Introduction to Machine Learning

Mini-Project 1

Compressed Sensing Image Recovery

Libo Zhang (lz200)

Table of Contents

1. Introduction (Problem Description)

(1) What is the nature of this project?

This project applies the theory of regularized sparse regression to compressed sensing image recovery (as illustrated by the mini-project name). Pixels of an image could be either manually sampled or accidentally corrupted, and the image quality will be explicitly degraded. Although an image might have some missing pixels, with the help of regression, we can still try to reconstruct a full image with relatively good quality, given even a small number of sampled pixels, and this procedure is known as compressed sensing [R1].

Then, as for the “sparse” term, it is more about the constraints we need to consider when implementing regression. This is because instead of displaying smoothness, natural images tend to exhibit sparsity. Therefore, when we try to implement regression with the purpose of estimating unknown pixels of a sampled or corrupted image, we must take sparsity into consideration, which eventually leads to the L1-norm regularization of (LASSO) regression models [R2].

1. Introduction (Problem Description)

(2) What are the goals of this project?

There are several goals that should be achieved at completion. These goals are shown below.

Firstly, explain, analyze, and demonstrate what, why, and how compressed sensing image recovery is a reasonable and appropriate application of regularized sparse regression.

Secondly, generate the transformation matrix for a 2-Dimensional (denoted as 2-D) Discrete Cosine Transform (denoted as DCT) and apply 2-D DCT to model images in the frequency domain vertically and horizontally.

Thirdly, simulate a compressed sensing scenario by randomly selecting pixels from each block of an image.

Fourthly, reconstruct a corrupted image by implementing random subset cross-validation and LASSO regression.

Fifthly, qualitatively interpret and quantitatively evaluate the recovered image quality.

Sixthly, try to develop, experiment, and demonstrate an understanding of the correlation between important parameters including but not limited to the block size, the sample size, the application of median filter, the range of regularization strength, and the cross-validation with random subsets.

1. Introduction (Problem Description)

(3) What is the brief summary of key results?

Several key results are summarized below.

Firstly, the image recovery problem can be formulated as an under-determined linear system mathematically.

Secondly, the image recovery problem can be solved by applying LASSO regression and cross-validation with random subsets. The quality of recovered image can be very good given appropriate parameter settings.

Thirdly, it is very important to consider a large range of L1-norm regularization strength (denoted as λ) because when implementing cross-validation with random subsets, the optimal λ could be very different for each block.

Fourthly, given a certain block size (denoted as K) and a certain original image, with the sample size (denoted as S) increasing, the quality of recovered image becomes better both qualitatively (much clearer and more interpretable) and quantitatively (less mean square error between the original image and the recovered image).

Fifthly, for a small sample size, applying the median filter can help increase the recovered image quality. For a large sample size, however, applying the median filter will decrease the recovered image quality.

2. Mathematical Formulation

(1) How to formulate the image recovery problem as an under-determined linear system?

A 2-D signal, such as a grayscale image $C \in \mathbb{R}^{W \times H}$ (a grayscale image does not have a third dimension for RGB) with width W and height H, can be represented as a weighted sum of D 2-dimensional basis functions [C1]

$$C = \gamma_1 T_1 + \gamma_2 T_2 + \cdots + \gamma_D T_D$$

Here, $\gamma_d \in \mathbb{R}$ is the weight for the corresponding basis function $T_d \in \mathbb{R}^{W \times H}$. If we apply rasterization (or flattening) to reshape these matrices into column vectors, we can have [C2]

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{W \times H} \end{bmatrix} = [T_1 T_2 T_3 \ldots T_D] \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \vdots \\ \gamma_D \end{bmatrix}$$

Here, each element of C is a pixel of an image, each column of T is a basis function, and each element of γ is the weight for the corresponding basis function.

2. Mathematical Formulation

(1) How to formulate the image recovery problem as an under-determined linear system?

Then, we can abstract the previous matrix multiplication as the vector-matrix notation below [C3]

$$C = T\gamma$$

$$\{[C \in \mathbb{R}^{(WH)\times 1}], [T \in \mathbb{R}^{(WH)\times D}], \text{and } [\gamma \in \mathbb{R}^{D\times 1}]\}$$

For the 2-D Discrete Cosine Transform (DCT) [R3], the transformation matrix T can be calculated given a certain image size or block size (K = P = Q), and the integrated equations for $[Image\ Pixel] = [Transformation] \times [DCT\ Coefficient]$ are shown below [C4]

