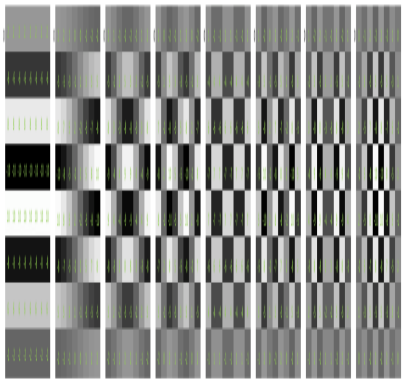


Visualising 2D Discrete Cosine Transforms

Description

Iâ??m interested in the Discrete Cosine Transform (DCT) method as a means of hiding one image inside another. Itâ??s also a key technique in image compression.

Iâ??ve implemented the DCT method on an Excel spreadsheet. The image below is an 8Ã?8 grid of pixel values, in grey ranging from 0 = black to 255 = white. You could think of this as a small fragment of a much larger image. The shades of grey in this demonstration are generated automatically by a formatting feature in Excel. Iâ??ll call this the â??target imageâ?• as I want to see if I can transform it to a series of coefficients using the the DCT method, which I can then transform back to this image.

34	35	70	106	143	143	107	71
35	59	106	154	179	178	154	107
71	107	179	202	226	227	227	178
106	131	178	155	179	202	251	227
106	107	131	82	106	155	226	100
71	58	59	35	83	154	202	179
34	59	82	82	106	155	179	130
34	59	106	131	179	178	155	83

The values on the grid squares could also be elevation information on a block of land, or any other mapped digital data. Researchers have used DCT to identify features of a landscape, to classify digital models, and in automated image recognition.

I used DCT to calculate how much of each of a set of 64 grey patterns is present in the target image. These 64 â??basis arraysâ?• are grey-scale patterns generated by a cosine formula at successively increasing frequencies from left to right and top to bottom. So the patchwork squares show possible gradations or undulations of 8Ã?8 pixel arrangements. This is analogous to the one-dimensional pixel

arrangement I showed in the previous post.

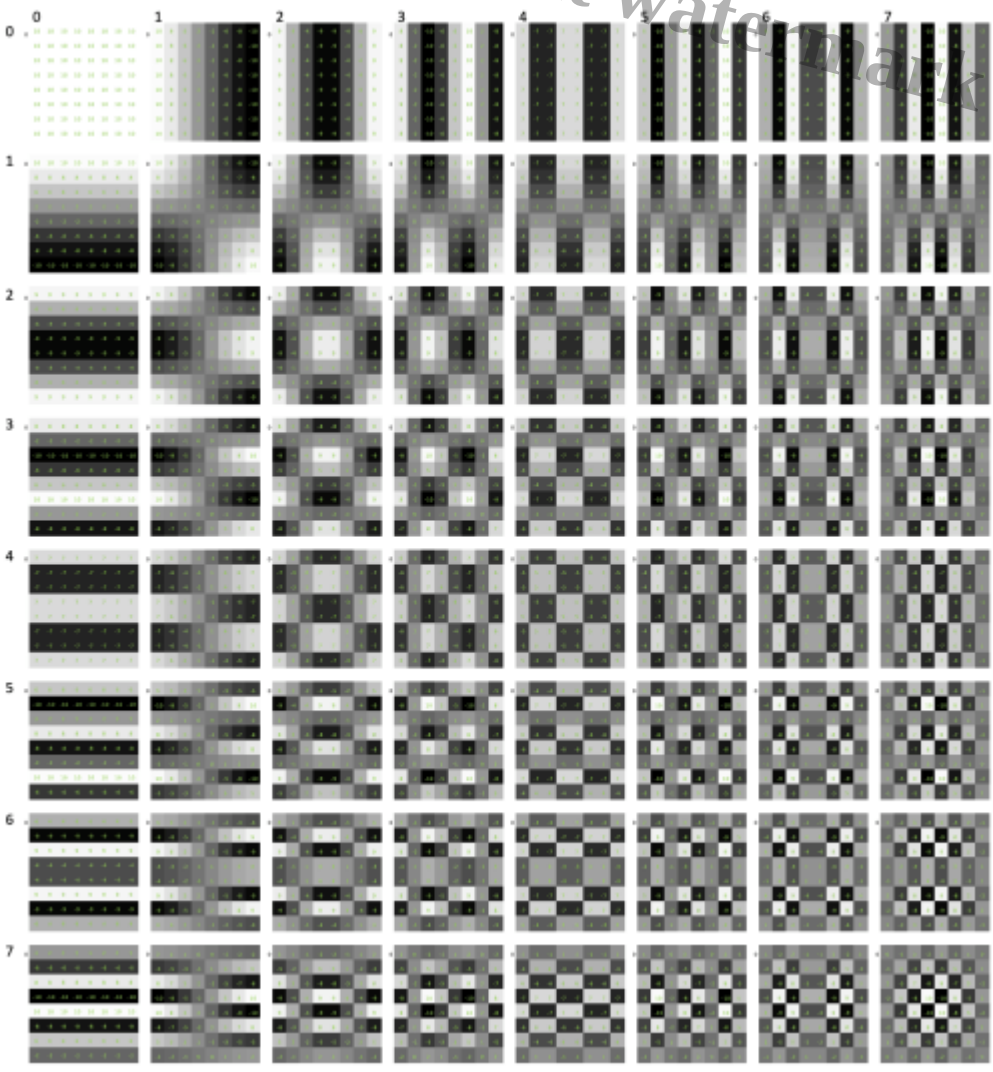
I generated this array of arrays for each cell in Excel where the formulas in the cells look looking something like this

