Project 1: Image Recovery

<Haoyun Chen>
<hc123@duke.edu>

Abstract

This project applied compressed sensing to recover full images from a small number of sampled pixels. It used 2-D DCT (Discrete Cosine Transform) algorithm to generate frequency distributions of the images. OMP (Orthogonal Matching Pursuit) algorithm were used to get sparse solution of frequency coefficients. We used random subsets to do cross validation to get the best regularization coefficient. Then we did inverse DCT transformation to recover the image. In this project, we recovered 2 images by using different numbers of samples and compared the result under different conditions.

1. Overview

In this project, we wanted to recover pictures by limited number of pixel samples. We divided each picture into 8 by 8 or 16 by 16 pixel value matrices. On each of those blocks, we sampled a fixed number of pixels and aimed to recover them by using DCT and OMP algorithm. After recovering each block, we combined them to a whole image.

DCT (Discrete Cosine Transform) is widely used in 2D image processing. It does bidirectional mapping from pixel value domain to frequency domain. By DCT, even if we only have limited number of samples on pixels, we still can learn enough number of frequency values to generate the recovered image. After we got satisfied sets of frequency values, we applied inversed DCT to recover the image.

OMP algorithm, which is also known as a compressed sensing algorithm, was characterized by its simplicity and efficiency. It is a heuristic algorithm to solve the L0-norm regularization problem. We applied L0-norm regularization because in image recovery problems, it is usually considered that the frequency values of images were sparse. By using OMP, we can identify a subset of DCT coefficients that are non-zero.

Also, we had to determine the best regularization parameter lambda of each block. Because we applied L0-norm regularization, the regularization parameter simply represents the number of non-zero values of the frequency domain. We got the best lambda by doing cross validation. The lambda which provided the smallest MSE (Mean Squared Error) can be thought as the best one in the current block. Since different blocks would generate different best lambda, we recovered different blocks based on different lambda.

After we recovered all the blocks and combined, we discovered there was obvious discontinuity between different blocks. Therefore, we added a median filter to the image to smoothen it and made it more visually friendly.

As a result of the project, we successfully recovered two grayscale images, which were 'boat.bmp' (192*200) and 'nature.bmp' (512*640). We also researched about the improvement of the process, such as multithreading and changing the block size to reduce the running speed, and using multiple methods to reduce the MSE of the recovered image.

2. Mathematical Formulation

To reduce the dimension of the images, we first divided each image into 8*8 or 16*16 blocks. Then we applied DCT to each block one by one. The equation of 2D DCT is shown below:

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

Image pixel | Transformation | DCT coefficient

$$C = T\cdot\alpha$$

$$x,u\in\{1,2,\cdots,P\}$$
$$y,v\in\{1,2,\cdots,Q\}$$
$$\alpha_u=\begin{cases}\sqrt{1/P} & (u=1)\\ \sqrt{2/P} & (2\le u\le P)\end{cases}$$
$$\beta_v=\begin{cases}\sqrt{1/Q} & (v=1)\\ \sqrt{2/Q} & (2\le v\le Q)\end{cases}$$

In the equation, g(x, y) is the pixel values on the position (x, y). G(u, v) is the DCT coefficient in the frequency domain on the position (u, v). α and β are simply coefficients determined by P and Q, where P and Q are the row and column number of the blocks. (e.g. P and Q are both 8 when the block is 8*8.) The two cosine coefficient are determined by current x, y, u, v values. Since g(x, y) is always a linear combination of G(u, v), we can write it as the form of C = Tα. C is all the pixel values, whose dimension is the total number of pixels. T is the transformation matrix. Each row of T fixes x and y, and each column of T has fixed u and v. We observed that the pixels were in 2D blocks and we wanted to convert their values into 1D in g(x, y). So we had g(x1, y1), g(x1, y2) up to g(x1,yQ) as the first Q elements. Then we had g(x2, y1) up to g(x2,yQ) as the next Q elements. Then the last element would be (xP, yQ) so that there are totally P*Q elements in C. This is similar for each row of T and α, except that u and v were changed but not x and y. Since T is constant, we formed an equation to recover all the pixels C. What we want is to learn an approximate set of DCT coefficient, which is α.

