Project 2: Image Recovery

<Jin Gu>
<jingu@andrew.cmu.edu>

Abstract
In this project, we are given two images. One is a small image named "fishing boat", another is a bigger image named "lena". Our task is to recover the images with partial pixels of these images with OMP algorithm. What is more, we will apply cross validation and medium filter to improve the accuracy.

1. Overview
In order to recover an image, it is better to divide the whole image into many small blocks. Therefore, the first thing I did is divide the image into blocks. It is not very hard to implement with for loops if you know the size of the block. But it is convenience to have a size which can be divided exactly by the row number and column number of the data image. After we get a small block, I got several sample points from the block as my starting points. Then I separate these sampling points into training data and test data to do a cross validation in order to find the best lambda for OMP algorithm. Finally I apply OMP algorithm to recover the image with the best lambda. Therefore I got a block of recover data. Then I changed to recover another block until the end.
Because we will apply 2-D DCT transform to recover the images and the matrix is fixed given the block size, I calculated the DCT transformation matrix before recovering the data.

2. Mathematical Formulation
Describe how you formulate the image recovery problem as an under-determined linear system. Describe how you implement the OMP algorithm.

2.1 Calculate the DCT transformation matrix T
According to 2-D DCT transform theory, an image pixel can be represented by the combinations of DCT coefficient which can written as:

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

In the equation above, g(x, y) is an image pixel and G(u, v) is DCT coefficient. The left term is the DCT transformation matrix T. Therefore, we can write it into matrix form.

$$C = T\alpha$$

Here, C is a pixel column vector, α is the DCT coefficient vector.

From the equations above, matrix T is fixed if the P, Q are given. Here P, Q is the block size. It is not very hard to calculate it from the equations.
After we get matrix T, I sampling the matrix C and T to get partial pixel matrix B and partial transformation matrix A. And the relationship between A, B and α is:

$$B = A\alpha$$

And this is **the under-determined linear system** we need to solve in this project. Therefore, we can calculate α with OMP algorithm and finally get the recover data matrix C with the equation C = Tα.

2.2 Cross validation
Because in OMP algorithm, we need to solve the problem like this:

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

We need to find the best lambda for OMP algorithm. Cross validation is a good method to solve this problem.
After getting sampling points matrix B, I separate them into training data and test data, then I use training data to do the OMP algorithm with a lambda in a list of lambda. Here the list of lambda I used is from 3 to number sample plus 10 increasing by 3. After doing OMP algorithm to get the recovered data with all lambdas, I calculate the error between recovered data and original data with each lambda. And the lambda with minimum error is the best one.

2.3 OMP algorithm
Because we get an under-determined linear system above, we can solve it with OMP algorithm.
In OMP algorithm, I followed several steps to solve the under-determined problem

$$\min_{\alpha} ||A\alpha - B||_2^2$$
$$S.T. \ ||\alpha||_0 \le \lambda$$

Firstly, I initialize the vector F as vector B.
Then I used a for loop to find the maximum inner product of A_i and F and record the index i in a vector Ω.
Then I used backslash "\" to calculate the α in problem

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

Then I updated F by the equation

$$F = B - \sum_{i \in \Omega} \alpha_i \cdot A_i$$

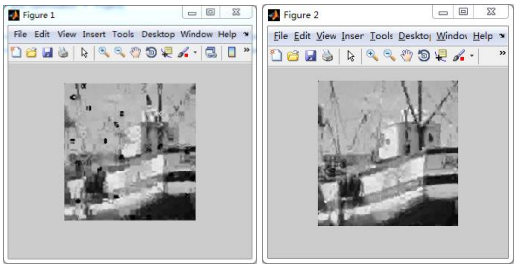
Finally, I check if the iteration time is bigger than the given lambda, if yes, I stop there, if no, I continue the progress from calculating the maximum inner product of A_i and F.

3. Experimental Results

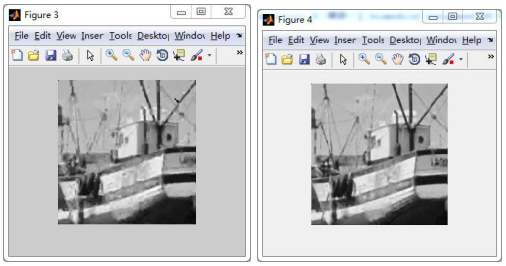
For the small test image fishing boat: set block size 8x8 and try five different sample size, i.e., S=10, 20, 30, 40, 50. Show your recovered images after median filtering for each sample size. Show the mean square error (as defined in the slides) between the recovered images and the original image. Provide a figure of recovery error vs. number of samples for both configurations, i.e., with median filtering and without median filtering.

