The Discrete Cosine Transform and JPEG Compression

Andrey Pluzhnik
Department of Physics, University of California, Santa Barbara, CA 93106
(Dated: March 15, 2020)

Image processing is a field that has a wide spectrum of applications ranging from robotics to neuroscience. As it turns out, Fourier Transforms are a common tool in the arena of image compression. This paper will discuss the application of the Discrete Cosine Transform in conversion of raw images to the commonly used JPEG format.

I. INTRODUCTION

Images can be compressed in two main kinds of ways, lossless and lossy. Lossless image compression is a procedure such that an image is completely recoverable from the compression operation. Whereas lossy techniques achieve a higher compression ratio at the cost of losing some of the data of the original image. We will mainly focus on the JPEG(Joint Photographic Expert Group) algorithm which is a lossy compression technique. This paper will begin by first discussing the Fourier and Discrete Fourier Transforms and their applications to image compression. Then the paper demonstrate how a variant of the Discrete Fourier Transform called the Discrete Cosine Transform is utilized in JPEG compression. In short, the Fourier Transform is a process which transforms a function from a spatial domain, to a frequency domain. A Discrete Fourier Transform is an analogous operation, but discreetly over a set of samples instead of a continuum. Specifically, this paper will discuss the utilization of the Discrete Fourier Transform in image compression. For images, the advantage of such a process is that the image before and after being transformed effectively express the same information. In a frequency domain images can be manipulated in ways that are not as natural in the spatial domain.

II. METHODS AND MATERIALS

Simply put, an image can be considered as a signal. This means that it is useful to consider it in both the spatial domain or the frequency domain. The image spatial domain is the set of pixels as you see them, while the frequency domain describes the rate at which pixels vary in the spatial domain. It has been determined that, when an image is transformed to the frequency domain, the information describing high frequency components is considered less important to the human eye[1]. To better motivate this premise, consider a set of alternating white and black lines. One can impose a set of narrower white lines on each black line at a given frequency. The idea is that if the imposed white lines are narrow enough the image will appear as the original set of black and white lines; removing the imposed lines will not significantly change the image. This is exactly the basis for JPEG compression; through a precise choice of which high frequencies are discarded, one can effectively compress an image without compromising its visual integrity.
To first remove unncessary frequencies one must transform the data of an image to the frequency domain. This transformation is denoted by the Fourier Transform(FT). It is an integral which is of the form[2]

$$g(w) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-iwt} dt \tag{1}$$

while the inverse case is:

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(w) e^{iwt} dw \tag{2}$$

Here, $f(t)$ and $g(w)$ are functions that describe a signal in a spatial or frequency domain respectively. Because images are stored with discrete variables, typically given by row and column coordinates, it is useful to develop a version of the Fourier Transform for discrete functions. Consider a periodic function f(t) that can only be known over a discrete set of N samples $f = [f_0, f_1, ..., f_N]$ where each sample is separated by a sample time step T. The fourier transform of this function would be

$$g(w) = \int_{0}^{(N-1)T} f(t) e^{-iwt} dt = f[0]e^0 + f[T]e^{-iwt} + \\ ... + f[(N-1)T]e^{-iw(N-1)T} = \sum_{t=0}^{N-1} e^{-iwtT} f_t \tag{3}$$

Notice that the $\frac{1}{\sqrt{2\pi}}$ factor was omitted; for normalization purposes, a factor of $\sqrt{1/N}$ is used instead. The general operation described above is called the Discrete Fourier Transform(DFT). This operation treats a periodic function, so the frequency w can be rewritten as k multiples of the fundamental frequency $\frac{2\pi}{NT}$. Implementing the discussed changes gives the final form of the DFT [2]:

$$g(k) = \frac{1}{\sqrt{N}} \sum_{t=0}^{N-1} e^{-2\pi ikt/N} f_t \tag{4}$$

In an image, a pixel in a rectangular grid is described by two coordinates based on its row and column positions. Hence, a two dimensional discrete fourier transform is utilized to account for the two dimensional nature of storing pixels. For a square pixel grid, the transform will look like[3]:

$$g_{m,n} = \frac{1}{\sqrt{2N}} \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} e^{-2\pi i(xm+yn)/N} f_{x,y} \tag{5}$$

The conversion to a sum is important because this is the step that converts the image into the frequency domain where high frequency data can be thrown out without losing much image information. To reobtain the image in the spatial domain, the matrix containing all elements $g_{m,n}$ is passed into the inverse DFT.

