COMPRESSED SENSING APPLICATIONS IN LARGE IMAGES

Lihan Zha
Weiyang College,
Tsinghua University
Beijing, China
clh20@mails.tsinghua.edu.cn

ABSTRACT

Compressed Sensing (CS) is super-powerful in reconstructing signals from highly-imcomplete data samples. This paper focuses on two different approaches to preprocess true color images for CS, and how to deal with memory limitations when applying CS to reconstructing large images. Convex optimization is introduced in this paper and are tested with numerical examples.

***K*eywords** Compressed Sensing, Large Images, Convex Optimization, OWL-QN

1 Introduction

Due to memory limitations, traditional approaches of Compressed Sensing (CS) become impossible for ordinary computers, when size of images is large, usually larger than 10,000 pixels. Hence, optimization techniques have to be introduced to ensure scalability of CS. Moreover, techniques of dealing with true color images are not frequently discussed, which will also be handled in this paper.

This paper first briefly introduces the basic principles of CS. Then, Convex Optimization (CO), one of the most classic CS algorithm, is discussed, and the problem of CO is pointed out when handling large images, followed by the technique to solve it. What's more, approaches to preprocess true color images are also discussed in Formulation section. Furthermore, all approaches discussed are tested in section Numerical Tests, and conclusions and discussions are included in the last section.

2 Formulation

2.1 Basic Principles of Compress Sensing

The basic reason that Compressed Sensing (CS) can work is that images we really care about only account for a really small percentage of the vast pixel domain. Those images that people see in daily life are all continuous, which means that the adjcent columns of image matrix are really similar, thus image matrices in real life are all low-ranked. It makes it possible that we can only measure a small percentage of the whole image while reconstructing it to a satisfactory level.

Compared to Nyquist–Shannon sampling theorem, CS does not take samples evenly. Instead, CS take samples of signals randomly. The randomness guarantees that we can take much sparser samples than required by Nyquist–Shannon sampling theorem, which can lead to much more efficient sampling.

The core idea of CS is that we can transform images to a domain where their expressions are sparse, and find the expressions of them in the sparse domain. Then, through direct inverse transformations, we can reconstruct the images. With math formula, this can be expressed as:

$$y = \Phi\Psi s \tag{1}$$

where y stands for the vector of samples from the original signals, Φ stands for the sampling matrix, Ψ stands for the transformation matrix, and s stands for the sparse expression of the original signal in its sparse domain. In 2D cases,

Discrete Cosine Transform (DCT) is the most common transformation, which is adopted in this paper. 2D-DCT can be constructed by applying 1D-DCT respectively to columns and rows of the matrix.

2.2 Convex Optimization

The goal of CS is to find the most appropriate solution s, that is,

$$\underset{s}{argmin} \, ||\Phi\Psi s - y||_2 \tag{2}$$

However, (1) is an underdetermined equation since s is of higher dimensions than y, which means there are infinite solutions that satisfy (2). Remember that we want to find as sparse solutions as possible, which means that there should be as many zeroes in s as possible, or at least most elements of s should be around zero. Thus, we can rewrite our goal, as in (3):

$$\underset{s}{argmin} \, ||\Phi\Psi s - y||_2 + \lambda||s||_1 \tag{3}$$

where λ is a constant that is adjustable. If λ is too big, then s may forget the characters of the original image. If λ is too small, then s may not be sparse. Hence, it is necessary to find appropriate λ through trial-and-error.

The reason why L_1 regularization is adopted here is vividly shown in Figure 1, cited from Steven Brunton[1]. L_1 regularization calculates the sum of absolute values of elements in all dimensions. When the sum is given, the values in few dimensions so that L_1 norm is the smallest, which leads to the sparsest solution. While for L_2 regularization, when the sum is given, values are distributed in all dimensions so that L_2 norm is the smallest, which more likely leads to uniformly-distributed solutions instead of sparse solutions. Hence, L_1 regularization is preferred here.