3.1 Result of image "fishing boat"

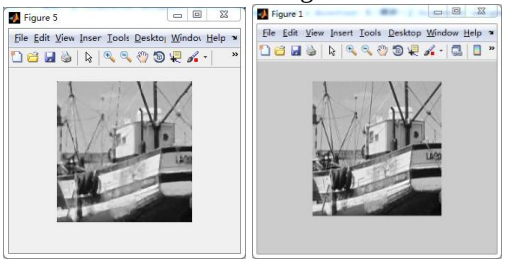
Size = 10 Size = 20



Size = 30 Size = 40



Size = 50 Original



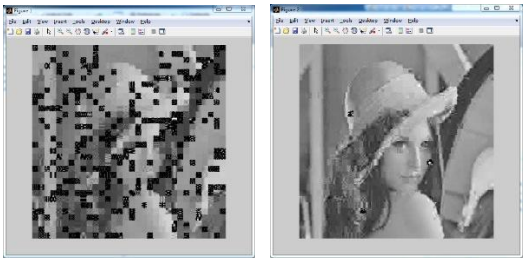
Error table

Size	Best lambda	Error	Error(medfilter)
10	6	2611.82	1374.5
20	9	676.22	441.36
30	33	742.63	362.78
40	33	196.84	213.22
50	9	98.32	158.77

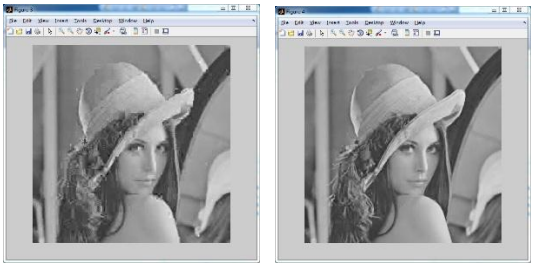
For the large test image lena: set block size 16x16 and try five different sample size, i.e., S=10, 30, 50, 100, 150. Show your recovered images after median filtering and recovery error.

3.2 Result of image "lena"

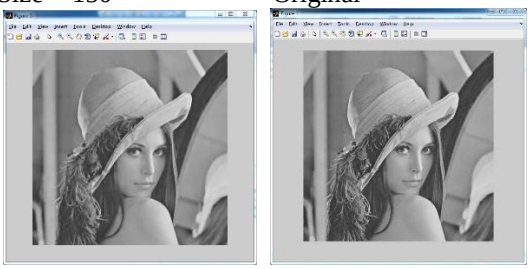
Size = 10 Size = 30



Size = 50 Size = 100



Size = 150 Original



Error table

Size	Best lambda	Error	Error(medfilter)
10	9	10318.12	5380.42
30	18	666.38	269.28
50	54	164.36	119.96
100	15	66.14	53.05
150	9	33.41	32.19

3.2 **Conclusion**

From the result above, we know that the images can be well recovered, it means that our project achieves an excepted goal. And I found several interesting things.

1 There may be a minimum value for sampling number to get an "acceptable" result. In other words, when the sampling number reach to a certain point, the error will not reduce significantly which means we can know more than 80% information of the original image from the result image. I found this minimum value should be 35%-45% of all data points.

2 Medium filter may not always help

From the error table, we can find that the med-error is not always smaller than original error, especially when sampling number is big.

3 Best lambda is irrelevant to sampling lambda and may be fixed relatively.

4. Discussion

Please try to interpret your experimental results. Topics may include, but not limited to:

- Factors that may impact quality of the recovered image
- Limits or problems with your approach
- Possible improvements that can be made
- Anything unique you have done to improve/validate your program's accuracy/efficiency

4.1 Factors that may impact quality of the recovered image

From the result we can know, there are several factors will impact the result, which are the sampling number, block size and lambda. And sampling number is the primary factor.

4.2 Limits or problems with your approach

The main problem of my approach is time limit. If the image is very big, it needs a very long time to get a result. And I already tried my best to optimize the code. But it still takes a long time if the image is very big.

4.3 Possible improvements that can be made

The first improvement we can make is to speed up the code. What is more, we used a fixed block size in this project, maybe we can develop an algorithm to find the best block size. Finally, the relationship between lambda and project parameters is also a very interesting topic to improve the result.

4.4 Anything unique you have done to improve/validate your program's accuracy/efficiency

1 Tricks of speeding up the code

Because the code is very slow with a big image, I did several optimizations. For example, I use matrix operation to do calculation as possible as I can for the reason it is faster, and I reduce repeated calculation. For instance, I calculated the DCT matrix once instead of calculating it many times.

2 Avoiding bad sampling

From the project, I found that sometimes the error of a special lambda calculated in cross validation will increasing suddenly. I think it should be a problem of bad sampling. To solve these problem, I used a judgment to check if the error increased significantly, if yes, I reset the error as 10 times of the average error. Here are what I did.

```
dif = norm(testData - predictData);
if((count ~= 1 )&& (dif > 10 * error(lam, :) / (count - 1)))
    error(lam, :) = error(lam, :) + 10 * error(lam, :) / (count - 1);
else
    error(lam, :) = error(lam, :) + dif;
end
```

References

[1] Use an enumerated list here for any references, such as books or journal/conference papers.