=COS((2***$A16**+1)*PI()***$K$16**/16)*COS((2***$B$15**+1)*PI()***$M$15**/16)

The elements in bold are variables. **K16** and**M15** are the frequencies of undulations in the horizontal and vertical directions in an 8Ã?8 image; **$A16** and **$B$15** are the coordinates of each pixel in the image.

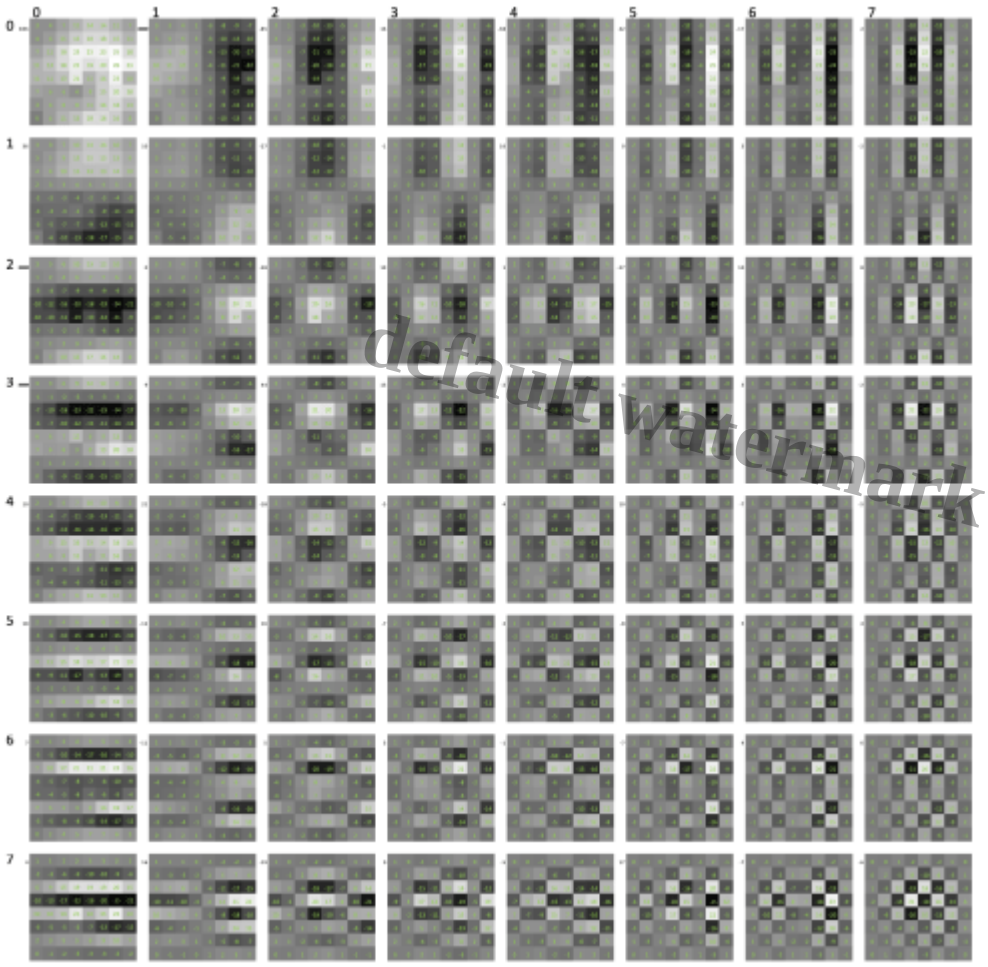
Visualising the basis functions

DCT experts call the 64 patterned squares â??basis functions.â?• The 8 columns and 8 rows of this basis function matrix are labelled 0-7. The values in the pixels range from -1 to +1 and the spreadsheet calculates these to several decimal places. I multiplied these by 10 in this diagram so the range is -10 to +10 and I rounded out the decimal places. That way the values could fit into the crowded cells of the spreadsheet.



The DCT procedure generates an array of multiplication factors (coefficients) that indicate the strength of each basis function in the target image. I calculated the coefficients by multiplying each of the basis function array pixel values by the corresponding value in the target array. The coefficient for each of the basis arrays is simply the sum (or average) of all those new values for each basis array.

Hereâ??s what the basis arrays look like as a result of these multiplications. Too small to see on this screen shot, the coefficient values are written next to each basis function matrix.



Here are the coefficients on a table. So the target 8Ã?8 array of pixels is converted to an 8Ã?8 array of coefficients, one for each basis function.