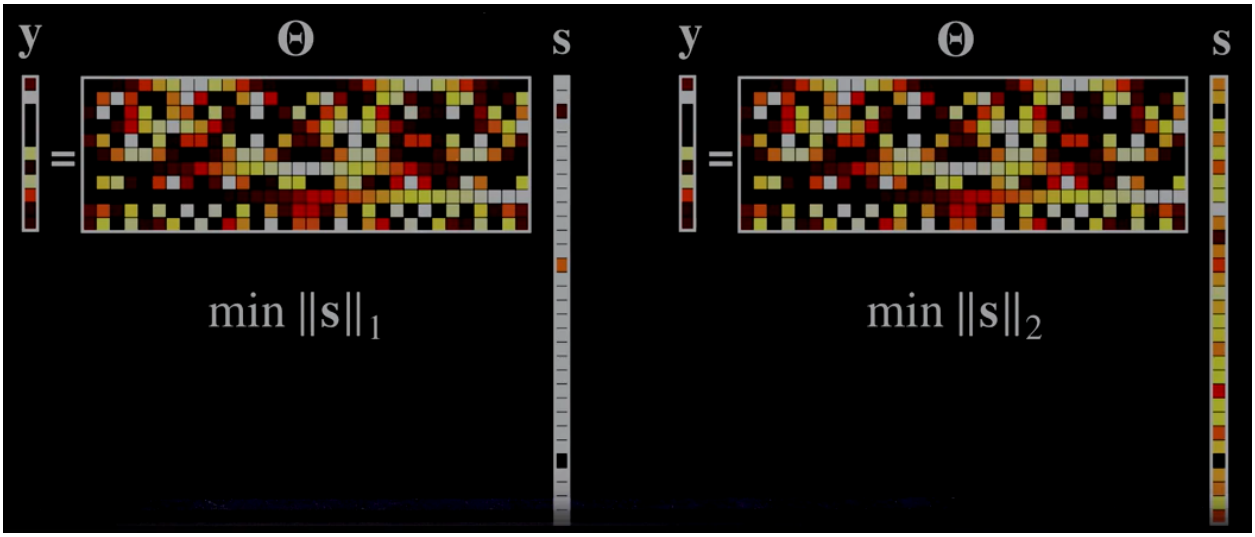

Figure 1: Cited from Steven Brunton. L_1 regularization gives the sparsest solution, while L_2 regularization distributes values among all dimensions and does not lead to sparse solutions.

2.2.1 Problems and Optimizations of Convex Optimization

According to (3), let $A = \Phi\Psi$, it is inevitable to calculate A when adopting Convex Optimization algorithm. However, A may become truly massive when the images get larger, and result in memory allocation errors. To solve this, one straight forward approach is that we zoom resize the original large image to a smaller size at a cost of intelligibility. This can be achieved through bilinear interpolation of images. Another idea is that we can calculate the gradient of convex loss objective function f, which can be defined as whatever we want, here we defined f as:

$$f(s) = ||As - y||_2^2 + \lambda||s||_1 \tag{4}$$

The gradient of which is:

$$\nabla f(s) = 2(A^T As - A^T y) + \lambda \cdot sgn(s^T) \tag{5}$$

[1]Data Driven Science and Engineering, Steven Brunton

Only $A^T As$ and $A^T y$ is concerned here, which means we do not to calculate A at all. The realization of this approach is also known as OWL-QN algorithm[2]. Due to time limitations, this paper only introduces this approach rather than adopt it. In Section 3, image resizing is adopted to deal with large images.

2.3 Preprocessing of true color images

Common CS algorithms are ready-made for gray images. To apply them to true color images, preprocessing of images are necessary. Next, two different approaches of preprocessing are introduced respectively. Both approaches can be applied to different image formats including JPEG, JPG and PNG.

2.3.1 Array-To-Integer Mapping

This approach is something like transformations between different number systems. We can map the 3-channel color array to an unique integer through the following law:

$$F(R, G, B) = R + 256 \times G + 256^2 \times B \tag{6}$$

Through (6), any unique combination of (R, G, B) can be uniquely represented by an integer F(R,G,B), which means 3-channel true color images can be converted into some 1-channel image, and we can easily make inverse conversions based on the bijective mapping defined in (6).

2.3.2 Separation of channels

This approach is very straight-forward. We just apply CS to 3 channels of true color images respectively and then put the results together.

Both of the two approaches will be tested in Section 3 to see which performs better.