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

$$x, u \in \{1,2,\dots,P=K\} \text{ and } y, v \in \{1,2,\dots,Q=K\}$$

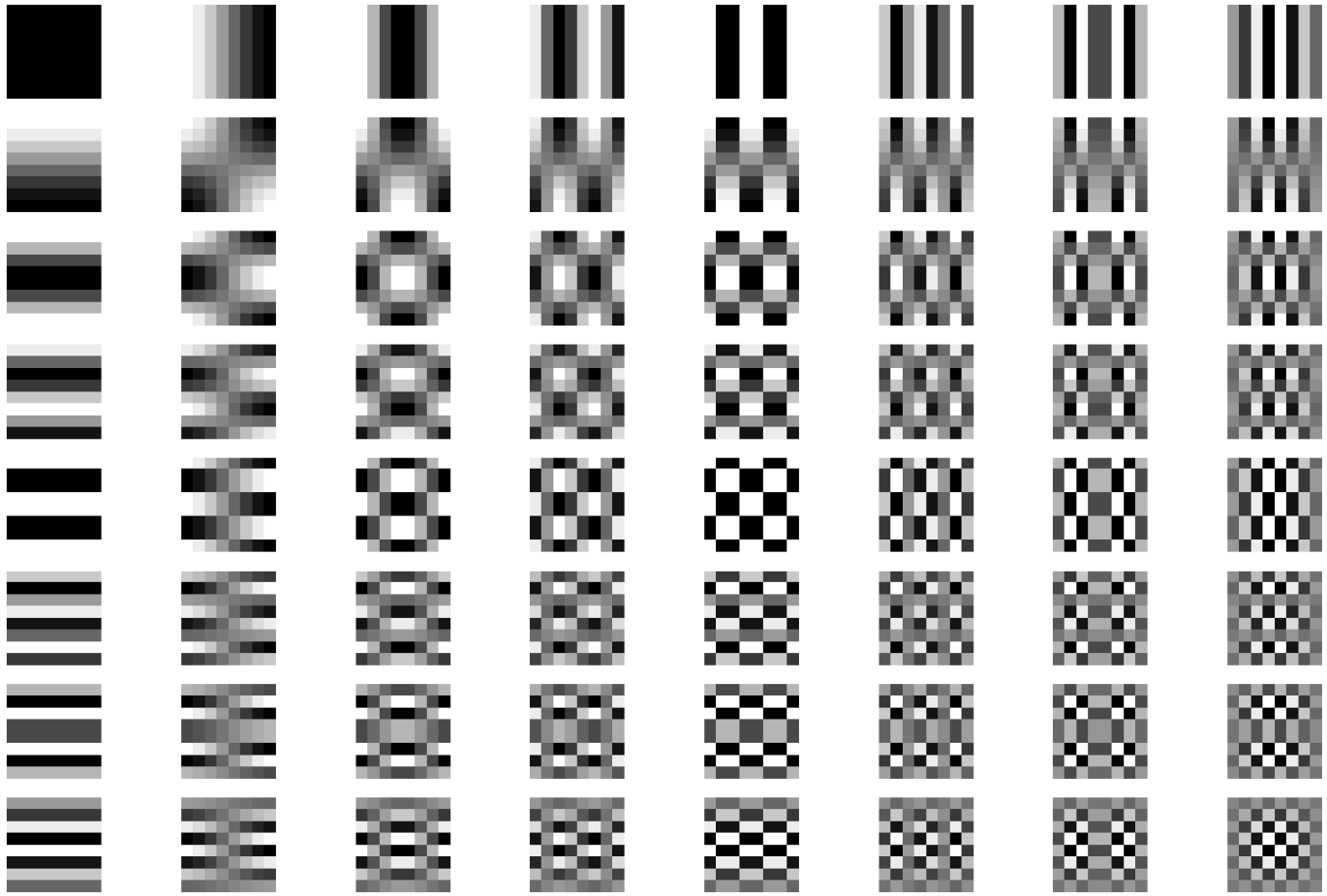
$$\alpha_u = \begin{cases} \sqrt{1/P} & (u=1) \\ \sqrt{2/P} & (2 \le u \le P) \end{cases} \text{ and } \beta_v = \begin{cases} \sqrt{1/Q} & (v=1) \\ \sqrt{2/Q} & (2 \le v \le Q) \end{cases}$$

2. Mathematical Formulation

(1) How to formulate the image recovery problem as an under-determined linear system?

To help understand the complicated equation for 2-D DCT presented in the previous slide, on the right side I plot the 8*8 = 64 basis functions for the 2-D DCT transformation matrix assuming the block size is 8 (K = 8).

I think this also helps to provide better **visualization** in the mathematical formulation section.



8*8 = 64 Basis Functions for 2-D DCT Transformation Matrix
Block Size K = 8 = P (Horizontal) = Q (Vertical)

2. Mathematical Formulation

(1) How to formulate the image recovery problem as an under-determined linear system?

If the image C is completed available without any missing pixels, then we can easily solve the corresponding DCT coefficients by [C5]

$$\gamma = T^{-1}C$$

However, what if the image C is manually sampled or accidentally corrupted, and we can only have access to samples of C (denoted as sample vector B). Under such case, how to accurately calculate or estimate the DCT coefficients γ ?

For a rasterized image vector $C \in \mathbb{R}^{(W*H)\times 1}$ with $W \times H$ pixels, if we manually sample S pixels to form a corrupted rasterized image vector $B \in \mathbb{R}^{(S)\times 1}$ [R1], and record the pixel sampling position to correspondingly form a corrupted rasterized transformation matrix $A \in \mathbb{R}^{(S)\times D}$ from the originally known rasterized transformation matrix $T \in \mathbb{R}^{(W*H)\times D}$ [R3], then we can build an under-determined linear system as [C6]

$$B = A\gamma \text{ and } \gamma \in \mathbb{R}^{D\times 1}$$

Therefore, we have successfully formulated the image recovery problem as an under-determined linear system, and all the previous context clearly explained ***why* the image recovery problem is within the class of problems of solving underdetermined linear systems**. However, currently we can not recover the sample/corrupted image because we do not have the complete image C to directly calculate the DCT coefficients γ. How do we approximate or estimate the DCT coefficients γ given only the sampled or corrupted image?

2. Mathematical Formulation

(2) How to solve the regression problem with L1-norm regularization to recover the image?

Firstly, *why* is L1-norm regression chosen as the regularization approach? To answer this question, we should start with the property of underdetermined linear system. An underdetermined linear system has either infinite number of solutions or no solution at all, so that we need to add some constraints to solve it [R4]. Although smoothness is the common constraint type, natural images have a trend of being sparse in the Discrete Cosine Transform (DCT) domain, therefore we will add sparsity constraint to the system.

In addition, since we are trying to approximate/estimate DCT coefficients based on sampled pixels, this can be done by regression, which belongs to one of the four categories of machine learning (regression, classification, clustering, and information retrieval).

Furthermore, there are two types of regularization for regression, L1-norm and L2-norm. Both types of regularization will try to make weights smaller, but only L1-norm can induce sparsity, which means shrinking many weights to 0s [R2]. If we regard the DCT coefficients as the weights we want to estimate using regression, then the sparsity induced by L1-norm regularization corresponds to the sparse constraint that we want to add to the under-determined linear system.

The context above thoroughly explains ***why* L1-norm (LASSO) regression is chosen as the regularization approach**, but how should we appropriately solve the regression problem with L1-norm regularization to recover the sampled/corrupted image?

2. Mathematical Formulation

(2) How to solve the regression problem with L1-norm regularization to recover the image?

If we add sparsity constraint to the under-determined linear system $B = A\gamma$, we will get an optimization problem as [C7]

$$\begin{cases} min_{\gamma} \|A\gamma - B\|_2^2 \\ \|\gamma\|_1 \leq \lambda \end{cases}$$

Since we are using LASSO regression with L1-norm regularization [R2], this optimization problem can be modified as [C8]

$$min_{\gamma} \|A\gamma - B\|_2^2 + \lambda \|\gamma\|_1$$

It is important to note that when we work on compressed sensing, we should first divide the original image into many small blocks with a predefined block size (K). This is because a large image tends not to have sparse DCT coefficients, and this property could make our sparse constraint meaningless and degrade the quality of recovered image.

