COMPRESSED IMAGE RECOVERY

ABHISHEK SHARMA

Mechanical Engineering Department, University of Washington, Seattle, WA
`as711@uw.edu`

Abstract. In this work, we investigate the compressibility of a given image using discrete cosine transform (DCT). We also use the observation that the DCT transform of our image is sparse, to test an image recovery method from limited random observations of its pixel. We test image recovery for different number of random samples of pixels from the image. We then test the method for recovery of an unknown image from limited random measurements.

1. Introduction and Overview

We have Rene Magritte's "The Son of Man", shown in Fig. 1. We work with a grayscale and low resolution version of this image. We downsample the original image to 53×41. Firstly, we analyze the compressibility of image using Discrete Cosine Transform (DCT), by observing the values of the DCT coefficients of the image. We observe the images as represented by top 5, 10, 20 and 40 % of the DCT coefficients.

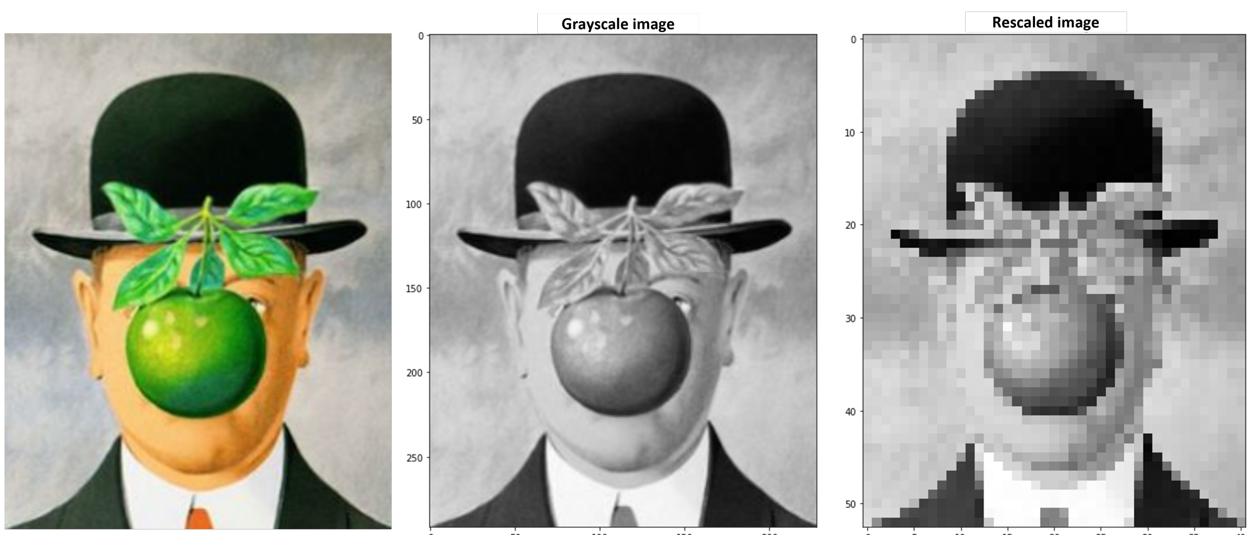

Figure 1. The original image (left), the grayscale version (center) and rescaled version (right).

In the second analysis, we use the observation of sparsity in the DCT domain to frame the image recovery problem as an optimization problem (discussed in Theoretical Background section). We perform image recovery for the following 3 sampling rates of image pixels: 1) 20%, 2) 40% 3) 60%. We perform 3 trials for each of these conditions.

Date: March 18, 2022.

Lastly, we use the algorithm top recover and unknown image from limited samples of its pixel.

