

Supplement to Chapter 2 of *The Science of Digital Media* – Digital Image Representation

Worksheet – Digital Imaging > The Discrete Cosine Transform and Frequency Components¹

Modeling environment: MATLAB

Exercise 1

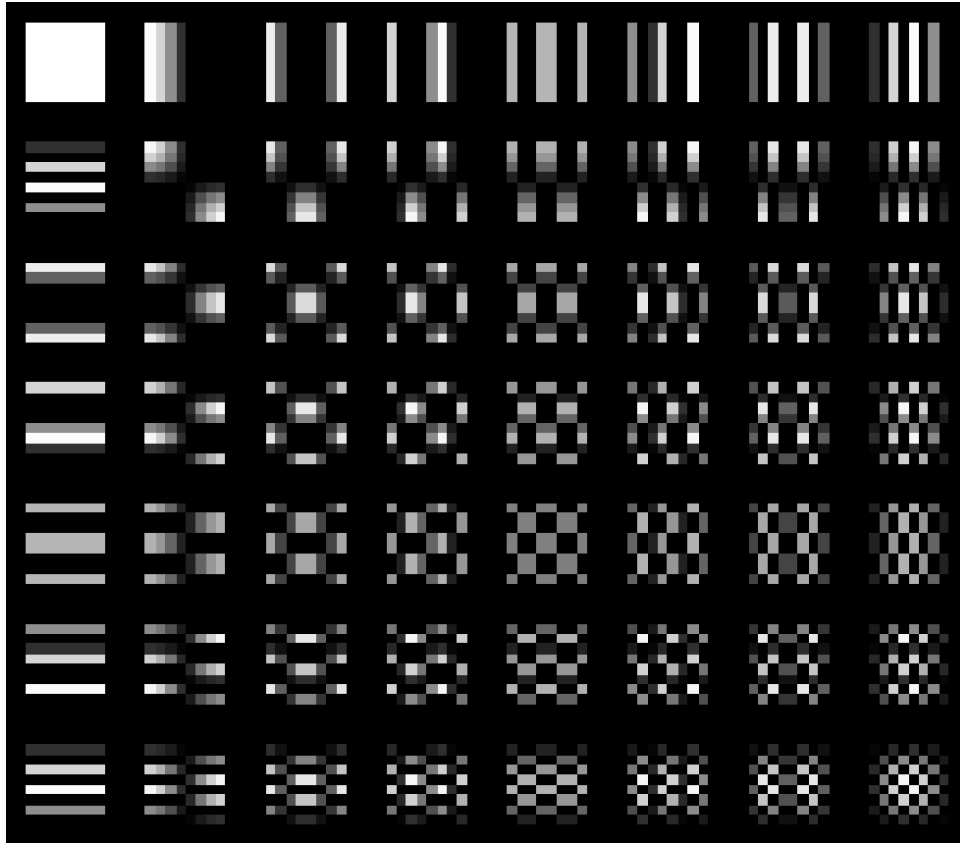


Figure 1

$$f(r, s) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \frac{2C(u)C(v)}{\sqrt{MN}} F(u, v) \cos\left(\frac{(2r+1)u\pi}{2M}\right) \cos\left(\frac{(2s+1)v\pi}{2N}\right)$$

where $C(\delta) = \frac{\sqrt{2}}{2}$ if $\delta = 0$ otherwise $C(\delta) = 1$

Equation 1

¹This material is based on work supported by the National Science Foundation under Grant No. DUE-0340969. This worksheet was written by Dr. Jennifer Burg (burg@wfu.edu).

For simplicity, we will use grayscale images where the pixel values range from 0 to 255. Extending the operations to RGB color is straightforward.

We'll begin with a very small, simple image so that you don't have too many data values to examine at once. You can use the image called *8x8checkerboard.bmp* supplied with this exercise, or create one of your own. It is just an 8 x 8 pixel image drawn as a black and white checkerboard, shown in an enlargement below.

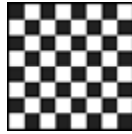


Figure 2

Read *8x8checkerboard.bmp* into MATLAB and store the values in a variable called *im*.

Display the image in a Figure window.

Graph the pixel values using the *bar3* command. (Note that in MATLAB, arrays and matrices are numbered from 1, while our equation for the DCT numbers from 0.)

Perform a DCT on the image data and store the results in a variable called **F**. (**F** is a two-dimensional array corresponding to *F* in Equation 1.)

Graph the *imdct* DCT values using *bar3*. Export the graph as a *.emf* file and paste it into your worksheet.

Exercise 2

The figure you created in the previous exercise represents how much of each frequency component there is in the image. The frequency components we're talking about are pictured in Figure 1.