If we have an image with size (200*192), and we set K = 8, we will have (200/8)*(192/8) = 25*24 = 600 small blocks. Each block has 64 pixels in total and a size of 8*8. Then for each small block, we randomly sample S pixels, apply LASSO regression to find DCT coefficients $\hat{\gamma}$, and estimate missing pixels in each block based on [C9]

$$\hat{C} = T\hat{\gamma}$$

Here we should only recover missing (unknown) pixels and leave all S sampled pixels (already known) unchanged.

2. Mathematical Formulation

(2) How to solve the regression problem with L1-norm regularization to recover the image?

It is also very important to mention that, since we are dividing an original image into many small blocks, different blocks could contain dramatically various properties in terms of sparsity. Therefore, we should predefine a very large range of L1-norm regularization strength λ and consider many potential values. In my experiments, for example, I consider 25 possible λ values ranging from 10^{-6} to 10^{+6}.

Then, how to choose a good λ for each block? To answer this question, we need to introduce cross-validation with random subsets [R5].

Step 1 – For each block, we first predefine the number of independent repetitions for such cross-validation process as $M = 20$.

Step 2 – We randomly extract $m = floor(S/6)$ pixels for testing and send the rest $(S - m)$ pixels for training.

Step 3 – We go over such training and testing process for each candidate λ, record the mean square error (MSE) for each repetition.

Step 4 – After all 20 repetitions, we compute the average of the 20 recorded MSEs for each candidate λ and find the best λ which has the least average MSE. This best λ will be regarded as the optimal L1-norm regularization strength for this specific block.

2. Mathematical Formulation

(2) How to solve the regression problem with L1-norm regularization to recover the image?

Now we have successfully addressed how to appropriately apply sparse constraint to each image block.

For each block, we use the optimal λ just found and all S sampled pixels to implement LASSO regression. After fitting, the model weights are the approximated DCT coefficients $\hat{\gamma}$ we want.

Then, we apply the previously mentioned equation $\hat{C} = T\hat{\gamma}$ [C9] to estimate the missing pixels. Two key notes are shown below.

Note 1 – We only estimate the missing pixels (unknown), we should leave all S sampled pixels (already known) unchanged.

Note 2 – When coding for LASSO, I make all data centered so that the model intercept will be 0, and no intercept needed for model prediction.

Finally, after estimating all missing pixels in all blocks, we concatenate all small blocks into a recovered large image, based on the same order (I use row-major order in this project) which we used to divide the original large image into many small blocks.

Getting the recovered large image means that we have finally solved the regression problem with L1-norm regularization to recover the image. Next, we should apply median filtering to see if we can improve the recovered image quality.

2. Mathematical Formulation

(3) How to filter the recovered image to improve image quality?

To answer this question, first I want to describe *how* the median filtering works to improve the recovered image quality, then I will explain ***why* median filtering is chosen over other approaches such as frequency-selective (low-pass or high-pass) filtering.**

Median filter is a non-linear digital filter that runs through the image pixel by pixel, replacing each pixel with the median pixel value of its neighboring pixels [R6]. Let us assume a median filter has filter size (3×3), and it is currently processing an image pixel at position (x, y), the diagram below [C10] shows how this median filter works to provide better **visualization**.

124	126	127
120	**150 (x, y)**	125
115	119	123

115, 119, 120, 123, **124**, 125, 126, 127, 150

The Median Value is 124

Sort and Replace

124	126	127
120	**124 (x, y)**	125
115	119	123

2. Mathematical Formulation

(3) How to filter the recovered image to improve image quality?

Now we know how median filtering works, then ***why* is median filtering chosen over other filtering approaches such as frequency-selective (low pass or high-pass) filtering?** I conclude several reasons below to answer this question.

First, since median filtering always replaces a pixel with the median pixel value of its neighborhood, it can efficiently remove spike noise while keeping monotonic edges unchanged. This property tends to make our recovered image look smoother.

Second, since we need to divide the image into many small blocks, some blocks may have very good recovery quality, but some other blocks may not. This tends to make some area of the recovered image look like having some white dots known as the “salt and pepper” noise. Since median filtering can suppress noise and bring smoothness, such “salt and pepper” noise can be efficiently alleviated [R6].

Third, low-pass filter preserves low frequency and blocks high frequency, while high-pass filter preserves high frequency and blocks low frequency. Therefore, low-pass filter can also introduce smoothness to images, while high-pass filter enhances or sharpens images [R7].

To summarize, We do not want to further sharpen the recovered image which will make the reconstructed quality even worse, so high-pass filter is excluded. Low-pass filtering has similar smoothness effects as median filtering. However, for this entire mini project, we are not asked to work in the frequency domain at all. All we need to focus on are non-negative integer pixels, which also belong to the real number domain. Therefore, I think median filtering is the most appropriate choice to filter the recovered image to improve image quality.

3. Experimental (Simulation) Results

Before presenting all experimental results, I want to briefly describe my experimental (simulation) conditions.

For each small block, to implement cross-validation with random subsets, I consider 25 candidate λ values ranging from 10^{-6} to 10^{+6}, with 2 or 3 values every decade in the log space form. I also set the number of repetitions $M = 20$, split $m = floor(S/6)$ pixels for testing and use the rest $(S - m)$ sampled pixels for training.

For the small test image "fishing boat", I set the block size $K \times K = 8 \times 8$.

I consider 6 different sample sizes (1 more than required), and they are S = 10, 20, 30, 40, 50, **60**.

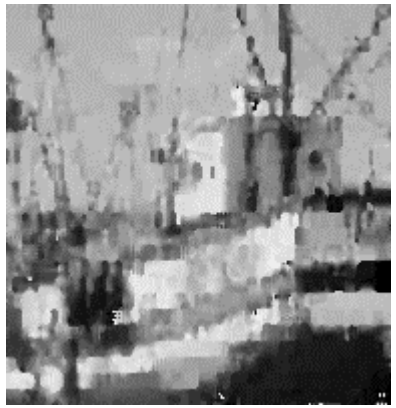
For the large test image "nature", I set the block size $K \times K = 16 \times 16$.

