Image Recovery by Compressed Sensing

Suqian Wang

sw443@duke.edu

1 Abstract

Compressed sensing provides a framework for the recovery of damaged or corrupted images. It assists in reducing redundant information and conserving resources. This project utilizes compressed sensing to reconstruct an image from a sample of the original image that varies in sizes, and measure the recovery quality. Discrete Cosine Transform, regularized regression, and cross-validation are the fundamental concepts that are applied to do compressed sensing. The results of the model details that the sample rate, image size, and image flexibility are factors that impacts recovery quality. This project demonstrates that an image can be recovered to varying degrees from different sized sample sets of an image using compressed sensing.

2 Overview

According to "Compressive Sensing Recovery Algorithms and Applications - A Survey," compressed sensing is defined as the collection and reconstruction of a complete signal or image through retrieving sparse data about that signal or image [1]. Compress sensing was a response to a demand for acquiring accurate information about a data set without loading the entire data set beforehand. The ability to reproduce an image using just sparse samples from the corrupt image is beneficial when working with large images.

In this project, the image recovery problem is converted to an under-determined linear system by discrete cosine transformation (DCT). Regularized regression is used to establish the sparse solution of that system. The sparse solution is the DCT coefficients we need to recover missing pixels. The regularized regression in this project utilizes the LASSO model from scikit learn. To select the ideal damping factor , cross-validation using random subsets is implemented. The missing pixels were then recovered by inverse DCT transformation. The median filter was applied to improve the noise of the image.

3 Mathematical Formulation

A graphical image can be depicted as a three-dimensional signal. The X and Y-axis of the image represent the two dimensions of the screen and each (x,y) pairing corresponds as a coordinate location of each image pixel. The Z-axis is the amplitude of the signal and it represents the value of the pixel at location (x, y). Therefore, an image can be represented by a two-dimensional array (matrix) where each cell contains the value of the pixel at that location g(x, y).

3.1 under-determined linear system representation

The Discrete Cosine Transform (DCT) was used to convert a signal in the spatial domain into its frequency domain. For an image that is transformed using DCT, that image's information can be manipulated for compression because it is in a quantitative form. The formula of DCT of each image pixel is denoted as follows:

$$g(x,y) = \sum_{u=0}^{P-1}\sum_{v=0}^{Q-1} \alpha_u \cdot \beta_v \cdot \cos\frac{\pi(2x+1)u}{2\cdot P}\cdot \cos\frac{\pi(2y+1)v}{2\cdot Q}\cdot G(u,v) \tag{1}$$

where $x, u \in \{0, 1, 2, ..., P-1\}$
$y, v \in \{0, 1, 2, ..., Q-1\}$

$$\alpha_u = \begin{cases} \sqrt{1/P} & u = 0 \\ \sqrt{2/P} & 1 \le u \le P-1 \end{cases}$$

$$\beta_u = \begin{cases} \sqrt{1/Q} & v = 0 \\ \sqrt{2/Q} & 1 \le v \le Q-1 \end{cases}$$

In equation (1), g(x, y) denotes the image pixel, 'P' is the number of pixels in the horizontal direction and 'Q' is the number of pixels in the vertical direction. The variable 'u' is the spatial frequency in the horizontal direction and 'v' is the spatial frequency in the vertical direction, and G(u, v) is the DCT coefficient.

The DCT transformation can be represented as a system of linear equations and is illustrated in Figure 1.

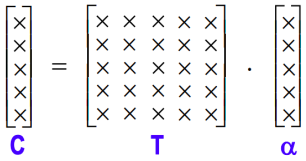


Figure 1: DCT transformation matrix representation

In Figure 1, vector C is the flattened image matrix that contains the value of each image pixel, T is the DCT transformation matrix, and α is the DCT coefficient. Each row of T corresponds to each image pixel. Each column of T corresponds to a basis function that is defined by a pair of spatial frequencies (u, v).

Image recovery by compress sensing is to recover a full image from a compressed/damaged image. In this project, a sampled image is used to simulate a real-life compressed/-damaged image. The sampled image is generated by the random selection of a few sampled pixels of the full image and setting the remaining pixels to 0. The main idea of compress sensing is to calculate the DCT coefficient α, from the sampled pixels of the spatial domain and their corresponding rows in the DCT transformation matrix. The image recovery problem would be represented as solving an under-determined linear system that is represented in Figure 2.

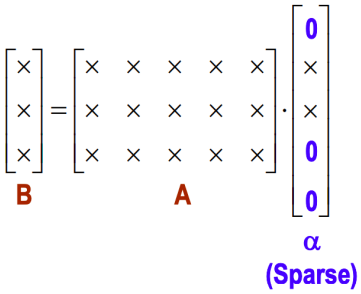


Figure 2: image recovery represented as an under-determined linear system

In Figure 2, B is the sampled image matrix (flattened) with S pixels, matrix A retains the corresponding rows from the DCT transformation matrix T, and α is the DCT coefficient.

