Image Compression Performance Comparison Using Total Variation Minimization and Noiselet Transform

Devin Cornell

April 29, 2015

Abstract

This project explores the use of Total Variation Minimization as a method of reconstruction for sparse-coded images. Both the Discrete Cosine Transform and Noiselet Transform compression algorithms were combined with Least Squares and Total Variation Minimization recovery algorithms to reproduce two different source images. Comparisons were performed by both qualitative and quantitative measures, and it was found that while the Noiselet Transform combined with Total Variation Minimization can have a dramatic effect on performance for certain images as found in literature, it can produce very poor results for some images that make it a less consistent option for compression than encodings such as the Discrete Cosine Transform.

Contents

1 Introduction

This project involves the creation of Matlab code to implement algorithms for Discrete Cosine Transform (DCT) and noiselet transforms reconstructed by least squares or total variation minimization. The compression and reconstruction algorithms will be compared with two different images for their ability to reconstruct the image in a way that has a strong visual resemblance to the original image, and also compared quantitatively to the original image to determine the signal noise power that is introduced into the system after compression and reconstruction.

2 Background

2.1 Sparse Encoding Theory

Consider the following transformation $T(x)$, where x represents some original image. Note that the image is expressed as a vector for simplicity of expressing linear operators.

$$T(x) = \Phi x = y \tag{1}$$

And when $T : \Re^n \rightarrow \Re^n$, the compression is lossless because it can be recovered exactly. In the case that $\Phi \in Re^{n \times n}$ and Φ is positive semi-definite, the recovery method is obvious and depends on the inverse transform $T^{-1}(y)$.

$$T^{-1}(y) = \Phi^{-1} y = x \tag{2}$$

But in order to be useful as a practical compression algorithm, the input vector x should be represented in a basis set that is smaller than the original basis. This gives $T : \Re^n \rightarrow \Re^m, \Phi \in Re^{m \times n}$. This is in general a lossy transformation because the original x cannot necessarily be recovered, but in practice can be recovered for some x. This fact leads to the obvious solution to the recovery problem.

2.2 Least Squares Recovery

Given that the transformation system $T(x)$ is under-determined, least squares is the most obvious recovery method. The least squares approximation is given by equation (3).

$$\hat{x} = \Phi^T (\Phi \Phi^T)^{-1} y \tag{3}$$

This is actually an approximation solution to the problem given by equation (1) where Φ is an under-determined system. Later it will be shown how this fact is taken advantage of in the case of a cosine-based Φ for the Discrete Cosine Transform (DCT).

2.3 Recovery Using $l1$ Minimization

As an alternative to least squares $l2$ recovery, $l1$ minimization can also be used to recover an image from transform coefficients. The general $l1$ minimization problem applied to this reconstruction is posed as a constrained linear programming optimization problem as given in equation (4), given in [2].

$$\hat{x} = \min_{x'} \|\Psi x'\|_{l_1} \quad subject\ to \quad \Phi x' = y \tag{4}$$

In equation (4), Ψ is a linear operator that performs a discrete wavelet transform on the basis image, and the $l1$ norm for some vector u is given in equation (5).

$$\|u\|_{l_1} = \sum_{i=1}^{n} |u_i|, \quad where \quad u = [u_1\ u_2\ ...\ u_n]^T \tag{5}$$

As we will see in the next section, the minimization in equation (4) can be given as a minimization of Total Variation.

2.4 $l1$ Minimization Recovery as Total Variation Minimization

In the case of the image compression problem, the general wavelet transform given as Ψ can be set as the image gradient, and therefore the $l1$ minimization problem can be expressed as the minimization of total variation, which is the sum of gradient magnitudes across the image. Note that instead of expressing the candidate image as a vector x', equation (6) (from [2]) expresses the candidate image X' as a two-dimensional discrete scalar function, to illustrate the nature of the gradient operator.

$$\hat{x} = \min_{x'} \sum_{i}, j|(\bigtriangledown X')_{i,j}| \quad subject\ to \quad \Phi x' = y \tag{6}$$

It will be demonstrated later that using the total variation (TV) minimization shown here, an image can be recovered from a set of sparse encodings in a way that produces an overall more accurate representation than the least squares method.