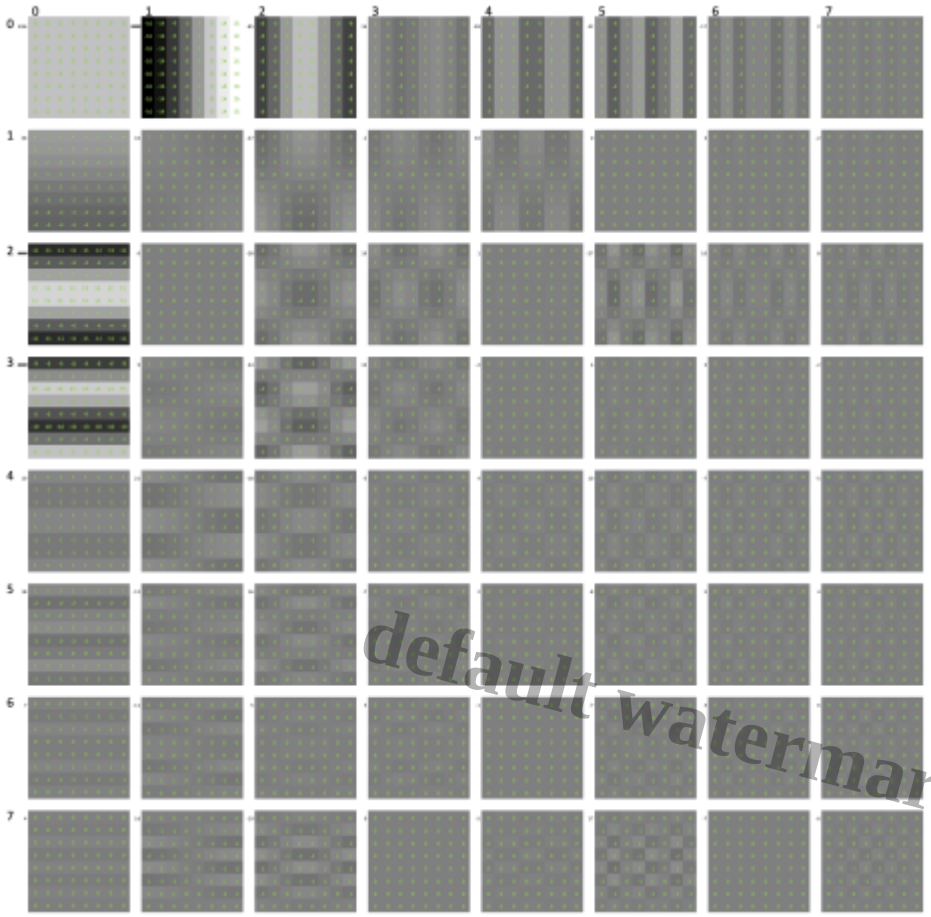
836	-164	-85	16	-38	42	-17	2
36	10	-27	-1	16	0	3	-2
-116	0	-28	19	1	-27	10	8
-100	9	44	13	-2	4	3	-2
10	21	-18	-5	4	10	-7	-5
18	-12	16	-7	2	-8	5	4
7	-11	5	5	-1	-7	4	6
4	16	-19	0	-5	17	-7	-6

These are not grey scale values, but show the strength of each of the basis functions in the target 8Ã?8 image.

Inverse DCT

From those 64 coefficients it should be possible to reconstruct an image close to the target image. As long as the algorithm for reconstructing the target image has access to the basis functions.

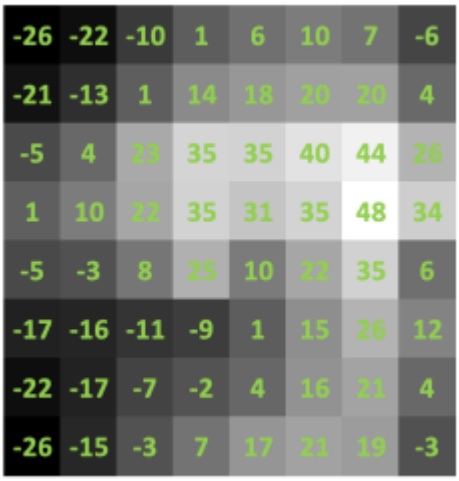
To test that I could reconstruct the target image I multiplied each of the coefficients by the values in the basis functions. That produced the following modifications to the arrays of the basis functions.