2. Theoretical Background

Compressed Sensing is paradigm of data acquisition which asserts that natural data can be represented in an appropriate basis using far fewer degrees of freedom than in the domain they are usually acquired [1]. We observed an example of this in our report for assignment 1 entitled "Finding the Submarine". We observed that the original pressure amplitude signal from the submarine could be represented using a signal 3 dimensional spatial frequency vector, which is much less than the signal representation in the space-time domain. We observe something similar in the domain of natural images. Natural images mostly show continuity across pixels i.e. neighbouring pixels generally have similar values. Sharp jumps usually occur at the edges of the objects or when fine details are present like patterns or designs on clothes. Regardless, it is possible to represent images using far fewer variables than then number of pixels present in the image. Another factor that comes at play while representing images for human perceptual system is that human vision is far less sensitive to the color than intensity of the images (Grayscale), which is often sufficient to identify objects. These observations are used in lossy compression of images, for example in JPEG images [2].

Discrete Cosine Tranforms (DCT) [3] are effective at encoding images using a few coefficients. They work better than Fast Fourier Transform (FFT) at representing images using fewer coefficients. DCT represents a signal of finite length as a linear combination of cosine basis functions as shown in equation (1). The coefficients $DCT(f)_k$ of these basis function, then encode the information about the signal. A signal of length N can be encoded using N basis functions.

The output of the discrete cosine transform is a vector $DCT(f) \in R^N$, where $f \in R^N$ is the original signal.

$$DCT(f)_k = \sqrt{\frac{1}{N}}\left[f_0\, cos\left(\frac{\pi k}{2N}\right) \;+\; \sqrt{2}\sum_{j=1}^{N-1} f_j\, cos\left(\frac{\pi k(2j\,+\,1)}{2N}\right)\right] \tag{1}$$

The k^{th} entry ($DCT(f)_k$) is the inner product of the original signal and the k^{th} discrete basis function given by the vector $v \in R^N$ shown in eq. 2

$$v_k = a_k\, cos\left(\frac{\pi k(2j\,+\,1)}{2N}\right) \tag{2}$$

where $a_0 = \sqrt{\frac{1}{N}}$ and $a_k = \sqrt{\frac{2}{N}}$ for $k = 1,\, 2,\, ...\, N-1$. The vectors $[v_k^T]$ can be stored as the row of a matrix D and then $DCT(f) = D * f$. The corresponding inverse matrix (D^{-1}) operates on the vector $DCT(f)$ and reconstructs the original signal $f = D^{-1} * DCT(f)$.

Often with natural images only a few coefficients contain most of information about image i.e. most of coefficient values are negligible. Therefore, we can drop most of the coefficients and reconstruct a recognisable image from the remaining important coefficients. We use this principle to reconstruct compressed versions of the 53×41 image that we are working with.

For 2D signal like images, the discrete cosine transformation is performed by successively applying 1D DCT ((1)) to the rows and columns of the image. The corresponding transform matrix D_{2D} is given by kronecker product of the 1D matrices corresponding to the row D_y and column D_x transforms. $D_{2D} = D_y \otimes D_x$. Similarly, the inverse transform matrix is given by $D_{2D}^{-1} = D_y^{-1} \otimes D_x^{-1}$. The matrix D_{2D} operates on the flattened image $I_{vec} \in R^{mn}$ to generate the $DCT(I_{vec}) \in R^{mn}$ vector, where m and n are the number of rows and columns of the image respectively.

A direct consequence of being able to represent images using only a subset of the original degree of freedom, is *Compressed Image Recovery.* In other words, we can sample only a few random pixels of the image and reconstruct an approximation of the source image. The recovery process is as the following- For an image $Im \in R^{m\times n}$ and the corresponding vector $Im_{vec} \in R^N$ $(where\, N = mn)$, consider a vector $y \in R^M\,(M < N)$. The vector y is the measurement of random pixels of an image. Then y has a corresponding measurement matrix $B \in R^{M\times N}$. B is constructed by selecting the rows of an identity matrix $(R^{N\times N})$ corresponding to the pixel number of the image being observed Im_{vec}. Then, the original image can be reconstructed by solving the following optimization problem (See eq (3)).

$$\underset{x\in R^N}{\text{minimize}}\ ||x||_1 \qquad \text{subject to: } BD_{2D}^{-1}x = y \tag{3}$$

where the matrix corresponding to the $2D$ inverse DCT and y is the measurement vector of pixels. The minimizer x^* is the vector $DCT(Im^*_{vec})$ of the image $Im^*_{vec} = D_{2D}x^*$, which hopefully resembles the true image Im_{vec}.