If we only have part of information of C, we can for an under-determined linear equation and apply L0-regularization to approximate α. The under-determined equation is shown below:

After sampling, we have B representing the sampled data, and A representing transformation on those sample data. Then we applied OMP algorithm to solve the approximated α.

In OMP algorithm, we set a regularization parameter λ. We iteratively chose a subset of A which best approximates B by

linear combination of a subset of α and add to the subset of A with 1 column of A which best approximate the error each time. The iteration stops when we have learned λ elements of alpha. This can be formulated as the following steps.

Step 1: Set F = B, Ω = { } and p = 1. Step 2: Calculate inner product values θi = <Ai, F>. Step 3: Identify the index s for which |θs| takes the largest value. Step 4: Update Ω by Ω = Ω ∪ {s}. Step 5: Approximate F by the linear combination of {Ai| i ∈Ω} by applying least square method to Aα = B, and $F = B - \sum_{i\in\Omega} \alpha_i A_i$. Step 6: If p < λ, p = p+1 and go to Step 2. Otherwise, stop.

We determined the best λ by applying cross validation. In each block, we did cross validation on λ from λ=1 to the number of samples. (λ cannot be larger than pixel value. Otherwise, the algorithm would have infinite solution and become unstable.) In each iteration, we randomly divided the samples into training set and test set. We set the number of the test set as floor(S/6), where S was the number of samples in each block. We trained the coefficients by the training set and calculated the MSE by the test set. We did this 20 times for each λ and calculated the mean of MSE. We chose the λ with the smallest mean of MSE as the regularization parameter of the block.

After we got the best λ and α, we converted DCT coefficients α to the pixel values of the block. Then we combined the block together. Finally, we used median filter to the recovered pixels. This filter looked the current pixel and all its neighborhoods, and convert its value to the median of the neighborhood. This will make the image smoother.

3. Experimental Results

The results can be got by the 'main.m' program.

For the small test image boat.bmp, we set block size 8x8 and try five different sample size S = 10, 20, 30, 40, 50. Each recovered images after median filtering are shown below.

S = 10

S = 20

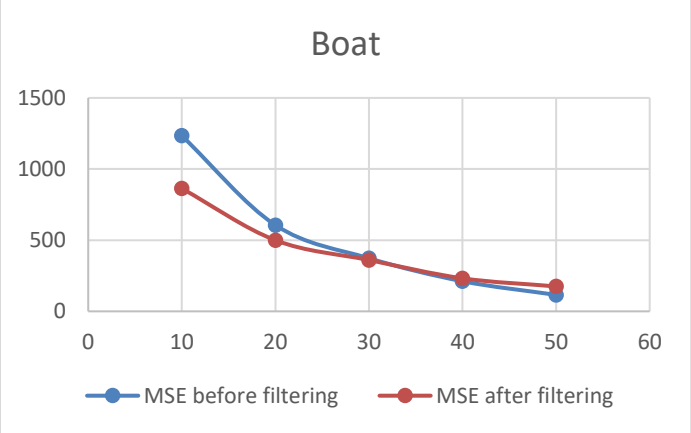
S = 30

S = 40

S = 50

The MSE between the recovered images (with filter and without filter) and the original image are shown below.

boat		
sample size	MSE before filtering	MSE after filtering
50	115.5545	175.0252
40	211.4166	231.9555
30	373.1368	360.4388
20	606.0514	499.8213
10	1234	863.1633



For the large test image nature.bmp we set block size 16x16 and tried five different sample size S = 10, 30, 50, 100, 150. The recovered images after median filtering and recovery errors are shown below.