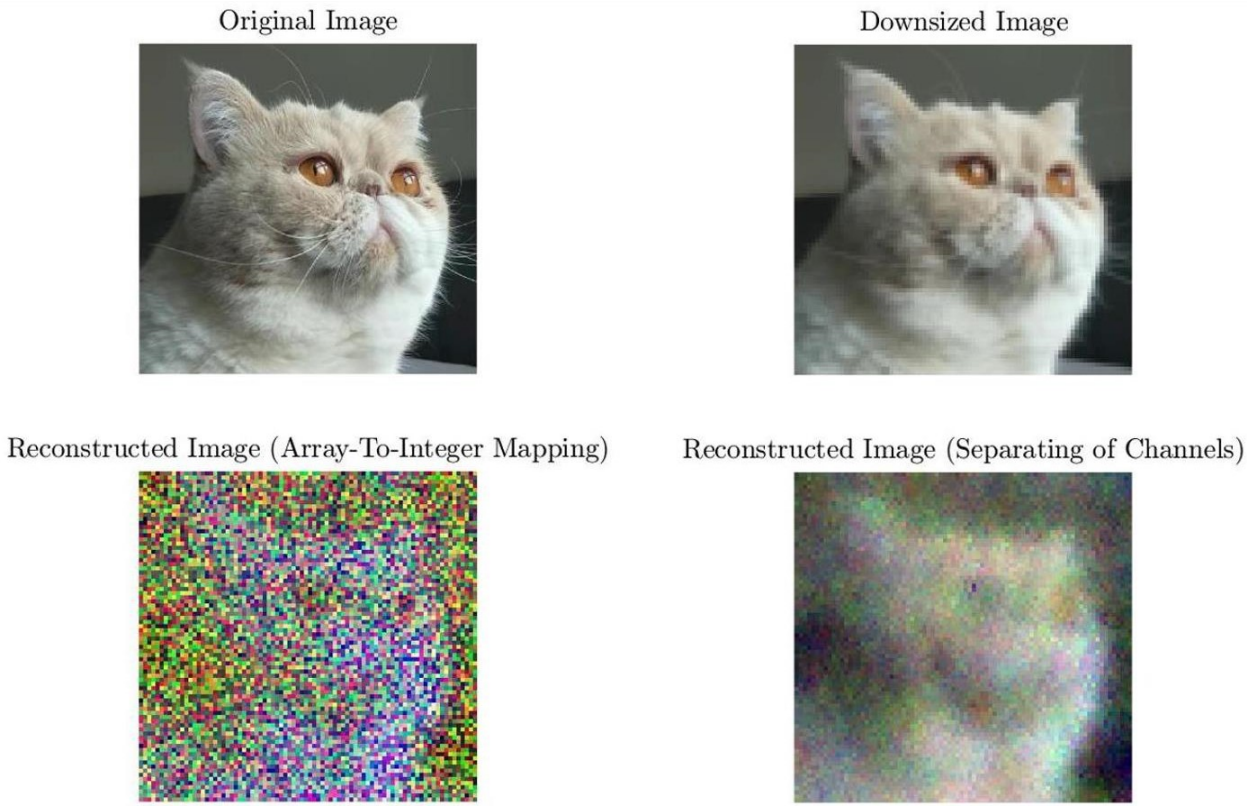

Figure 2: The Big Tiger image downsized and reconstructed by two different approaches using 30% pixels of the downsized image.