I consider 6 different sample sizes (1 more than required), and they are S = 10, 30, 50, 100, 150, **200**.

For each test image and every sample size, I save both recovered images with and without median filtering, I calculate the mean square error (MSE) between the recovered image and the original image to quantitatively evaluate recovery quality.

Next, I will present all experimental (simulation) results.

Recovered Small Image
Before Median Filtering
S = 10
Mean Square Error (MSE) = 1154.9



Recovered Small Image
After Median Filtering
S = 10
MSE = 727.0

Recovered Small Image
Before Median Filtering
S = 20
MSE = 430.6

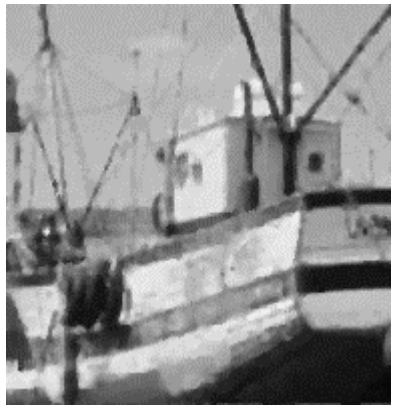
Recovered Small Image
After Median Filtering
S = 20
MSE = 349.5

Original Small Test Image
"fishing boat"
(200 * 192)

3. Experimental (Simulation) Results

Key take-away points – For the small test image, we can see that if the sample size (S) is too small, although median filtering can help reduce MSE and introduce smoothness, both recovered images before and after median filtering have generally bad quality.

Recovered Small Image
Before Median Filtering
S = 30
Mean Square Error (MSE) = 218.2



Recovered Small Image
After Median Filtering
S = 30
MSE = 222.5

Recovered Small Image
Before Median Filtering
S = 40
MSE = 95.5

Recovered Small Image
After Median Filtering
S = 40
MSE = 156.4

Original Small Test Image
"fishing boat"
(200 * 192)

3. Experimental (Simulation) Results

Key take-away points – If we increase the sample size (S), we can see that the recovered images have much better quality and less MSE. It should also be noticed that for $S \geq 30$, the recovered image without median filtering has less MSE than the recovered image with median filtering, which means that median filtering does not always improve image recovery quality.

Recovered Small Image
Before Median Filtering
S = 50
Mean Square Error (MSE) = 43.8

Recovered Small Image
After Median Filtering
S = 50
MSE = 128.0

Recovered Small Image
Before Median Filtering
S = 60
MSE = 7.2

Recovered Small Image
After Median Filtering
S = 60
MSE = 108.3

Original Small Test Image
"fishing boat"
(200 * 192)

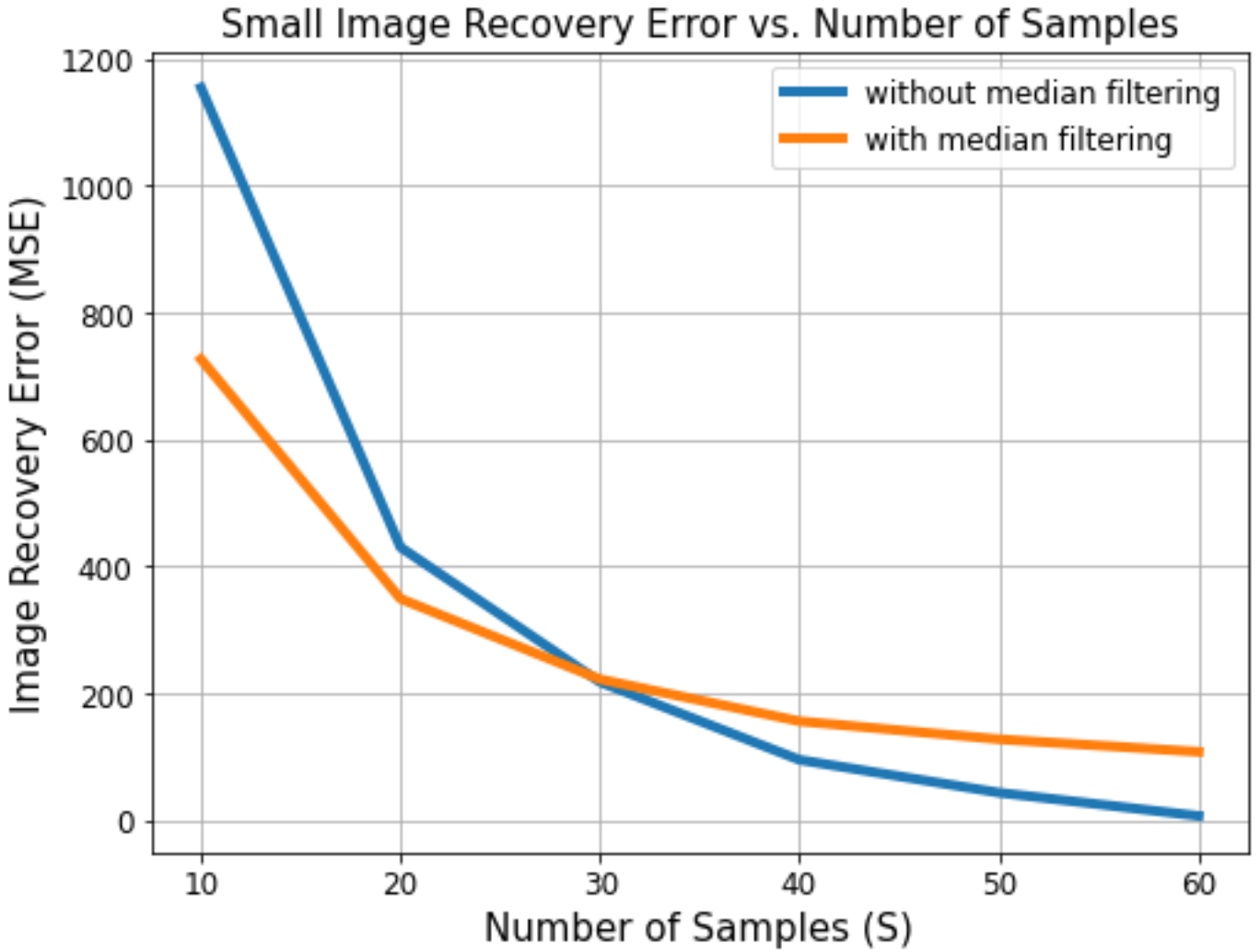
3. Experimental (Simulation) Results

Key take-away points – If we further increase the sample size (S) to a very large fraction of the total number of pixels (8*8 = 64) for each block, we can see that the recovered images have even better quality. Here, we can notice that the recovered images without median filtering have "faster" quality improvement and better image recovery quality, while the recovered images with median filtering has "slower" quality improvement.