S = 10

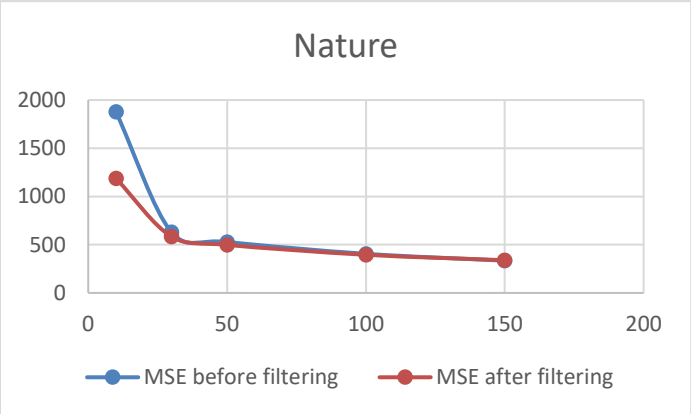
S = 30

S = 50

S = 100

S = 150

Nature		
sample size	MSE before filtering	MSE after filtering
10	1876.8081	1186.918
30	630.5287	584.008
50	529.0022	498.0824
100	405.3307	395.5612
150	334.4054	338.9124



From MSE of the 2 images, we observed that MSE decreased with the increase of sample size. Also we observed that when the sample size is large enough, the median filter would not improve the MSE. However, we could observe obvious visual improvement when comparing the two figures with and without filters.

4. Discussion

The quality of results of the recovery was determined by several factors.

First, the sample number will directly affect MSE of the result. The more samples we have, the less MSE the result will have. Despite this, we cannot recover the image perfectly, even if we had all the pixels sampled. This is because OMP is only an approximation of L0-norm regularization and cannot perfectly solve a determined linear equation.

Second, the median filter will influence MSE in different situations. When MSE before filtering is large, the filter can help reduce MSE. However, when MSE before filtering is small enough, the filter will increase MSE. This is because when MSE is large, there are many abnormal values in the image. The filter will help choose a value within reasonable range. When MSE is small, the recovered images contain more detailed information such as thin lines and characters. The filter will remove those details by smoothing the image and cause the MSE larger.

We have also done some improvements to the algorithm. First, we observed that since we divided the image into many small blocks and trained each α of the blocks separately, we could do this job in parallel. We put the process of each block into one for loop, and changed 'for' to 'parfor'. 'parfor' executes for-loop iterations in parallel. This can greatly reduce the training time. Take my computer as example, 'parfor' made concurrency on six cores and reduced the execution time of the whole program by nearly 6 times (originally around 18h and more than 3h after concurrency).

Second, we tried to reduce the block size, and observed that changing the block size will not greatly affect MSE when the ratio of sampling is not changed. In our experiment, for example, in comparison with 16*16 blocks in 'nature.bmp' with 150 samples per block, we divided it into 8*8 blocks with 36 samples per block. (36/64 is approximately equal to 150/256.) And we found those 2 had very similar MSE. (The one with smaller block had MSE of 337.2877.) However, the run time of the method with small blocks was much shorter. From our result, the run time decreased from 11000s to 400s. This is because although the number of blocks has increased by 4, the sample size, thus the range of λ is also decreased by 4. Also, the dimension of the transform matrix changed from 256*256 to 64*64. When the lambda and column of A gets large, the cost of doing least square regression by applying the inverse of {Ai} would get much larger. These factors caused the larger blocks consumes much longer training time.

There are still some problems needed to be addressed in the project. The median filter we used in the project became not suitable enough when MSE is small. So we need filters that could remain the details in those situations. High pass filter and Laplacian filter, which are known to reserve the high frequency information were applied. However, they cannot decrease the MSE either. The gradual change in the images would be omitted.

We can also make reutilization of samples of pixels when recover the image. After we recover the image from inverse DCT, we can directly apply the sampled values to our recovered image. And the error of the sampled position will be 0. This can greatly reduce MSE

References

[1] DKU ECE580 lecture notes 7&8, Xin Li.
[2] MATLAB image filtering: https://ww2.mathworks.cn/help/images/linear-filtering.html.