2.5 Discrete Cosine Transform for Image Compression

The most central part of the JPEG original standard was the Discrete Cosine Transform (DCT) [1]. This transform is largely similar to the Fourier transform, except that it only uses real values. In Figure 1 the DCT of cameraman shows that most of the energy of this image is contained in the low frequency components represented by the pixels at the top-left.

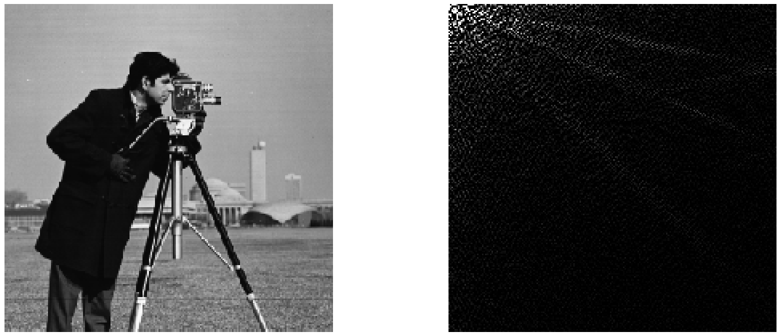

Figure 1: The famous cameraman photo on the left, the cameraman DCT on the right.

The JPEG algorithm is able to create a sparse representation of images by selecting only some of the coefficients in this DCT transform to store. The fact that most of the energy in a typical image is stored in low-frequency components is taken advantage of for compression. The JPEG algorithm provides an order of coefficient selection which begins at the top-left with the DC component of the image, and zig-zags across the image in such a way that the frequency of the newly added components is minimized. See Figure 2 for details on this scheme.

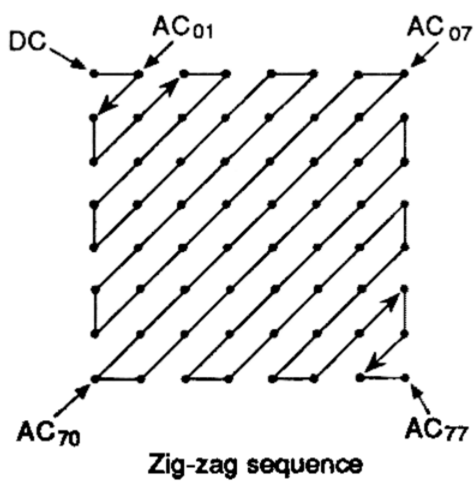

Figure 2: The zig-zag pattern of DCT coefficient selection, provided in [1].

3 Methodology

3.1 Overview

The following main functions were created for the purpose of performing these experiments.

compress_dct.m: Compress Using DCT Coefficients
compress_noiselet.m: Compress Using DCT + Noiselet Coefficients
decompress_dct_l2.m: Decompress DCT Using Least Squares
decompress_dct_tv.m: Decompress DCT Using TV Min.
decompress_noiselet.m: Decompress DCT + Noiselet Using TV Min.

The custom functions utilized the following utility and optimization functions provided by Justin Romberg and can be found in [4].

realnoiselet.c: perform noiselet transform in $n \log(n)$ time
cgsolve.m: solve symmetric positive definite problem using conjugate gradients
tvqc_logbarrier.m: solve quadratically constrained optimization problem
jpgzzind.m: produce a vector of indices for low-frequency DCT coefficients

To perform the experiment described in this project, the following three algorithms were compared. These algorithms were implemented using the functions listed above.

- DCT with Least Squares Recovery
- DCT with TV Minimization Recovery
- Noiselet Transform with TV Minimization Recovery

Now we will enumerate on each algorithm.

3.2 Evaluation Metric: Pseudo Signal-to-Noise Ratio

For the experiments included in this project, Pseudo Signal-to-Noise Ratio (PSNR) was used as a method to compare accuracy of images. This evaluation metric was given in [2]. For the purposes of this experimentation, this metric will take the form of equation (7).

$$PSNR(X, \bar{X}) = 20 \log_{10} \frac{255 * 256}{\|X - \bar{X}\|_{l_2}} \tag{7}$$

3.3 DCT with Least Squares Recovery

The DCT coefficient compression using least squares recovery will serve as a baseline for experimentation using other methods. While the theory has been derived to distinguish the recovery method from the TV Minimization, in actuality a compression-decompression cycle using DCT coefficients with Least Squares Recovery is similar to performing 2D filtering in the frequency domain.