3. Experimental (Simulation) Results

For the small test image “fishing boat”, the plot comparing recovery error vs. number of samples with median filtering and without median filtering is provided below.

Key take-away points – Based on the right figure, we can see that if we increase the sample size (S) or increase the fraction of sampled pixels for each block, we will have less image recovery error (MSE) and better image recovery quality.
In addition, we can notice that median filtering will slow down the “speed” of improvement of image recovery quality. We can also see that median filtering does not help improve image recovery quality when S is relatively large, so that we should not always use median filtering to recover sampled/corrupted images.



3. Experimental (Simulation) Results

For the large test image “nature”, since the original image size (512 * 640) is a bit large, I need adjust how I present the recovered images (with and without median filtering) compared with the original image. I will only show the original large image with its original size in this slide to save some space.

For the following slides, I will present 2 recovered images (with and without median filtering) per slide, with the purpose of utilizing all available space while still providing high quality visualization.

Original Large Test Image
“nature”
(512 * 640)

Recovered Large Image **Before Median Filtering**

S = 10

Mean Square Error (MSE) = 1874.9

Recovered Large Image **After Median Filtering**

S = 10

MSE = 1285.9

Key take-away points – Clearly, if the sample size (S) is small, both recovered images with and without median filtering will have very bad recovery quality and very high recovery error (MSE), but currently median filtering can introduce smoothness, reduce MSE, and slightly improve image recovery quality.

Recovered Large Image **Before Median Filtering**

S = 30

Mean Square Error (MSE) = 1125.1

Recovered Large Image **After Median Filtering**

S = 30

MSE = 659.5

Key take-away points – If we increase the sample size (S), the recovered image will have less MSE and better quality. The median filter can help improve image recovery quality and reduce MSE by removing “salt and pepper” white dots noise and introducing smoothness.

Recovered Large Image **Before Median Filtering**

S = 50

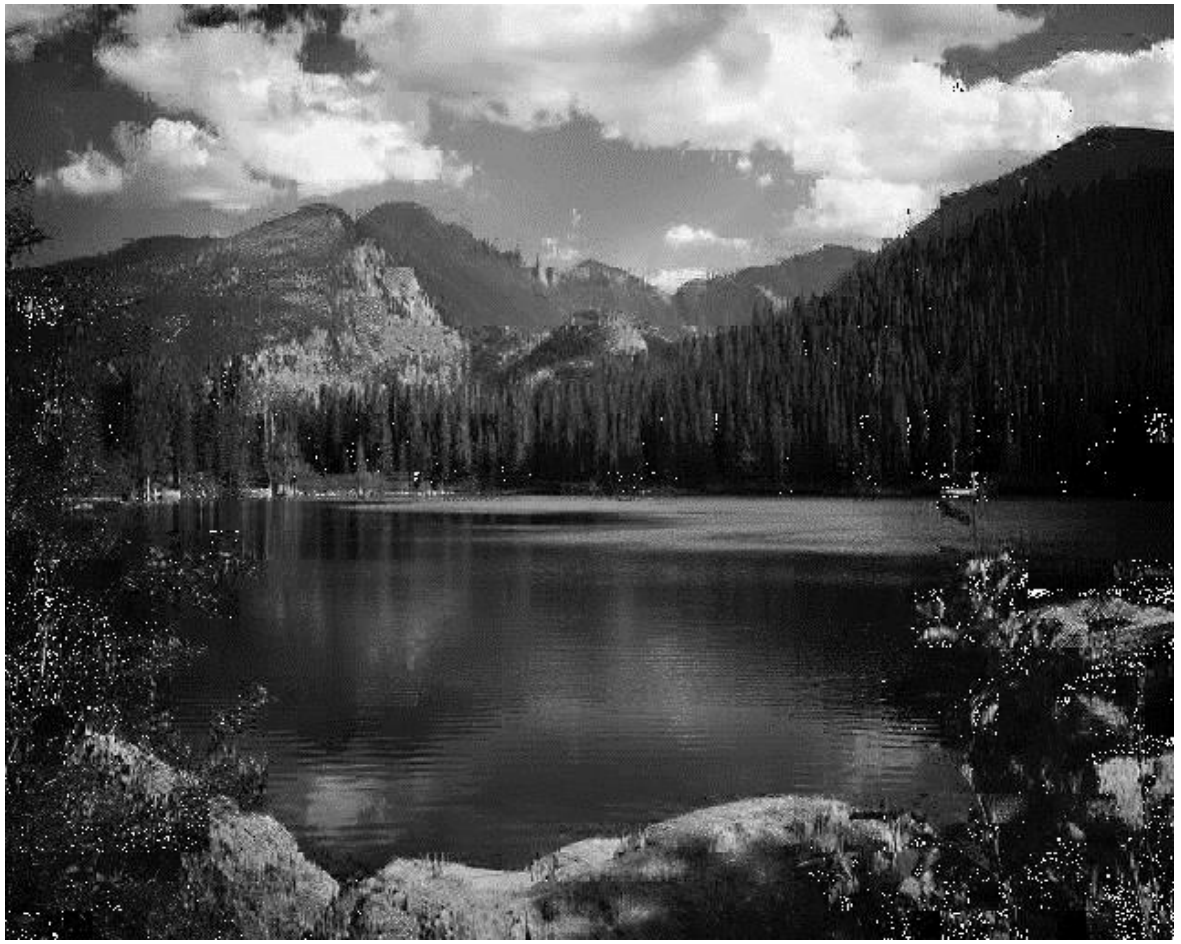
Mean Square Error (MSE) = 925.6

Recovered Large Image **After Median Filtering**

S = 50

MSE = 515.5

Key take-away points – If we continue to increase the sample size (S), the recovered image will have better quality! The median filter can improve image recovery quality and reduce MSE by removing “salt and pepper” white dots noise and introducing smoothness.