[2]http://www.pyrunner.com/weblog/2016/05/26/compressed-sensing-python/

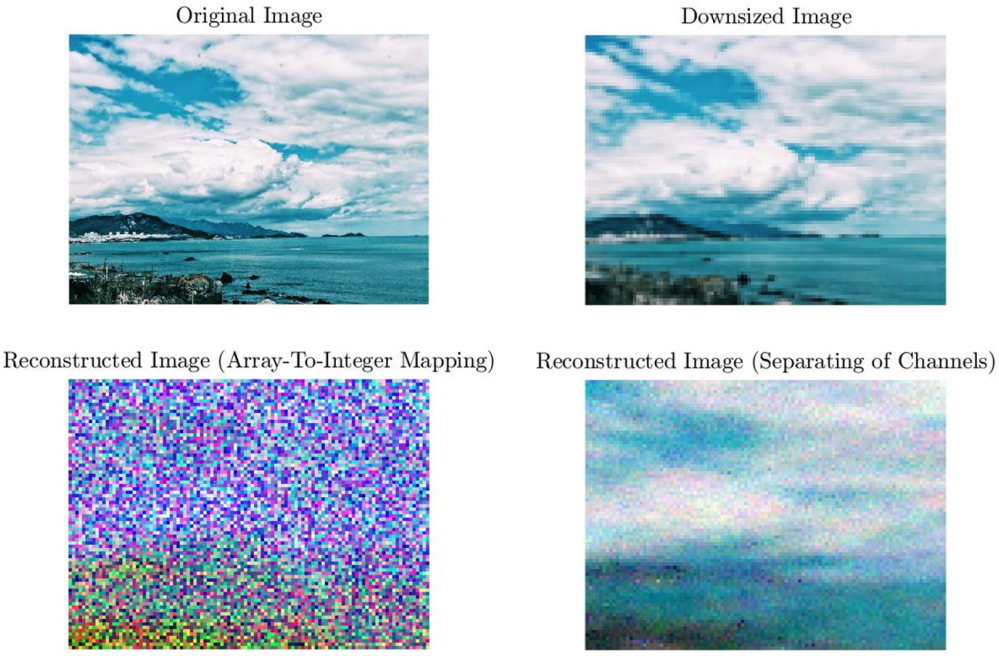

Figure 3: The Sea and Sky image downsized and reconstructed by two different approaches using 30% pixels of the downsized image.

3 Numerical Tests

Two images are adopted for numerical tests. The first image is the Big Tiger. The original size of the Big Tiger is 837×818, and it is downsized to 80×78 utilizing bilinear interpolation. At first, 30% pixels are sampled from the original image, and two different preprocessing approaches are respectively adopted. See Figure 2. Clearly, Array-To-Integer Mapping perform badly compared to Separation of Channels. Too many colorful noises arise, and the reason may be that a small error in $F(R, G, B)$ can cause a great bias to the value of the red channel, according to (6). Separation of Channels approach does not give rise to this problem because all channels carry the same weight, and the error evenly distributed among all channels. The second image is Sea and Sky. Similar conclusions can be drawn from Figure 4.

Different percentage of pixels sampled leads to different qualities of reconstructed images. Of course, the more pixels we sample, the more intelligible the reconstructed image is. Figure **??** shows the quality of images reconstructed from three different percentages of pixels sampled. Obviously, the image taking 50% samples approximate the original image best, while the image taking 1% pixels is hardly intelligible.

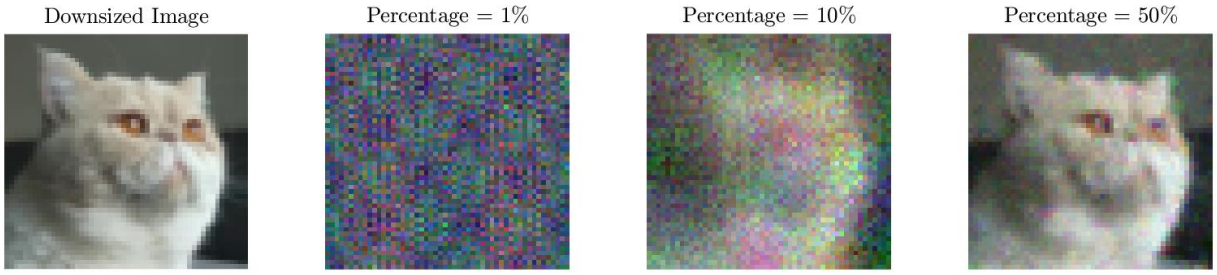

Figure 4: The Big Tiger image reconstructed through Separations of Channels preprocessing approach using 1% , 10% and 50% of pixels of the downsized image.

4 Conclusion

This paper focuses on how to apply Compressed Sensing to large images and how to preprocess true color images. Through bilinear interpolation, large images can be resized into smaller images at the cost of poorer intelligibility. Otherwise, OWL-QN algorithm can be adopted to maintain the resolution of the original image and implement Compressed Sensing as well, which is not tested in this paper. Separating channels is proved to be a more efficient approach in preprocessing true color images compared to Array-To-Integer Mapping approach. Through numerical tests, it is amazing to see that Compressed Sensing really works. Actually, many advanced algorithms have been proposed since the proof of Compressed Sensing in 2006. The Convex Optimization algorithm adopted in this paper is very junior, and serves as a start for me to comprehend the wonderful Compressed Sensing tool.