Where as Figure 1 shows the original image in the spatial and DCT domains, Figure 3 shows the result of the coefficient removal in the spatial and DCT domains. Notice how the coefficients are removed towards the bottom-right of the DCT image, resulting in a low-pass filter effect. This also has the expected result of causing a ”ringing” effect on the image transform.

1. load the image into memory
2. perform DCT on image to get basis coefficients
3. remove basis coefficients corresponding to highest frequencies
4. perform inverse DCT on image to restore image
5. calculate PSNR to indicate performance

This process was used to generate Figure 3.

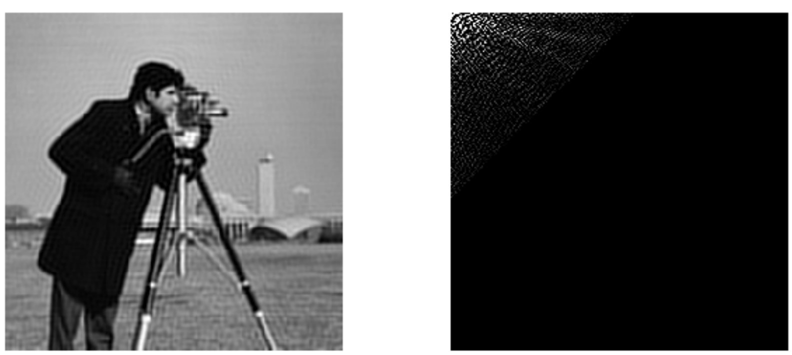

Figure 3: The removal of high-frequency components of the DCT image results in a slightly blurred image with ringing effects.

3.4 DCT with TV Minimization Recovery

The DCT transform using TV recovery uses the exact same compression algorithm given in section 5.2, but this method uses $l1$ minimization for estimation of the original image instead of the least squares method.

1. load the image into memory
2. perform DCT on image to get basis coefficients
3. remove basis coefficients corresponding to highest frequencies
4. perform inverse DCT on image to get optimization initial point
5. perform TV Min. to reach optimal recovery image based on $l1$ norm
6. calculate PSNR to indicate performance

The image recovery resulting from this process is given in Figure 4.

Figure 4: Recovery of DCT compression with 11000 coefficients using TV Min.

This image appears like a painted version of the Figure 3 recovery. It tends to have sharper corners than the $l2$ reconstruction and obtains typically a better SNR.

3.5 Noiselet Transform with TV Minimization Recovery

The final comparison procedure that was performed involved using a different compression algorithm that uses both DCT coefficients as well as random "noiselet" basis functions. The reconstruction of a system involving random noiselets is possible only by using TV Minimization. The procedure for using noiselet basis with TV min. recovery is given below.

1. load the image into memory
2. perform DCT on image to get DCT basis coefficients
3. remove DCT basis coefficients corresponding to highest frequencies
4. perform noiselet on image to get noiselet basis coefficients
5. extract random noiselet coefficients, keeping track of which ones were selected
6. perform least-squares reconstruction of image for optimization initial point
7. perform TV Min. to reach optimal recovery image based on $l1$ norm
8. calculate PSNR to indicate performance

The result of applying this procedure can be seen in Figure 5 during both the least-squares reconstruction for initial point used in $l1$ optimization and after $l1$ optimization.

Figure 5: Recovery of Noiselet + DCT compression using (a) $l2$ optimization and (b) TV Min. Note that (a) was used as an initial point in optimizing to reconstruct (b).

Notice how Figure 5(b) shows significantly more noise than Figure 4 does. Despite this observation, the reconstruction in 5(b) gives a much clearer picture of the details near the center of the image. This will ultimately result in a larger PSNR score for the noiselet transform than the pure DCT coefficients.

4 Results

For the three implemented algorithms, the following comparisons will be made. Note that 1 and 2 are performed in [2], and 3 is an analysis unique to this project.