Recovered Large Image **Before Median Filtering**

S = 100

Mean Square Error (MSE) = 552.4

Recovered Large Image **After Median Filtering**

S = 100

MSE = 332.5

Key take-away points – We can notice that increasing the sample size (S) will improve image recovery quality, and it can also suppress the "salt and pepper" white dots noise without median filtering! However, median filtering can still improve image recovery quality currently.

Recovered Large Image **Before Median Filtering**

S = 150

Mean Square Error (MSE) = 323.7

Recovered Large Image **After Median Filtering**

S = 150

MSE = 261.7

Key take-away points – As we continue to increase the sample size (S), the recovered images have better quality, less MSE, and less "salt and pepper" white dots noise. For now, we might notice that the recovered images with median filtering tend not to have clearly better quality than the recovered images without median filtering.

Recovered Large Image **Before Median Filtering**

S = 200

Mean Square Error (MSE) = 132.4

Recovered Large Image **After Median Filtering**

S = 200

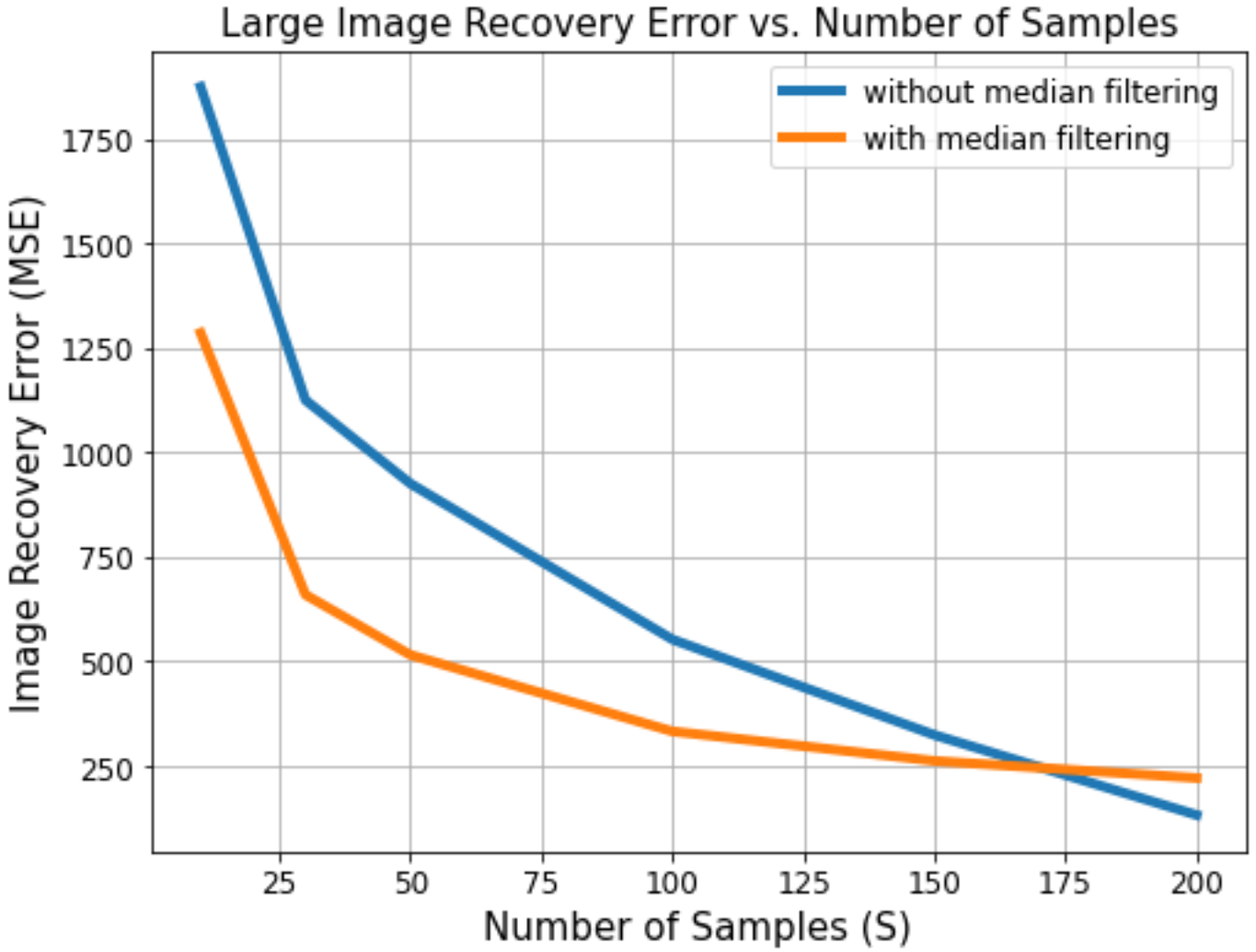
MSE = 220.9

Key take-away points – When the sample size (S) is very large, or the fraction of sampled pixels for each block is very large, we can see that the recovered images have extremely good quality. In addition, the recovered image without median filtering has better quality and less MSE than the recovered image with median filtering.

3. Experimental (Simulation) Results

For the large test image “nature”, the plot comparing recovery error vs. number of samples with median filtering and without median filtering is provided below.

Key take-away points – Based on the right plot, we can see that if we increase the sample size (S) or increase the fraction of sampled pixels for each block, we will have less image recovery error (MSE) and better image recovery quality. This trend is very similar to what we have observed from the small test image. The absolute MSE value for the large test image tends to be higher, which is understandable because the large image has more missing pixels to estimate. In addition, we can notice that the median filter will slow down the “speed” of improvement of image recovery quality. We can also see that median filtering does not help improve image recovery quality if we select a very large sample size (S). Therefore, we should not always use median filtering to recover sampled/corrupted images.



3. Experimental (Simulation) Results

To earn a 9.5 on this section, apart from the previously presented S = 60 for the small test image and S = 200 for the large test image, I also want to provide and interpret additional simulation results during my own explorations in the next few slides.