A further improvement on the DFT is to consider splitting up the image in terms of a cosine basis instead of the typical sine and cosine basis. One may wonder why this step is an improvement, and furthermore, why the sine basis would not also be viable. The reason for this point is quite subtle, and an explanation is given in the discussion section. With that said, it is no surprise that the task of converting into a frequency domain of cosine functions is performed by the Discrete Cosine Transform(DCT); it utilizes the same concept of the DFT but is given in a slightly different form of [4]:

$$g_{m,n} = \frac{1}{\sqrt{2N}} C(m)C(n) \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} cos([2x+1]\frac{m\pi}{2N}) cos([2y+1]\frac{n\pi}{2N}) f_{x,y} \tag{6}$$

Where $C(i) = \frac{1}{\sqrt{2}}$ if $i = 0$ and $C(i) = 1$ if $i > 0$. This form explicitly expands a function in terms of cosines, in which the argument of the exponent in (5) is altered to account for boundary conditions.

III. RESULTS

The JPEG algorithm is the process which takes advantage of obtaining an image in a frequency domain. More importantly, the DCT aids JPEG compression in this endeavor. The JPEG algorithm consists of three main steps:

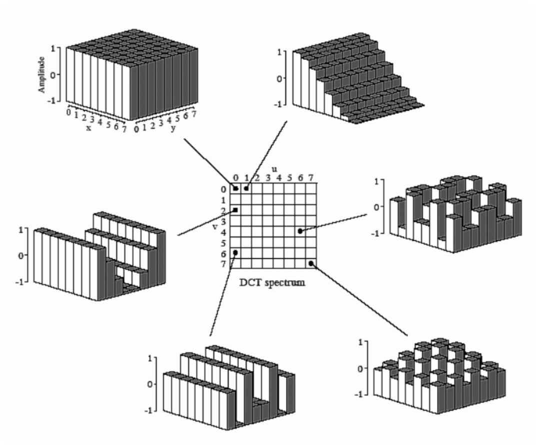


FIG. 1. This figure displays the spectrum of basis functions of the DCT. Each element of the shown array would be amplitude of the basis function in an 8x8 square of an image[6].

1. DCT is applied on a set of 8x8 blocks in an image
2. Certain frequencies are pruned in each block with a process called quantization
3. A reverse DCT is applied to retrieve the image.
4. Between steps 2. and 3. the image is compressed by encoding

In the first step, the DCT is applied to groups of 8x8 matrices of pixels(for purposes of simplicity, we will consider images of one color component) which it maps into the frequency domain. Because the sum is performed over an 8x8 block, N in (6) is simply 8, thence the value at a coordinate (m,n) is determined by equation below:

$$g_{m,n} = \frac{1}{4} C(m)C(n) \sum_{y=0}^{7} \sum_{x=0}^{7} cos([2x+1]\frac{m\pi}{16}) cos([2y+1]\frac{n\pi}{16}) f_{x,y} \tag{7}$$

It is important to note that at each coordinate (m,n) in the 8x8 block corresponds to an amplitude of a particular basis function. Furthermore, the sum can be rewritten with the help of a matrix T containing elements $T_{m,n} = \frac{1}{2} cos([2m+1]\frac{n\pi}{2N})$ if $m > 0$ and $T_{0,n} = \frac{1}{\sqrt{8}}$ if $m = 0$.

The T matrix would act on an 8x8 block F of an image in the form of $D = TFT^T$ where F is our 8x8 block in the spatial domain and D is our 8x8 block in the frequency domain. An 8x8 square block was chosen specifically for the reason that only square matrices are invertible. If a non-square matrix was chosen the inverse transform could not be performed. Once this process is complete we have a block which represents the amplitude of the basis functions in the image.

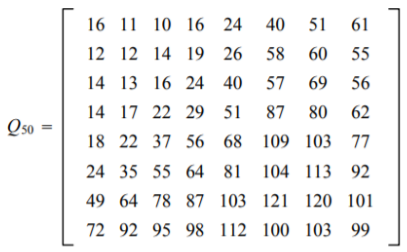
As an example, consider an image composed of horizontal black and white lines.

$$F = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \end{pmatrix} \quad (8)$$

The given matrix represents an 8x8 block of the image with all rows either describing a completely white or black line. Applying DCT on the matrix with $D = TFT^T$ yields:

$$D = TFT^T = \begin{pmatrix} 1019 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -183 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -216 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -324 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -924 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (9)$$

This array represents the amplitudes of the basis functions prevalent in the original 8x8 block, similar to those in Fig. 1. In this example the (0,0)(top left) element holds information about the lowest frequency basis function, while the (7,7)(bottom right) element describes the highest frequency. Evidently, because information in (8) is stored in a uniform manner, we should not expect non-zero elements in the lower right of the matrix where high frequency basis functions are held. Additionally, colors vary in the vertical direction, so it is sufficient to store the information of the image in terms of basis vectors of a fixed horizontal frequency. This is why the last seven columns have '0' elements. To confirm this intuition, the termination of these horizontal basis vectors can be demonstrated with a quick computation. In (8), all columns are identical so elements in the spatial do not depend on column, so $f_{x,y} \to f_x$.