Overlaying each of these 64 ghostly images should then reconstruct the target. That is accomplished by adding the pixel values for each of these adjusted basis functions. The more pronounced the basis array, the stronger its contribution to the target image. Negative coefficients invert the effect of a basis array. Grey arrays have less effect on the reconstruction of the target image.

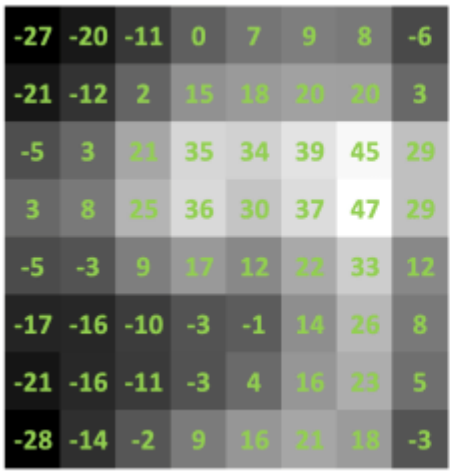
Re-constituting the target image

Sure enough, the process does re-generate a shaded image very close to the target. For simplicity, Iâ??ve left out the constants that bring the values back to positive numbers within the 0-255 colour space, but the relationships are correct. The Excel shading function generates the white to black grey range automatically, taking the lowest number as black and the highest number as white.