The minimization of $||x||_1$ ensures that the vector x^* is sparse and the constraint ensures that x^* resembles the DCT vector of the true image.

3. Algorithm Implementation and Development

All the algorithms were implemented in Python [4]. Numpy package [5] was used for data array manipulations, the DCT and inverse DCT matrices were constructed using the *scipy.fftpack.dct* method from SciPy [6]. The optimization problem was solved using 'CVXOPT' solver from the the convex optimization package *cvxpy* [7]. All the visualizations and figures were generated using matplotlib [8].

4. Computational Results

Fig 2 shows the values of DCT coefficients for the image. We observe that most of the coefficients have a small magnitude, indicating sparsity. Thus, we can neglect most of the coefficients and still preserve most of the information.

Fig 3 shows the reconstructed image for different levels of compression. We observe that with only top 5% of all the coefficients, we can generate an identifiable image, albeit a blurry one. As we include more coefficients, the reconstructed images become sharper.

Fig 4 show the recovered images from random sampling of pixels. Even though we do not use any prior knowledge about the contents of the image (i.e. randomly sampled pixels), we are still able to reconstruct decent approximations of the original image. With only 20% of the sampled pixels we get the outline of the man in the picture. The apple is still hard to identify, even with 60% sampling rate but can we identified from the context of the image i.e. "Son of Man" is a popular work of art. This however indicates that details of the image are hard to recover if no prior information is available. The image recovered using 40% sparse sampling is much worse compared to the one reconstructed using top-40% of the DCT coefficients, as expected.

Finally, fig 5 shows the recovered unknown image from limited random samples of its pixels. The author could not identify the unknown image, which could be attributed to the author being too young or too old. The author is unsure about this.

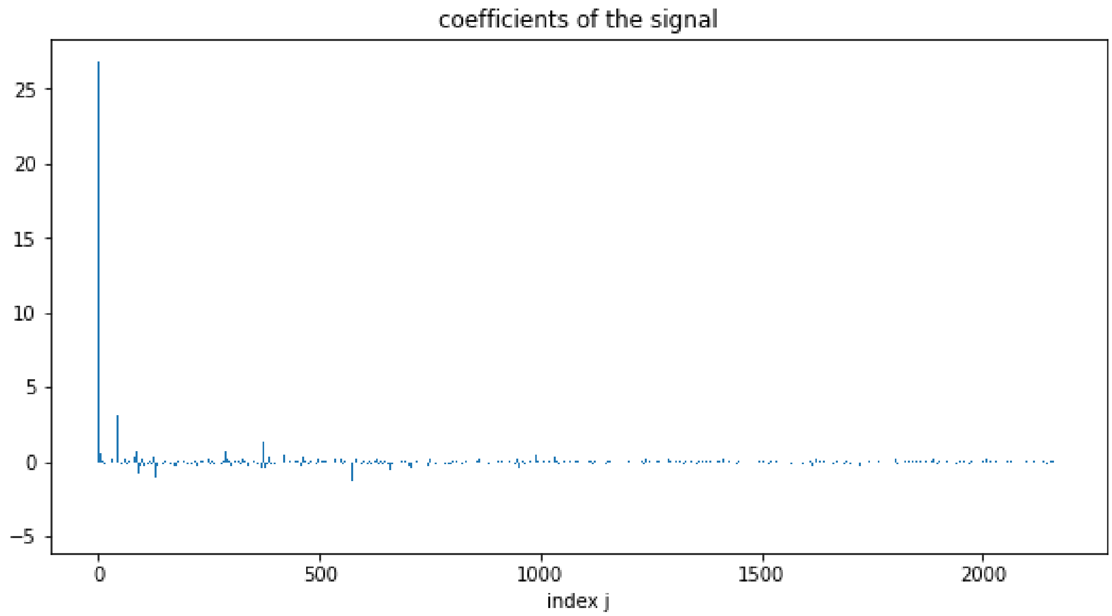

FIGURE 2. The DCT coefficients of "Son of Man"