$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

FIG. 2. This quantization matrix gives the values by which elements in array D are divided by. The subscript 50 represents the quality level. The quality level, which ranges from 0 to 100, balances compression with image quality[4].

Consequently (7) is modified to

$$g_{m,n} \sim \sum_{x=0}^{7} cos([2x+1])\frac{m\pi}{16})f_x \sum_{y=0}^{7} cos([2y+1]\frac{n\pi}{16}) \quad (10)$$

Simplifying further, the sum in y is a trigonometric telescoping series and resolves as[5]:

$$g_{m,n} \sim sin(\frac{n\pi}{2})cos(\frac{n\pi}{2}) \quad (11)$$

This expression is 0 for any non-zero integer value of n, thus demonstrating the termination of elements after the first column. Fortunately, the human eye is more perceptive to low frequency elements, and altering high frequency ones will minimally alter the look of the image. The quantization matrix takes advantage of this fact.

Step two is implemented by dividing each element of the amplitudes matrix by an element with the same coordinates in the quantization matrix. The seemingly random matrix, such as the one in Fig. 2, has been developed through different kinds of experimentation and testing. It is important to notice that typically the elements closer to the bottom right are divided by higher numbers than are the ones to the top left. This is to emphasize the high frequency pruning. After rounding each element, the higher frequencies are then 'pruned' which means their corresponding elements are rounded to 0. Once elements are 0, remulitplying by the quantization matrix elements does not change them from 0, hence the 'lossy' aspect of JPEG compression. To be precise, lossy nature of the JPEG compression is a result of this rounding done directly after dividing by quantization elements. For the most part, rounding does not affect image quality as higher frequencies disappear under rounding.

One can also conceive of versions of this matrix that either offer lower compression and higher

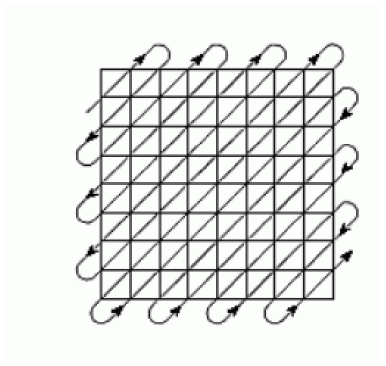


FIG. 3. This is the 'zig zag' pattern used to convert image in frequency domain to string of 64 values.[6] .

quality or higher quality and lower compression. Typically high compression matrices create more 0 elements while the opposite is true for low compression matrices.

Step three is implemented by reversing the implemented operations to retrieve the original image. Suppose that D is your starting matrix; first each element would be multiplied by its corresponding quantization matrix element, Q_{ij}. The next step is to perform the inverse cosine transform by converting $D = TFT^T$ to $F = T^TDT$. This is possible because T matrices of the DCT have a useful property called orthonormality meaning that $T^T = T^{-1}$. Because colors are denoted by integer values, each element is then rounded to the nearest integer and then added to 128.

Step four, the logical intermediate between steps two and three, does not involve the DCT but it is still vital for the size reduction of the image. In the so-called 'encoding' process, an 8x8 array is converted into a series of bits. Post quantization, F contains a wide spread of elements that are zero. To exploit this, a zig-zag(Fig. 3) pattern can be imposed over F to set the order of the storage of each pixel. Encoding converts the elements picked up by the zig-zag into a string of bits and then compresses long runs of repeating zeroes via the Huffman encoding algorithm.[6] Of course, the pattern does not have to be zig-zag; a zig-zag just happens to be optimal for commonly used quantization matrices. This final step is a lossless procedure and only reduces the image into a size that is reasonable for storage.