Let's look at how these relate to the inverse DCT. Each term in the nested summation of Equation 1 has a product of cosines multiplied by the coefficient in position $F(u, v)$ of the DCT matrix. Consider the frequency pictured at position (1,1) in Figure 1. This is the waveform frequency associated with setting u to 1 and v to 1 in the cosine product, i.e.

$$\cos\left(\frac{(2r+1)\pi}{16}\right)\cos\left(\frac{(2s+1)\pi}{16}\right)$$

Since our image is 8 x 8 pixels, both M and N are 8. Graph this function in MATLAB.

How does the graph compare to the waveform pictured at position (1,1) of Figure 1?

What function would you graph to see the frequency associated with position (7,7) of Figure 1? Graph the function in MATLAB.

$$\cos\left(\frac{(2r+1)7\pi}{16}\right)\cos\left(\frac{(2s+1)7\pi}{16}\right)$$

You should be able to recognize this as the bar graph version of the image in $F(1,1)$.

Exercise 3

In the previous exercise, you took the DCT of *8x8checkerboard.bmp* and stored the result in F . Print out the values stored in F .

There are negative values in F . Explain why this makes sense. To do this, look at the frequency component corresponding to position (7,7) in Figure 1. Why does it make sense that you would have a negative amount of this component?

What kind of image would you need to have so that the DCT of the image would be identical to the DCT of *8x8checkerboard.bmp*, except that all AC coefficients would be multiplied by -1? (The AC coefficients are all the values in F except for $F(0,0)$. $F(0,0)$ is the DC component.) Create the image and run the DCT on it to verify your answer.

Exercise 4

How would you generate an image whose only frequency component is the one pictured in position (7,7) of Figure 1?

Try it and display it in MATLAB.

Exercise 5

The discrete cosine transform is used as one of the key steps in JPEG compression. The idea is that if you transform the image data to the frequency domain before you quantize it, then you can quantize the high-frequency components more coarsely than the low-frequency ones (because they're stored separately from each other). Quantizing more coarsely helps to reduce the amount of data stored for the image. However, when you quantize the high-frequency components by dividing them by relatively large

integers and rounding, you lose some of the precision – the detail. But this isn't a problem, because high-frequency components correspond to quickly-repeating patterns in the image – like the small checkerboard patterns we've been using for examples. The human eye doesn't pick up detail like that very well anyway, so it's a good place to apply compression.

Where are the high frequency components in Figure 1?

If you can analyze an image and determine that there aren't many high frequency patterns in the image, then you know it's a good candidate for JPEG compression.

One way to do this is to take the DCT of the image and see where the "energy" is located in the image – that is, where are the highest-amplitude frequency components located?

Try this with the image called *lucy.bmp* supplied with this exercise, or try it on an image of your own. We chose this image because of the horizontal stripes of the blinds, which will illustrate something in the frequency components.

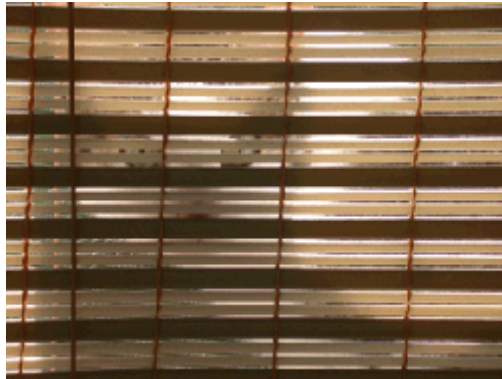


Figure 3

Take a DCT of the image with

```
im=imread('lucy.bmp');  
imdct = dct2(im);
```

Now look at the DCT values as if they made up an image themselves. You can do this with

```
imshow(imdct, [0, 255]);
```

Adding [0, 255] to the command scales the DCT data to values between 0 and 255 so that you can view this as a grayscale image.

Export the image file as a *.emf* file and paste it into your answer sheet. Analyze what you see in this picture.



Figure 4

Try the same thing on a large checkerboard. Read in the file *128x128checkerboard.bmp*. It's a 128 x 128 pixel checkerboard where the first row is white, black, white, black, etc. Do a DCT on the image and then look at the DCT data as an image. Export the image and paste it into your answer sheet.

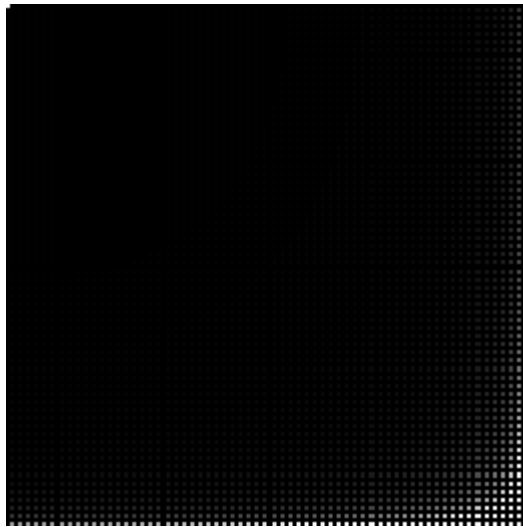


Figure 5