In specific, for the small test image “fishing boat”, I set the block size to 4×4 instead of 8×8. I set the sample size (S) as 10.

I run the image recovery simulation again and save the recovered small images with and without median filtering.

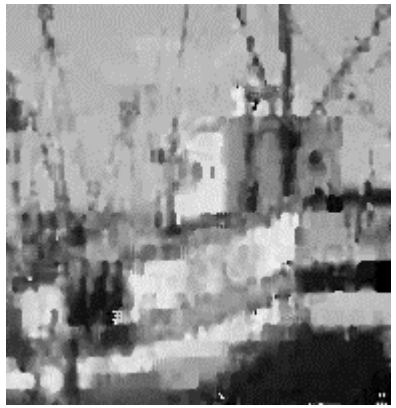
For the large test image “nature”, I also set the block size to 4×4 instead of 16×16. I also set the sample size (S) as 10.

I run the image recovery simulation again and save the recovered large images with and without median filtering.

In the previous experimental results presenting, both the small image and the large image with a sample size S = 10 have very bad image recovery quality, with or without median filtering. With the current block size 4×4, however, 10 sampled pixels are a large fraction of the total 4*4 = 16 pixels for each block. Will the image recovery quality be better?

In fact, presenting and interpreting these simulation results can also help me prepare for more comprehensive questions in the following Discussion/Conclusions section.

Recovered Small Image
Before Median Filtering
K = 8, S = 10
Mean Square Error (MSE) = 1154.9



Recovered Small Image
After Median Filtering
K = 8, S = 10
MSE = 727.0

Recovered Small Image
Before Median Filtering
K = 4, S = 10
MSE = 19.2

Recovered Small Image
After Median Filtering
K = 4, S = 10
MSE = 38.8

Original Small Test Image
"fishing boat"
(200 * 192)

3. Experimental (Simulation) Results

Key take-away points – We can see that for the same sample size (S), if the block size (K) is smaller (the fraction of sampled pixels is larger for each K*K block), the recovered image will have much better quality. However, since we are dividing the original image into more smaller blocks, the entire image recovery simulation will be more computationally expensive.

Recovered Large Image **Before Median Filtering**

K = 16, S = 10

Mean Square Error (MSE) = 1874.9

Recovered Large Image **Before Median Filtering**

K = 4, S = 10

MSE = 18.1

Key take-away points – We can see that for a certain sample size (S), if we decrease the block size (K) which is equivalent to increasing the fraction of sample pixels for each K*K block, the recovered image will have much better quality.

Recovered Large Image **After Median Filtering**

K = 16, S = 10

Mean Square Error (MSE) = 1285.9

Recovered Large Image **After Median Filtering**

K = 4, S = 10

MSE = 37.0

Key take-away points – For the smaller block size K = 4, compared with the previous slide, we can also see that although median filtering can help remove “salt and pepper” white dots noise and introduce smoothness, such smoothing operation will decrease the recovered image quality a little bit.

4. Discussion / Conclusions

Discussion Question (1) – Why we see trends we see in the experimental results?

For the basic experimental requirements, we can see that for both the small test image “fishing boat” and the large test image “nature”, if we select a certain block size (K), and increase the sample size (S), then the recovered images will have better quality. I think there are 2 major reasons to explain why this makes sense.

Firstly, if we have more sampled pixels (already known to us) for each K*K block, then within the same block, we will have fewer missing pixels (unknown to us) to estimate. Such sampling and missing trade-off itself can help reduce image recovery error.

Secondly, when we are working on regression models, it is always helpful to have more observations (data points). Therefore, when we are implementing LASSO regression for each block, if we have more sampled pixels for model training, it can be expected that the LASSO regression model after cross-validation with random subsets will have better performance in predicting/estimating the missing pixels. This also explains why the image recovery error (MSE) will decrease if we just increase the sample size (S).

4. Discussion / Conclusions

Discussion Question (2) – Should we always median filter the recovered image?

No, we should not always use median filtering to recover the sampled/corrupted image. As I explained in the key take-away points in the section of experimental results, for both the small test image “fishing boat” and the large test image “nature”, when the sample size (S) is small, median filtering can clearly improve image recovery quality. When the sample size is large, applying the median filter will decrease the recovery quality a bit, compared with the recovered image without median filtering. In addition, median filtering will slow down the speed of improvement of the image recovery quality.

To explain the above trends for median filtering, I think we should focus on the mathematical foundation of a median filter, which is to replace each pixel value with the median value of its neighboring pixels. Such operation will introduce smoothness to suppress the spike noise and the “salt and pepper” noise we have seen in experimental results, which explains why median filtering can improve image recovery quality when the sample size is small. However, when sample size is large, we will be able accurately estimate missing pixels with a good regression model for each block. The estimated pixels have already been accurate enough, if we still apply median filtering, then the smoothness introduce by the median filter will have negative effects by “distorting” the accurately estimated pixels. I think this explains why we can see the recovery error (MSE) would increase after median filtering in the section of experimental results.

In summary, we should not always median filter the recovered image. If the sample size is small (the fraction of sampled pixels for each block is small), we should apply median filtering, but if the sample size is relatively large (the fraction of sampled pixels for each block is large), then we should not apply median filtering.

4. Discussion / Conclusions

Discussion Question (3) – How does the proportion of missing pixels influence the choice of block size?

We have previously observed in experimental results and discussed in Discussion Question (1) that for a certain block size (K), increasing the number of sampled pixels (decreasing the number of missing pixels) will result in better image recovery quality.

When it comes to the choice of block size (K), my explanation will be based on the additional simulation results I provide in experimental results, in which I changed the block size of both small and large test images to K = 4 and set the sample size S = 10. We have observed significant recovery quality improvement compared with the original (K = 8, S = 10) for the small image and the original (K = 16, S = 10) for the large image.

Firstly, this can be expected because now we have a larger fraction of sampled pixels for each block (the number of sampled pixels is unchanged while the total number of pixels with each block decreases).

Secondly, with a smaller block size, we will divide an image into much more smaller blocks. If we keep the sample size unchanged, such as S = 10, we can not save computational resources, within each block, when using the sampled pixels for cross-validation with random subsets. Therefore, a smaller block size will make the image recovery simulation more computationally expensive.