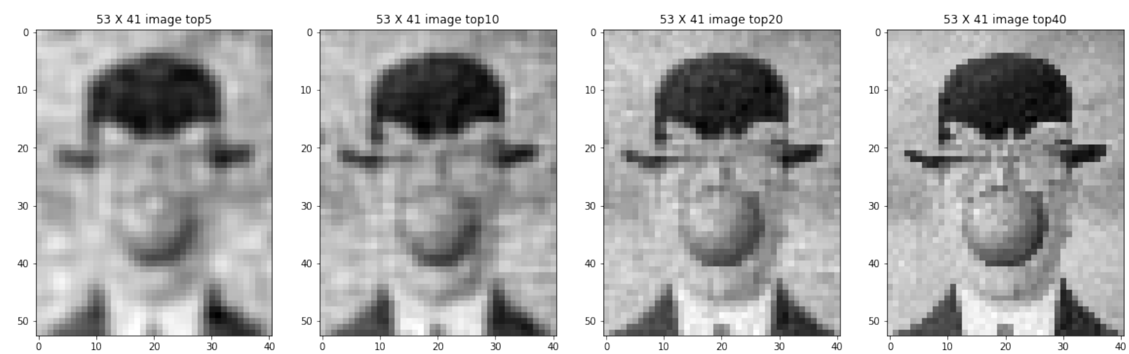

FIGURE 3. The reconstructed images (from left to right) for different levels of compression using top 5%, 10%, 20% and 40% of the DCT coefficients.

5. Summary and Conclusions

In this report we used Discrete Cosine Transform for compression of the "Son of Man" image. We then used this observation of sparsity in the DCT domain to formulate an optimization problem for recovery of image from few random samples of image pixels. Then, we tested this method for recovery of an unknown image.