IV. DISCUSSION

To summarize, an image can be compressed by means of transforming it into a frequency space and removing details corresponding to high frequency components. The DCT is an exceptionally effective transform for this kind of lossy based compression. But, it turns out that the methodology by which compression is applied can be modified in a number of ways. As mentioned in Methods and Materials, the DFT or DST(Discrete Sine Transform) can be considered in the place of the DCT; however, the compression ratio for either of these transforms is much higher[8]. Thus, it is accepted that algorithms using DST and DFT are inferior for the purposes of compression. The reason for this claim is that choosing DCT over DFT(or DST) is necessary for higher compression quality. An important property of the DCT is that it is able to fit a higher proportion of coefficients in a small range of basis functions. This means that quantizations will sparsely prune the important frequencies which correspond to larger coefficients. For DFT and DST, however, the coefficients are spread more evenly which results in more widespread pruning during quantization. The difference in coefficient density can be explained by the boundary conditions implied by each transform. The DCT in particular introduces an even extension outside of the range of the transformed signal. The DFT, due to its periodicity outside of the range of the signal, introduces discontinuities at the signal boundaries. Qualitatively, this means more high frequency terms are necessary to smooth aberrations at the borders of signals under the DFT compared to

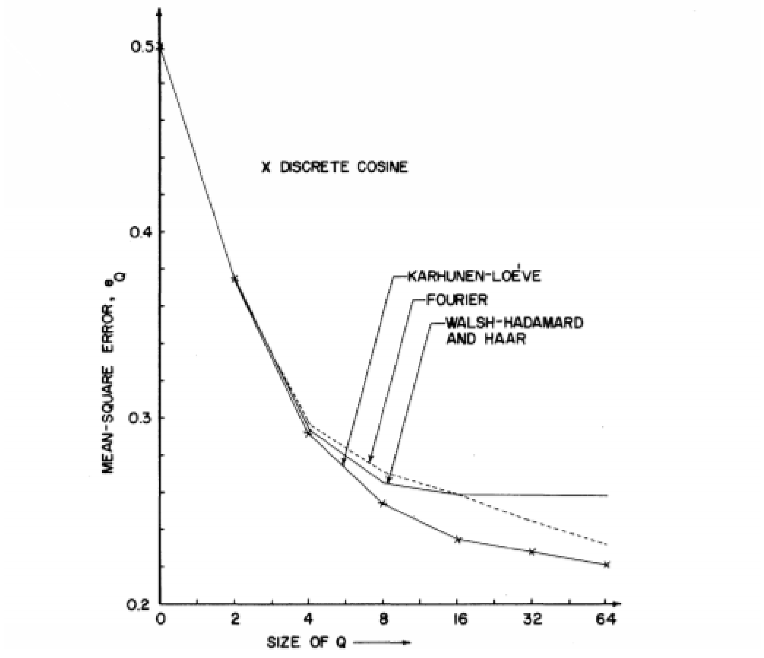


FIG. 4. Mean square error performance of various filters compared to the DCT for Wiener Filtering[7].

the DCT. [3]

Another alternative, namely the Karhunen-Loueve transform(KLT), offers the lowest possible lossy compression ratio. It is not used so widely due to being significantly harder to implement [3]. As you can see in Fig. 4, the performance of KLT and DCT are very close, further demonstrating that DCT is quite effective for lossy compression.

ACKNOWLEDGMENTS

I would like to thank Toby Chu and Carlos Ortiz for reviewing my paper, also for providing useful advice and pointers. I would also like to thank Mihir Shevgaonkar for useful discussion on the nature of Fourier Transforms.

[1] G.K. Wallace *The JPEG still picture ompression standard* IEEE Transactions on Consumer Electronics, vol. 38, no 1, pp. xviii-xxxiv, Feb 1992.
[2] Stephen Roberts *The Discrete Fourier Transform* Department of Engineering Science, Oxford http://www.robots.ox.ac.uk/ sjrob/Teaching/SP/l7.pdf
[3] A.K. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.
[4] S. Cabeen, P. Gent *Image Compression and the Discrete Cosine Transform.* College of the Redwoods, (2011).
[5] Samuel Greitzer *Many Cheerful Facts* Arbelos 4 (1986), no. 5, 14-17.
[6] Steven W. Smith. *The Scientist and Engineers Guide to Digital Signal Processing.* California Technical Pub., San Diego, Ca, 1999.
[7] N. Ahmed, T. Natarajan K. R. Rao. *Discrete Cosine Transform.* IEEE Transactions on Computers, vol C-23, no. 1, pp. 90-93, Jan 1974.
[8] Burger, W. Burge, M.J. *Digital Image Processing An Algorithmic Introduction Using Java.* Springer, London 2008