In summary, I think the trade-off between the recovered image quality and the required computational cost is also a very important topic when we try to select an appropriate value for the block size.

4. Discussion / Conclusions

Conclusion Topic (1) – Factors that may impact quality of the recovered image?

According to all previous experimental results and explanations, we can conclude that, given one certain grayscale image, the factors that may impact quality of the recovered image in this mini-project include but are not limited to the sample size (S), the block size (K), the range and total number of candidate L1-norm regularization strength values λ during cross-validation with random subsets, and whether to apply median filtering or not.

Conclusion Topic (2) – Limits or problems with your approach?

Since I have eventually achieved very good image recovery quality for both the small and large test images, I do not think there are any very big problems with my approach to recover the sampled/corrupted images. However, I do think there are 2 very explicit limits when I work on the compressed sensing image recovery project.

First, when I work on my Python codes, I use many “for” loops to divide the image into small blocks, to implement cross-validation with random subsets, and to concatenate small blocks back into the complete image. We all know that Python is not optimized with many “for” loops written in codes, so that the long runtime of my experimental simulation is partially related to this limit.

Second, I do not think I have conducted comprehensively ample simulation to thoroughly research on possible factors that may impact quality of recovered image. For example, what about another median filter size instead of 3*3, or is there a good way to quantitatively measure the trade-off between image recovery quality and computational cost? These questions remain unanswered at the end of this mini-project.

4. Discussion / Conclusions

Conclusion Topic (3) – Possible improvements that can be made?

I think further effort can be made to improve the 2 limits mentioned in Conclusion Topic (2).

First, I should refine my code and avoid the abuse of multiple “for” loops. To do that, I should get myself more familiar with vector and matrix implementation in Python, which have been efficiently optimized and accelerated. For example, if I can build a two-dimensional array instead of one, the additional one dimension may help me save one “for” loop. With less “for” loops, I might be able to accelerate my simulation a bit.

Second, maybe I could conduct more experiments/simulations and research more comprehensively or thoroughly on factors that impact the compressed sensing image recovery quality. Future work research questions could include but might not be limited to

1. Instead of 3*3, what about other median filter size, such as 2*2, 4*4, and 5*5?
2. How to select good metrics to quantitatively measure the computational cost for each image recovery task?
3. In this project we only consider 2-D grayscale images, what about 3-D color images with RGB channels?

4. Discussion / Conclusions

Conclusion Topic (4) – Anything unique you have done to improve/validate your program's accuracy/efficiency?

As presented in additional experimental results and explained in Discussion Question (3), I think the only unique thing I have done to improve image recovery quality is to set a very small block size, which is $K \times K = 4 \times 4$ for both the small and large test images. It turns out that I can achieve very good image recovery quality, even with the small sample size $S = 10$.

However, a much smaller block size means the computational cost is much more expensive, which is the trade-off price I must pay to get the high-quality recovered image.

It is true that I have conducted, presented, and analyzed more experiments than the standard requirement, but I do not think they belong to the "unique" category, because my own machine learning pipeline for this project strictly follows the standard instructions and requirements in the mini-project 1 handout.

5. References / Citations

References [R#] from books or journal articles

[R1] Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on information theory*, *52*(4), 1289-1306.

[R2] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267-288.

[R3] Ahmed, N., Natarajan, T., & Rao, K. R. (1974). Discrete cosine transform. IEEE transactions on Computers, 100(1), 90-93.

[R4] Foucart, S., & Lai, M. J. (2009). Sparsest solutions of underdetermined linear systems via ℓq-minimization for $0< q\leqslant 1$. Applied and Computational Harmonic Analysis, 26(3), 395-407.

[R5] Refaeilzadeh, P., Tang, L., & Liu, H. (2009). Cross-validation. Encyclopedia of database systems, 5, 532-538.

[R6] Narendra, P. M. (1981). A separable median filter for image noise smoothing. IEEE Transactions on Pattern Analysis and Machine Intelligence, (1), 20-29.

[R7] Makandar, A., & Halalli, B. (2015). Image enhancement techniques using highpass and lowpass filters. International Journal of Computer Applications, 109(14), 12-15.

5. References / Citations

Citations [C#] from ECE 580 course materials (my own modified or reproduced work included)

[C1] This equation is reproduced from the ECE 580 course mini-project 1 description material "MP1.1 – Compressed Sensing Image Recovery" (denoted as MP1.1 below), Page 6.

[C2] This equation is reproduced from MP1.1, Page 6.

[C3] This equation is reproduced from MP1.1, Page 6.

[C4] This equation is reproduced from MP1.1, Page 9.

[C5] This equation is reproduced from MP1.1, Page 10.

[C6] This equation is reproduced from MP1.1, Page 11.

[C7] This equation is reproduced from MP1.1, Page 12.

[C8] This equation is reproduced from MP1.1, Page 12.

[C9] This equation is reproduced from MP1.1, Page 11.

[C10] This median filtering diagram example is reproduced from MP1.1, Page 16.

5. References / Citations

Provide a citation for every toolbox or package I leverage when coding

I use Python to work on this project. All leveraged packages and the correspondingly leveraged purposes are described below.

(1) I leverage “math” package to introduce special values like π, and to implement mathematical operations like cosine.

(2) I leverage “matplotlib” package to read and plot original images, and to plot basis functions of transformation matrix.

(3) I leverage “imageio” package to read and save recovered grayscale images.

(4) I leverage “warnings” package to ignore the printed warnings during regression, which helps save computer memory.

(5) I leverage “scikit-learn” package to implement cross-validation with random subsets, to implement LASSO regression, and to calculate the mean square error (MSE).

(6) I leverage “SciPy” package to apply median filtering.

(7) I leverage “NumPy” package to process multi-dimensional array/matrix such as a grayscale image.

6. Collaboration Descriptions

I only shared and debated some **general/conceptual ideas** such as transformation matrix and cross-validation with my Peer Feedback team members during 2 Peer Feedback Sessions held in class.

Apart from the 2 Peer Feedback Sessions held in class, **I did not collaborate with anyone.** This statement means that

(1) I did not share code with anyone while working on this project.

(2) I did not compare results with anyone while working on this project.

(3) I did not help anyone overcome an obstacle while working on this project.

(4) No one helped me overcome an obstacle while working on this project.