1. appearance of images for different compression methods
2. PSNR performance for different compression methods
3. PSNR performance for different images

4.1 Image Appearance Comparison

While PSNR is a good quantitative way to analyze image quality, the appearance factor cannot be overlooked. This section compares the appearance of different algorithms using 6000 coefficients each. The noiselet transform algorithm used 1000 DCT coefficients and 5000 noiselet coefficients. The results of this comparison can be seen in Figure 6 and Figure 7.

Figure 6: The recovery of cameraman image using (a) DCT compression and least squares recovery, (b) DCT compression and TV min. recovery, and (c) Noiselet + DCT compression and TV min. recovery.

This test was also run on the popular ”man” image, as shown in Figure 7.

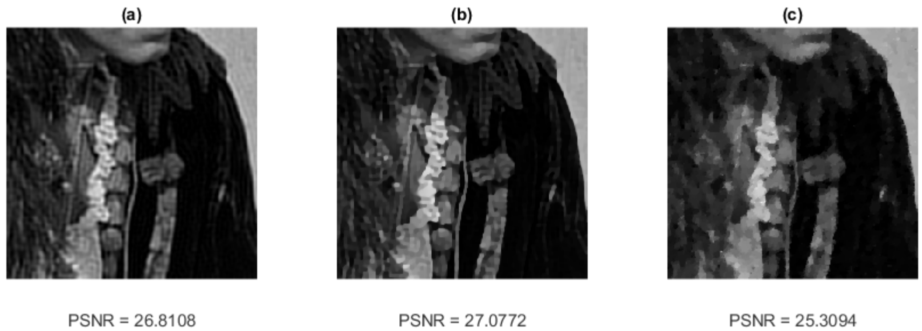

Figure 7: The recovery of man image using (a) DCT compression and least squares recovery, (b) DCT compression and TV min. recovery, and (c) Noiselet + DCT compression and TV min. recovery.

4.2 PSNR Performance for Different Compression Methods and Images

A comparison of PSNR Performance for different compression ratios was created. Note that the image is size $256 \times 256 = 65536$, and that gives the total number of DCT coefficients that can be created. As such, any encoding which contains fewer elements than that number can be considered a lossless compression, because the original image cannot be exactly recovered. Figure 8 and Figure 9 show the results for two different images. Note that the differences in the images are significant, suggesting that the Noiselet transform is less effective than DCT compression for some images.

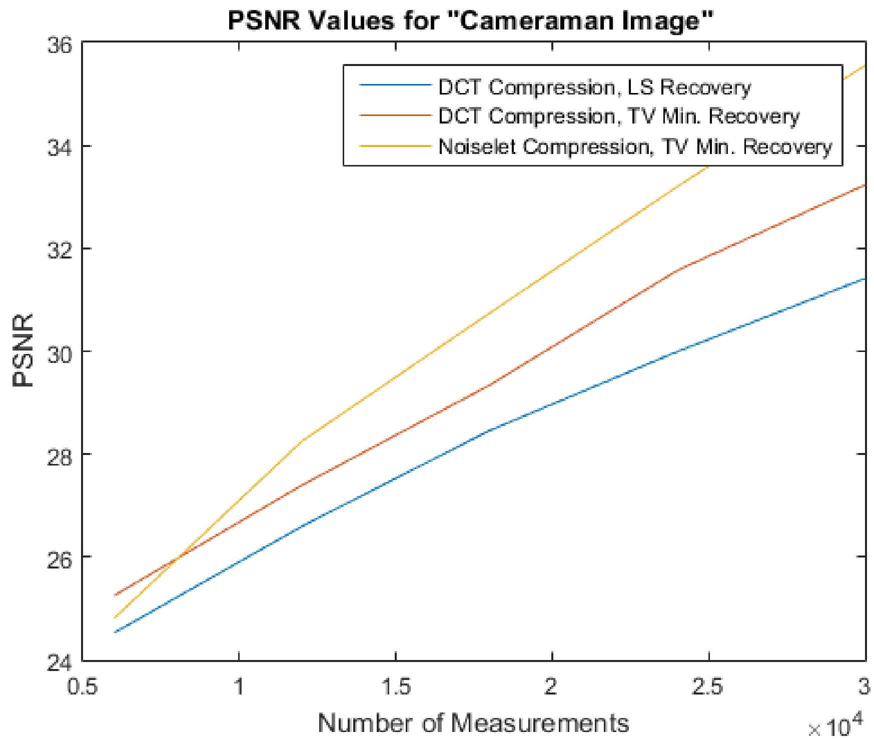

Figure 8: The PSNR metric for the "cameraman" image across different algorithms and coefficient numbers.

