Project 2: Image Recovery

Name : Jiaxin Chen

Email: jiaxinc@andrew.cmu.edu

Abstract

The project can fulfill the image recovery by MATLAB. It can read image(fishing_boat.bmp or lena.bmp) and break image into blocks. And for each block it can sample a few pixels randomly and compute DCT coefficients from the sample pixels by OMP algorithm and cross validation. Finally, it can combine all recovered blocks into a full image and improve its quality with median filter.

1 Overview

This project requires us to recover the provided image from the sampled pixels in MATLAB.

Firstly, we can generate 2-D DCT transform from the original image. And due to the fact that DCT coefficients of a large image are often not sparse, we can break the image into small fixed-sized blocks.

Furthermore, we can utilize randperm function to sample pixels randomly to generate the testing set and training set separately to complete cross validation to obtain the optimal λ for OMP algorithm. And we can repeat the training and testing process for M times.

Eventually, we can apply OMP algorithm with the optimal λ to recover the image block and combine them together. And we can use the median filter to improve the recovered image's quality.

2 Mathematical Formulation

1) DCT Transformation

According to the 2-D DCT transformation theory and the given fixed block size P and Q, we can generate the transformation matrix T:

$$T = \sum_{u=1}^{P}\sum_{v=1}^{Q} \alpha_u \beta_v cos\frac{\pi(2x-1)(u-1)}{2P} \times cos\frac{\pi(2y-1)(v-1)}{2Q} \tag{1}$$

Therefore, the image pixel $g(x, y)$ can be represented by the combinations of transformation matrix T and the DCT coefficients $G(u, v)$. And we obtain the DCT equation:

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

Moreover, we can represent $g(x, y)$ as the pixel column vector C and $G(u, v)$ as the DCT coefficient vector α, which means that we can rewrite the equation above as:

$$C = T\alpha \tag{3}$$

And we can use randperm function in MATLAB to sample pixels to generate the corresponding vector B from vector C as well as the corresponding matrix A from the transformation matrix T. Consequently, we can obtain the under-determined linear system:

$$B = A\alpha \tag{4}$$

And due to α is sparse, we can find the sparse solution α by L_0-Norm regularization. Once DCT coefficients α are computed, we can recover the full image by equation (3).

2) OMP Algorithm

When we obtain the under-determined linear system (4), we can utilize the Orthogonal Matching Pursuit(OMP) algorithm to solve it. And we can rewrite the under-determined linear system as:

$$\begin{aligned} \min_{\alpha} \quad & \|A\alpha - B\|_2^2 \\ S.T. \quad & \|\alpha\|_0 \le \lambda \end{aligned} \tag{5}$$

And we can implement OMP algorithm in the following steps:

(a) Initialize $F = B$, vector $\Omega = \{\}$

(b) Utilize for loop to traverse p from 1 to λ to update α.

(c) In the for loop, we can calculate the maximum $< A_i, F >$ as the largest inner product value and identify its index to update vector Ω.

(d) Utilize the least square fitting to update A and α and use backslash to calculate in MATLAB:

$$\min_{\alpha_i, i \in \omega} \left\| \sum_{i \in \omega} \alpha_i \cdot A_i - B \right\|_2^2 \tag{6}$$

(e) Update F and α_i according to the following equation:

$$\begin{aligned} F &= B - \sum_{i \in \Omega} \alpha_i \cdot A_i \\ \alpha_i &= 0 \quad (i \notin \Omega) \end{aligned} \tag{7}$$

(f) Repeat the process above again and again until p reach to λ, then we obtain the optimal solution α for the $L_0 - Norm$ regularization.

3) **Cross Validation**

Due to the under-determined linear system (5), we need select the optimal λ for the OMP algorithm. Therefore, we need find out the optimal λ by cross validation method as the following steps:

(a) Utilize DCT transformation to generate T and C, and implement randperm function to sample pixels get matrix A and matrix B for cross validation.

(b) Implement randperm function again to select m samples to generate the training set and testing set. And calculate DCT coefficients for the training set.

(c) Estimate approximation error between training set and testing set with mean square for M times and calculate the average error.

(d) Select the optimal with the minimum error for OMP algorithm.

3 Experimental Results

1) fishing_boat.bmp Recovery

We can set the block size 8 × 8 and try five different sample size: 10, 20, 30, 40, 50. And the recovered images of fishing_boat.bmp after median filtering for each sample size is shown as Figure 1:

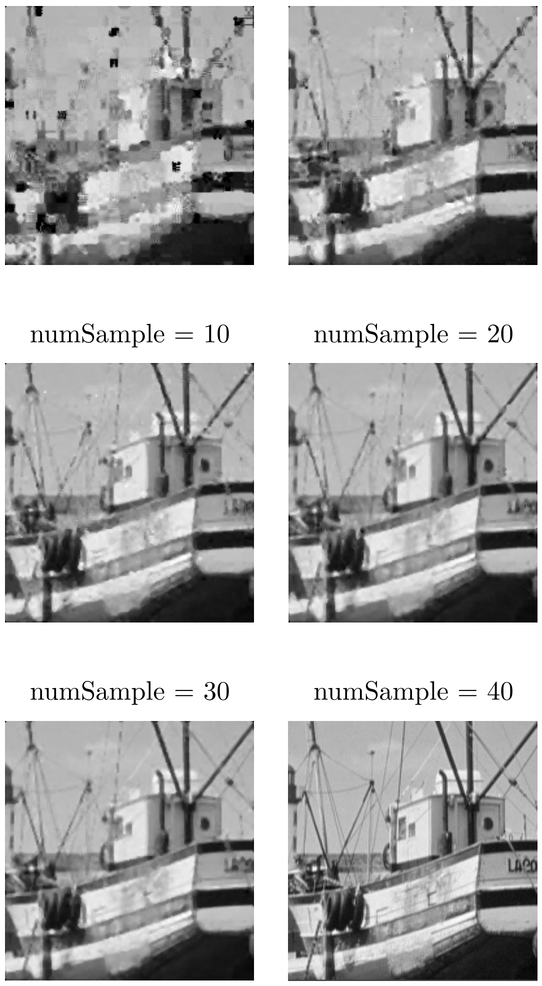

Figure 1: fishing_boat.bmp Recovery

Furthermore, the mean square error and median filter error of fishing_boat.bmp is represented as Table 1:

Table 1: fishing_boat.bmp's Recovery Error

numSample	**Error**	**Median Error**	**Time(s)**
10	2564.5	1349.5	3.5
20	704.2	396.3	3.5
30	306.3	240.9	4.3
40	159.2	177.9	5.5
50	99.7	156.6	6.2

And the relationship between sample number and error is shown as Figure 2:

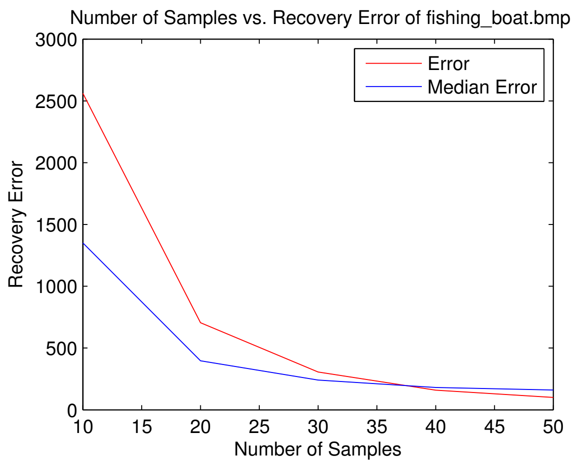

Figure 2: numSample vs. Error of fishing_boat

2) lena.bmp Recovery

We can set the block size 16 × 16 and five sample size: 10, 30, 50, 100, 150. The recovered image of lena.bmp after median filtering for each sample size is shown as Figure 3:

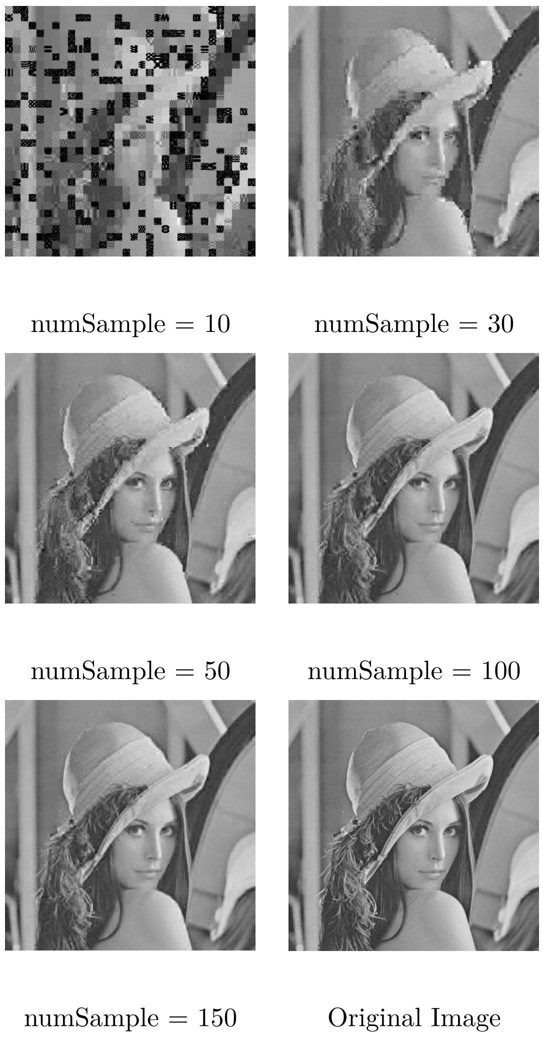

Figure 3: fishing_boat.bmp Recovery

The error and median error of lena.bmp is represented as Table 2:

Table 2: lena.bmp's Recovery Error

numSample	Error	Median Error	Time(s)
10	7671.2	4504.7	12.8
30	318.9	184.9	18.6
50	170.2	111.1	23.6
100	67.2	58.6	37.7
150	55.4	51.6	50.6

And the relationship between sample number and error is represented as Figure 4:

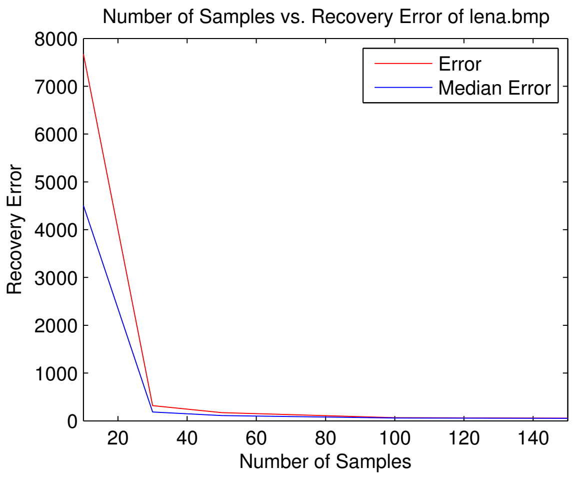

Figure 4: numSample vs. Error of lena

4 Discussion

1) Factors that may impact quality of the recovered image

We can use the mean square error between recovered image and original image to measure the quality of recovery:

$$error = \frac{1}{W \times H} \sum_{\substack{1 \le x \le W \\ 1 \le y \le H}} [\hat{g}(x,y) - g(x,y)]^2 \quad (8)$$

From the Table 1 and Table 2 as well as the plot figure we can find several factors:

(a) The number of sample size can impact the quality of the recovered image dramatically. If the sample size is too small, the image cannot be recovered very well. And both error and median filter's error decrease largely when the sample size increases in the begining. And once the sample number reaches to the certain size, the error won't decrease significantly.

(b) The median filter can improve the quality of the recovered image tremendously when the sample size is small. And one the sample number is up to the certain size, the median error is almost the same as the error, which means median filter may not help improve.

(c) The block size and the lambda selection can also affect the quality of the recovered image.

2) Limits or problems with my approach

Although I've tried my best to optimize the code to improve the accuracy, I can still focus on improving the performance of `CrossValidation.m` and `OMP.m`, which are the key to determine the error of the recovered image.

3) Possible improvements that can be made

(a) My program need invoke `DCT.m` to generate transformation matrix T for each block of the image in the for loop repeatedly, which may waste a lot of time. The possible optimized approach is to compute the DCT only once for the whole image and allocate for each block rather than invoke the `DCT.m` too many times.

(b) We can implement the method to find the optimal block size rather than fixed block size, which may improve the quality of the recovered image.

(c) If we set the sample size too large, the error won't derease significantly any more while the elapsed time is too long. We can also develop the algorithm to find out the optimal sample size to balance the error and the elapsed time.

4) Anything unique I have done to improve/validate program's accuracy/efficiency

(a) Improve Efficiency

I optimize the `OMP.m` code by setting the threshold for norm(F) to avoid meaningless calculation if F is too small, which achieves amazing performance.

18 seconds are the total elapsed time to run `run_fishing_boat_Recovery.m` for 5 different sample size to generate recovered fishing_boat.bmp images. And 140 seconds are the total elapsed time to run `run_lena_Recovery.m` for 5 different sample size to generate recovered lena.bmp images.

(b) Reduce Computation

I used `datasample` function to sample data in the begining. And I test the running time of `datasample` function, which is super time-consuming. I think maybe `datasample` is the high-level encapsulated function. So I changed the more based `randperm` function to sample the random pixels, which can achieve the same accuarcy and improve the efficiency significantly.

(c) Eliminate Warning

I optimize the `OMP.m` code to reduce warning and improve efficiency. Initially I update F just by calculating $F = B - A\alpha$, which is inaccurate, time-consuming and report lots of warnings. Therefore, I recalculate F by setting some condition and calculate for the certain columns of α and the certain rows of A, which improve the performance, reduce the unuseful computation and eliminate warnings dramatically.

(d) Avoid Bad Sampling

Sometimes the error will increase suddenly due to the certain λ when running `CrossValidation.m`, which indicates the bad sampling. Therefore, I set the threshold to check if the error increase too dramatically and meaningless.

(e) Improve Accuracy

At first I tried fixed step size to obtain the λ vector, whose results aren't very satisfactory. Therefore I utilize the increased step size to calculate λ vector for cross validation, which not only reduces the computation largely but also achieves the much better accuracy.

References

[1] Efron B., *"Estimating the error rate of a prediction rule: improve- ment on cross-validation."*, J. Am. Stat. Assoc., 78:316–331,1983.

[2] A. Fletcher and S. Rangan. *"Orthogonal matching pursuit from noisy random measurements: A new analysi"*. Advances in Neural Information Processing Systems 22 , pages 540–548. 2009.