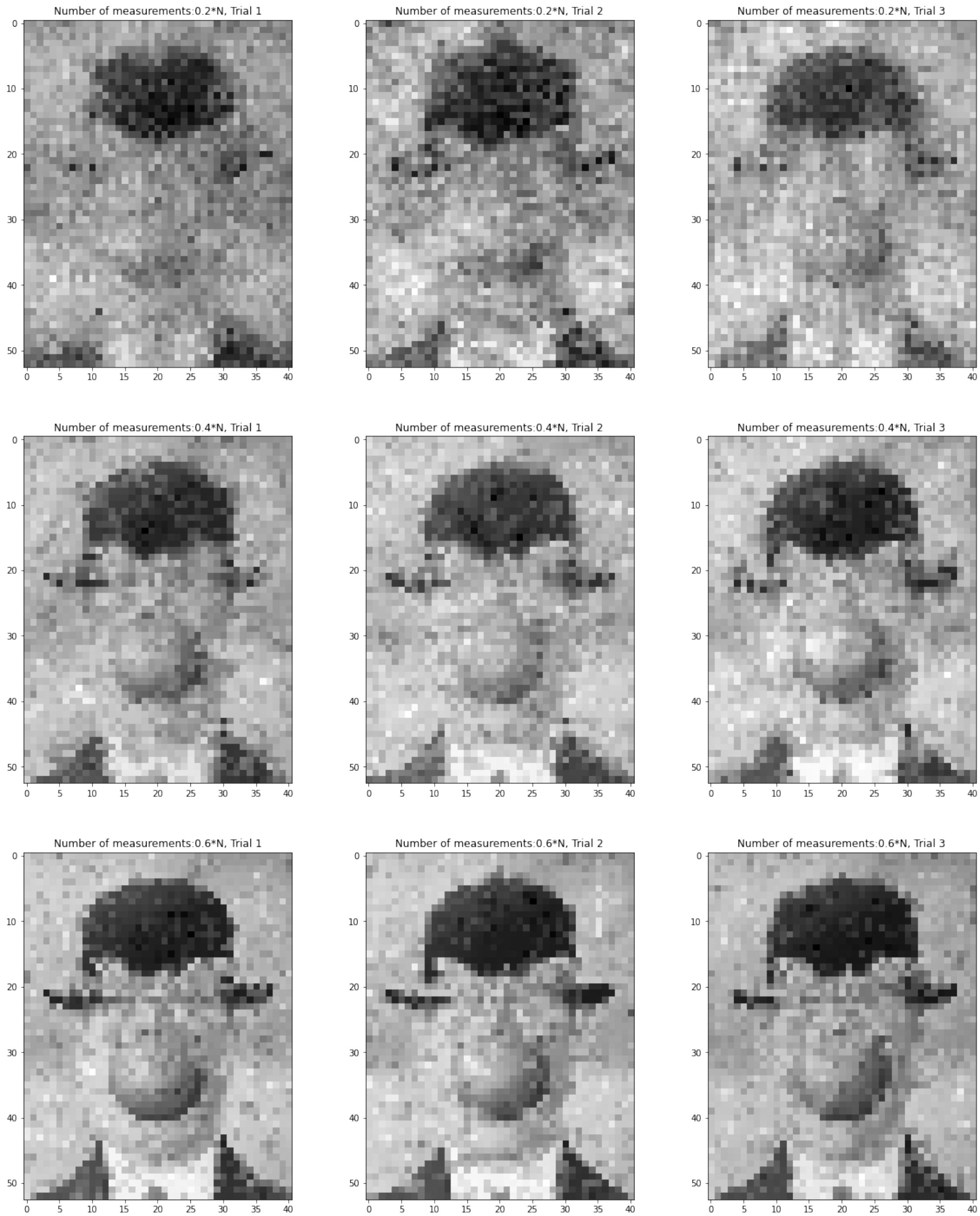

Figure 4. The recovered images for different sampling rates of random pixels. The rows correspond to different sampling rates and the columns are different trials at the same sampling rate. The top, center and bottom rows are constructed using random sampling of 20%, 40%, and 60% of the pixels respectively.

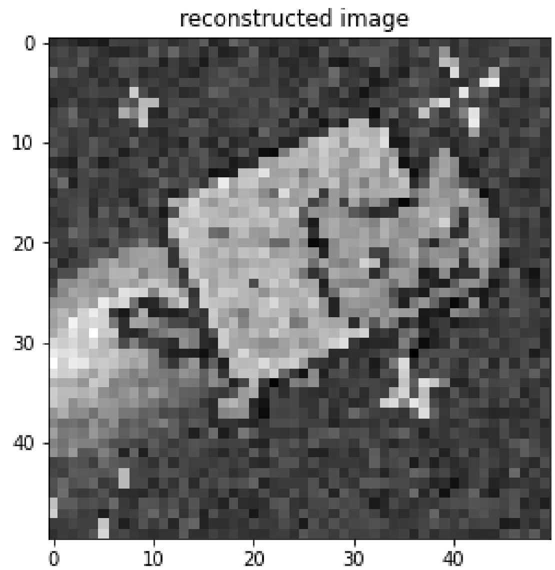

FIGURE 5. The unknown image. Seems like a cat with a square body, launched into space.

Acknowledgements

The author is thankful to Prof. Bamdad Hosseini for discussion and demonstrations related to compressed sensing and sparse recovery. We are also thankful to Katherine Owens for clarifying doubts regarding construction and use of DCT matrices. Furthermore, the discussion thread on discord was extremely helpful.

References

[1] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[2] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[3] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

[4] G. Van Rossum *et al.*, "Python programming language.," in *USENIX annual technical conference*, vol. 41, p. 36, 2007.

[5] T. E. Oliphant, *A guide to NumPy*, vol. 1. Trelgol Publishing USA, 2006.

[6] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[7] S. Diamond and S. Boyd, "Cvxpy: A python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.

[8] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 03, pp. 90–95, 2007.