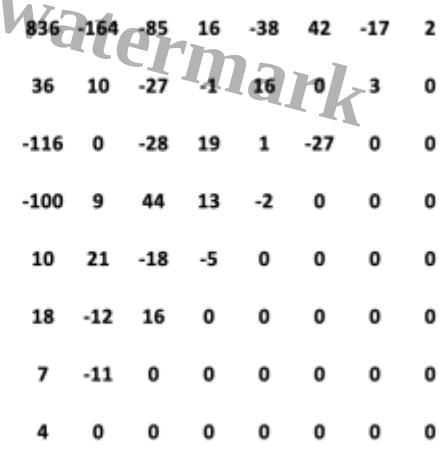


-26	-22	-10	1	6	10	7	-6
-21	-13	1	14	18	20	20	4
-5	4	23	35	35	40	44	26
1	10	22	35	31	35	48	34
-5	-3	8	25	10	22	35	6
-17	-16	-11	-9	1	15	26	12
-22	-17	-7	-2	4	16	21	4
-26	-15	-3	7	17	21	19	-3

Of note, here the higher frequency basis functions make little contribution to the reconstruction process as the values are so low. In the next calculation Iâ??ve eliminated the effect of the 28 coefficients in the higher frequency part (lower left) of the coefficient array by reducing them to zero.



-27	-20	-11	0	7	9	8	-6
-21	-12	2	15	18	20	20	3
-5	3	21	35	34	39	45	29
3	8	25	36	30	37	47	29
-5	-3	9	17	12	22	33	12
-17	-16	-10	-3	-1	14	26	8
-21	-16	-11	-3	4	16	23	5
-28	-14	-2	9	16	21	18	-3



836	-164	-85	16	-38	42	-17	2
36	10	-27	-1	16	0	3	0
-116	0	-28	19	1	-27	0	0
-100	9	44	13	-2	0	0	0
10	21	-18	-5	0	0	0	0
18	-12	16	0	0	0	0	0
7	-11	0	0	0	0	0	0
4	0	0	0	0	0	0	0

Thatâ??s the major part of the data compression process used in JPEG compression. The JPEG algorithm strings together all these coefficients for each 8Ã?8 parcel of the original image. It does this in a diagonal manner starting at the top left corner of each 8Ã?8 array. That produces a series of trailing zeros, and each list of 64 coefficients can be compressed using run length encoding, i.e. if there are 28 zeros in a row then indicate that in the file as 28 zeros rather than list all the zeros.

Hidden coefficients

I thought that DCT is a roundabout way of freeing up space in the file that contains the target image for other, hidden information, but a steganalysis algorithm (for detecting hidden content) would notice noise in the compressed JPEG file. Any data in the lower right sectors of the coefficient table would also get washed away by adjustments to the JPEG compression parameters.

According to a helpful online slide presentation by Ryan Gibson, the usual method of hiding information in the coefficients is to insert data in the least significant bits (LSBs) of the coefficients. It would leave any 0s or 1s untouched. Presumably the coefficients of another, hidden image could be concealed in

that way. There are many articles on improved steganography using DCT, and a plethora of methods.

Reference

- Gibson, Ryan. 2015. Steganography: Hiding Data In Plain Sight *Department of Computer Science, College of Arts and Sciences, The University of North Carolina at Chapel Hill*. Available online: http://www.cs.unc.edu/~lin/COMP089H/LEC/steganography.pdf (accessed 18 August 2020).

Category

1. Media

Tags

1. DCT
2. Discrete Cosine Transform
3. image compression
4. steganography

Date Created
August 22, 2020
Author
rcoyne99