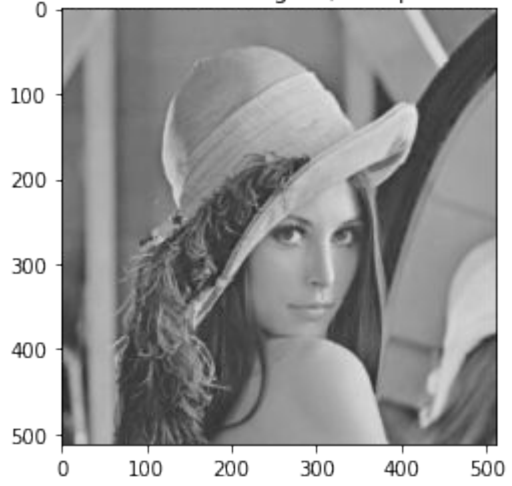
Lena (Sample 150)

- Left to right, we see that the image is reconstructed quite well and there is little noticeable change w/ applied filter
- MSE_unfiltered = 38.96 and MSE_filtered=40.41

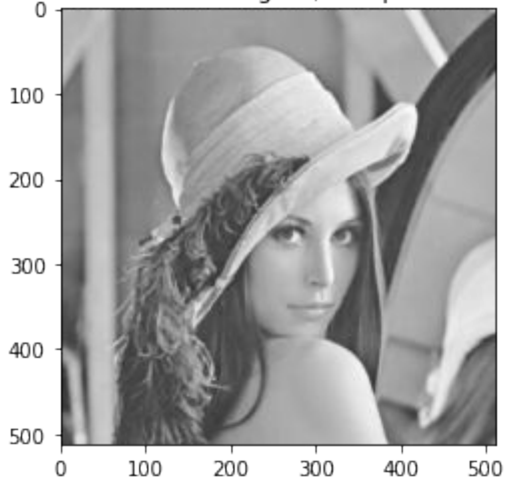
Sampled Image w/ sample number 150



Unfiltered Recovered Image w/ sample number 150

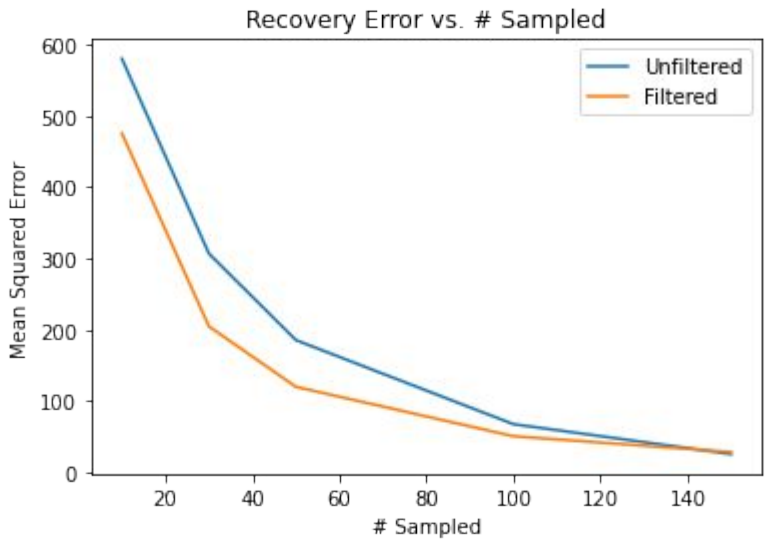


Filtered Recovered Image w/ sample number 150



Filtered vs. Unfiltered (Lena)

- The recovery error is quite high for the lower number of sampled pixels for both the unfiltered and filtered, although filtered tends to perform better
- The unfiltered reconstruction only begins to outperform the filtered reconstruction when the number of samples is around 150, at which point the difference in recovery error is quite small and the visual difference between the two is almost insignificant



Discussion

- Effect of image size on reconstruction quality
    - Reconstructions of the boat were visually effective with a sample size as low as s=30
        - Below s=30, the visualizations were too pixelated/incorrect to be seen as successes
    - Reconstructions of the Lena image were visually effective with a sample size as low as s=50
        - Median filtering helped minimize loss more effectively on this image
    - The higher the size of the image/block, the higher the sample rate needed to reconstruct the image
- Median filtering
    - Median filtering was more effective for lower sample rates
    - With less information, pixels are less likely to be correct. By applying the median filter, we drive pixels towards a smoothness condition in the visual domain, which is true outside of boundaries between objects in images
    - Leads to less sharp lines in images, but more consistently valued free space in the image
        - Even more valuable on higher resolution images

Discussion (cont.)

- Limits with the approach
  - Runtime!
    - Before any parallelization, the series of runs on the Lena image took upwards of 5 hours
    - Could also lower the regularization parameters sampled, but this could lead to finding less accurate DCT coefficients and having higher overall error
  - Assumes even distribution of pixels that must be estimated across the image
    - An 8x8 block that is missing most/all of it’s pixels will be very poorly reconstructed
  - Lasso regression is not guaranteed to find optimal DCT coefficients
    - Being a global minimum depends on the concavity of the function
- Lasso vs. Ridge Regression
  - As expected, reconstruction using Lasso regression consistently provided less recovery error than Ridge regression
  - This is likely because the L_1 norm will drive low coefficients to zero, whereas L_2 will focus on reducing large coefficients to smaller values
    - Doesn’t match the sparsity constraint in the DCT domain!

References

- Chen, Guang-Hong, Jie Tang, and Shuai Leng. "Prior image constrained compressed sensing (PICCS): a method to accurately reconstruct dynamic CT images from highly undersampled projection data sets." *Medical physics* 35.2 (2008): 660-663.
- Davenport, Mark A., et al. "Introduction to compressed sensing." (2012): 1-64.
- Eldar, Yonina C., and Gitta Kutyniok, eds. *Compressed sensing: theory and applications*. Cambridge university press, 2012.
- Justusson, B. I. "Median filtering: Statistical properties." *Two-Dimensional Digital Signal Processing II*. Springer, Berlin, Heidelberg, 1981. 161-196.
- Marquardt, Donald W., and Ronald D. Snee. "Ridge regression in practice." *The American Statistician* 29.1 (1975): 3-20.
- Mackenzie, Dana , Compressed Sensing Makes Every Pixel Count. (Mackenzie, Dana (2009), "Compressed sensing makes every pixel count", What's Happening in the Math. Sciences, AMS, 114-127)
- Ranstam, J., and J. A. Cook. "LASSO regression." *Journal of British Surgery* 105.10 (2018): 1348-1348.
- Python Packages:
  - Numpy
  - Pandas
  - Matplotlib
  - Scipy
  - Sklearn