3.2 regression with L1-norm regularization

Naturally, the DCT coefficients of a large image are not sparse, but small blocks of the large image tend to have sparse DCT coefficients. Considering the small blocks as sparse, it means there are only a small number of non-zero coefficients. Therefore, this project would start by breaking a large image into small blocks. Additionally, the calculation of the DCT transformation matrix was performed and regression with L1-norm regularization was done on those small blocks of the image.

The sparse solution for the under-determined linear system was able to be found by doing regression with L1-norm regularization, which is equivalent to LASSO (least absolute shrinkage and selection operator). The objective function is represented in equation (2).

$$\min_{\alpha} \|A\alpha - B\|_2^2, s.t. \|\alpha\|_2 \leq \lambda \tag{2}$$

In this project, the python package scikit-learn's `sklearn.linear_model.lasso` was used to fit the data (A, B) to calculate the model parameters (DCT coefficient α). To find the best value of λ, 26 different values were randomly chosen and tested within the range of $[1e-6, 1e6]$. For each λ that was selected, cross-validation was performed for 20 iterations and the average mean square error for data fitting was documented. Using the average mean squared errors previously documented, the λ that generates the smallest error was used for performing LASSO regression to obtain the model parameters α. After obtaining the parameter α, all the necessary data was calculated to perform the next step of the image recovery process; the recovery of the full image.

3.2.1 procedure for image recovery

Image prepossessing:

1. Break a large image into blocks.
2. Calculate the transformation matrix T for each image block.
3. Create the dataset for regression: randomly select S pixels from each blocks and the corresponding rows in their transformation matrix

Regression with L1-norm regularization for each image block:

1. Select a λ within range $[1e-6, 1e6]$.
   (a) Cross validation for λ (20 iterations).
      i. Randomly select $m = \lfloor S/6 \rfloor$ samples for testing, the remaining $S-m$ samples for training.
      ii. Perform LASSO regression to fit the data.
      iii. Calculate the mean square error.
      iv. Repeat.
   (b) Calculate the average mean square error for each λ
2. Repeat for all λ.
3. Select the λ with the smallest average mean square error which is the best λ.

Recovering missing pixels for each image block:

1. Perform LASSO regression on all S sampled data with the best λ found, and calculate the model parameter α.
2. Do inverse dct transformation with the dct coefficient α and DCT transformation matrix T to predict the missing pixels

3.3 median filtering to improve image quality

The median filter is a type of image filter that can be used to reduce the amount of noise in an image and to enhance the edges in an image. The median filter is good for removing "salt and pepper" noise while preserving the edges of an

image. The median filter replaces the center pixel by the median of its neighbors, covered by a n x n kernel.

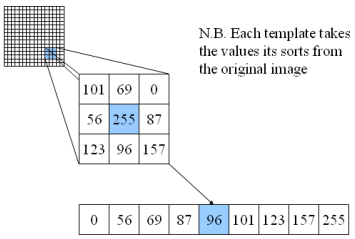


Figure 3: median filter algorithm [2]

Figure 3 is an example that visualizes the algorithm of the median filter. The median filter utilizes a kernel of 3 x 3. The algorithm first sorts all pixel values in this 3 x 3 kernel to find the median which is 96, and then replace the center pixel value (255) by the median (96).

4 Experimental Results

The image recovery is performed on two images: "fishing boat" and "lena". The small image "fishing boat" was broken into blocks of 8x8. The large image was broken into blocks of 16x16. Both images are sampled with five different sample sizes.

4.1 comparison of recovery images with sampled image in different sample sizes

4.1.1 "fishing boat" image recovery

(a) sample (b) recover (c) median filter

Figure 4: sample size = 10

(a) sample (b) recover (c) median filter

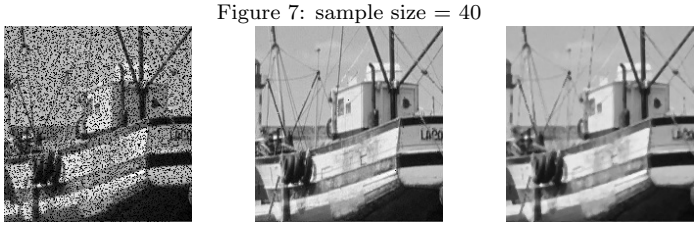
Figure 5: sample size = 20

(a) sample (b) recover (c) median filter

Figure 6: sample size = 30

(a) sample (b) recover (c) median filter

Figure 7: sample size = 40



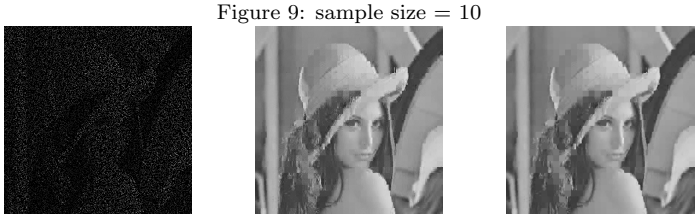
(a) sample (b) recover (c) median filter