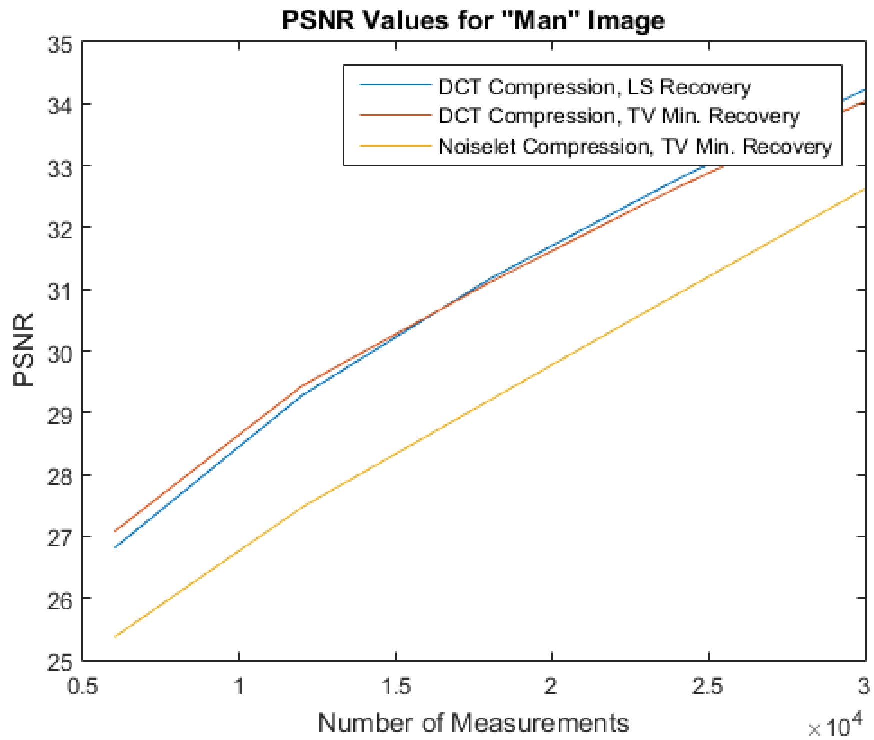

Figure 9: The PSNR metric for the "man" image across different algorithms and coefficient numbers.

5 Conclusions

Based on the qualitative and quantitative data provided in the results section as well as the experience gained from having implemented the procedures, a few conclusions can be made about these algorithms.

- TV Minimization is generally very slow. Could only practically be used when implemented on a GPU or some specialized hardware.
- While DCT compression produces "ringing" effects, the noiselet transform compression produces more random noise.
- In terms of performance, the DCT compression with TV min. recovery seems to perform the most consistently across images, and the performance is comparable with the noiselet transform at its best.
- For some images, the noiselet transform can obtain higher PSNR values for the same number of measurements, and it will also increase

faster with more measurements.

- Although the results in [2] were reproducible for the "Cameraman" image, similar performance characteristics do not seem to occur when the algorithm is applied on other images.

6 References

1. Wallace, G. K. (1992). The JPEG still picture compression standard. IEEE Transactions on Consumer Electronics, 38(1). doi:10.1109/30.125072

2. Romberg, J. (2008). Imaging via Compressive Sampling. IEEE Signal Processing Magazine, 25(2), 14–20. doi:10.1109/MSP.2007.914729

3. Coifman, R., Geshwind, F., & Meyer, Y. (2001). Noiselets. Applied and Computational Harmonic Analysis, 10(1), 27–44. doi:10.1006/acha.2000.0313

4. Rombert, J. (2008, January 1). Compressive Imaging Code. Retrieved April 29, 2015, from http://users.ece.gatech.edu/ justin/spmag/