Figure 8: sample size = 50

4.1.2 "lena" image recovery

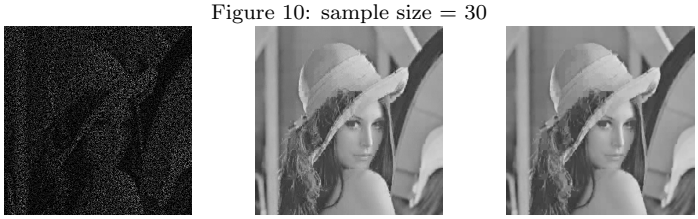
(a) sample (b) recover (c) median filter

Figure 9: sample size = 10



(a) sample (b) recover (c) median filter

Figure 10: sample size = 30



(a) sample (b) recover (c) median filter

Figure 11: sample size = 50

(a) sample (b) recover (c) median filter

Figure 12: sample size = 100

(a) sample (b) recover (c) median filter

Figure 13: sample size = 150

4.2 quality measurement

The recovery quality is measured by calculating the mean square error between original image and the recovered image. Because of the size difference in the "fishing boat" and "lena" images, their block size and sample sizes are different as well. To compare between these two images, sample rate = sample size/block size is used to incorporate the multiple factors that could potentially affect the recovery quality.

Sample Size	Sample Rate	Error without Median Filter	Error with Median Filter
10	0.16	814.41	65.676
20	0.31	402.23	55.175
30	0.47	191.98	45.951
40	0.63	94.528	38.963
50	0.78	37.848	33.961

Table 1: "fishing boat": The error between recovered image(with or without median filtering) and original image

Sample Size	Sample Rate	Error without Median Filter	Error with Median Filter
10	0.039	371.53	45.463
30	0.12	170.43	34.294
50	0.20	100.70	27.503
100	0.39	37.837	18.786
150	0.59	16.169	14.112

Table 2: "lena": The error between recovered image(with or without median filtering) and original image

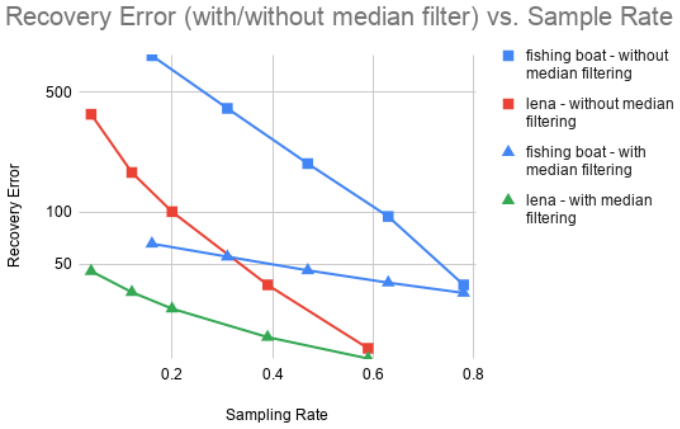


Figure 14: Recovery Error comparison between "fishing boat" and "lena" images

5 Discussion

5.1 recovery quality factors

Figure 4-8 and Figure 9-13 visualize the recovery quality changes as the sample rate changes. We can observe that, for the same image, as the sample rate increases, the model can more accurately reconstruct the original image. The decrease of recovery error can also be spotted in Table 1 and 2. We can also observe that with approximately the same sample rate, the "lena" image has better recovery quality than the "fishing boat" image. Because of the significant size difference between these two images, a larger image would have more sampled pixels for training and testing, which enhance the reliability of the model and the approximate DCT coefficient. That might be one of the reasons that the "lena" image is doing better than the "fishing boat" image. Also, the complexity of the "fishing boat" image is higher than the "lena" image. This may also contribute to the lower recovery quality in the "fishing boat" image. Thus we can conclude that the recover quality is also impacted by image size and image complexity.

From Table 1 and 2, we can observe that the median filter can successfully reduce noise, especially for a lower sample rate where the recovery image higher error. However, the median filter seems to take away finer details. In the "lena" image, we can observe the detail loss in areas like hair and fringes on the hat when comparing the recovery image with or without median filter.

5.2 limitation

In this project, due the limited computing resources, we can only cross-validate for a limited amount of λ. Another limitation is that we are not able to check the bias and variance for the model with single cross-validation.

5.3 future work

Enlarging the training and testing set can improve the stability of the selection result. One way is to do interpolation with the sampled data to generate more data for cross-validation. Another improvement I could make given better computing resources is about the cross-validation I mentioned above.

References

[1] S. Sreeharitha, G. Sabarinath, and B. R. Jose, "Compressive Sensing Recovery Algorithms and Applications- A Survey," IOP Conf. Ser.: Mater. Sci. Eng., vol. 396, p. 012037, Aug. 2018.

[2] "Median Filtering - Computer Vision Website Header." [Online]. Available: https://www.southampton.ac.uk/ msn/-book/new_demo/median/. [Accessed: 22-Apr-2020].