

Media Processing SIV 864

Assignment 2

Nitish Raj

2018MCS2140

PART A

Q1. How does DCT perform in relation to its energy compaction for the images which are correlated and decorrelated (not correlated)?

Answer: If signal haven't the same power at all frequencies which implies there are correlation between the samples in the time domain. This represent redundancy of data and should be removed with loss of information. This signal is also termed as "not white". So DCT is for frequency based transformation, the more non white will lead to more energy compaction.

Q.2 In what ways the properties of separability and symmetry of DCT are useful?

Answer:

3 people clipped this slide

Clip slide

Separability:

- The principle advantage that $C(u, v)$ can be computed in two steps by successive 1-D operations on rows and columns of an image.

$$C(u, v) = \alpha(u) \alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos \left[\frac{\pi(2x+1)u}{2N} \right] \cos \left[\frac{\pi(2y+1)v}{2N} \right]$$

Figure 8. Computation of 2-D DCT using separability property.

12

9 of 17

3 people clipped this slide

Symmetry:

- A separable and symmetric transform can be expressed in the form

$$T = A f A,$$

where A is an $N \times N$ symmetric transformation matrix with entries $a(i, j)$ given by

$$a(i, j) = \alpha(j) \sum_{k=0}^{N-1} \cos \left[\frac{\pi(2j+1)k}{2N} \right],$$

and f is the $N \times N$ image matrix.- This is an extremely useful property since it implies that the transformation matrix can be pre computed offline and then applied to the image thereby providing orders of magnitude improvement in computation efficiency.

10 of 17

References: <https://www.slideshare.net/rashmikarkra/discrete-cosine-transform-47528535>

Q.3 Consider the Table 1, what does reduction of entropy indicate? For example, in case of Baboon it is shown that there is a drastic reduction of entropy.

Answer: The entropy reduction is huge. Entropy of original image shows high frequency and detail spatial information. Grayscale is uniformly distributed across the image and so coding is inefficient when done in spatial domain.

PART B

Image References : https://www.google.com/search?q=jfif&rlz=1C1CHBF_enIN824IN824&sxsrf=ACYBGNTI2yKg3AXRQ-Gbp5TSJ9nL13cwbQ:1573016732145&source=lnms&tbn=isch&sa=X&ved=0ahUKEwirha7859TIAhXJdCsKHa3JB50Q_AUIEigB&biw=1536&bih=722#imgrc=jITsaf-KzJxQKM

Method References: <https://www.programcreek.com/python/example/107277/scipy.fftpack.dct>

The image shows a screenshot of a Python script in Notepad++ and its execution results in a plot window. The script, titled "Assignment 2 PART B SUBMITTED BY NITISH RAJ 2018MCS2140", performs a Discrete Cosine Transform (DCT) on a grayscale image of a child's face. It uses NumPy, SciPy, and Matplotlib. The script calculates the DCT of the image, applies a threshold, and then reconstructs the image using the inverse DCT. It also calculates the Mean Squared Error (MSE) between the original and reconstructed images. The plot window, titled "Assignment 2 PART B 2018MCS2140", displays three vertically stacked images: the original grayscale image, the DCT image, and the reconstructed image. The x-axis ranges from 0 to 600, and the y-axis ranges from 0 to 1200. The status bar at the bottom indicates the file is a Python file, with a length of 2,130 characters and 48 lines. The window title bar shows "Windows (CR LF)", "UTF-8", and "INS". The system tray at the bottom right shows the date and time as "06-11-2019 10:34".

```
1 # ASSIGNMENT 2 PART B SUBMITTED BY NITISH RAJ 2018MCS2140
2 import numpy as np
3 import scipy as scp
4 import cv2
5 import matplotlib.pyplot as plt
6 from scipy import r as rfc
7 from scipy import fftpack as fft_scp
8
9 original_image=cv2.imread('TestSubject.jpg')
10 original_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2GRAY)
11
12 bitsize = 16
13 size_image = original_image.shape
14 dct = np.zeros(size_image)
15 for i in rfc[:size_image[0]:bitsize]:
16     for j in rfc[:size_image[1]:bitsize]:
17         dct[i:(i+bitsize),j:(j+bitsize)] =fft_scp.dct( original_image[i:(i+bitsize),j:(j+bitsize)], axis=0, norm='ortho' ), axis=1, norm='ortho' )
18
19 threshold = 0.031
20 dct_thresh = dct * (abs(dct) > (threshold*np.max(dct)))
21 percent_nonzeros = np.sum( dct_thresh != 0.0 ) / (size_image[0]*size_image[1]*1.0)
22
23 dct_image = np.zeros(size_image)
24 for i in rfc[:size_image[0]:bitsize]:
25     for j in rfc[:size_image[1]:bitsize]:
26         dct_image[i:(i+bitsize),j:(j+bitsize)] = fft_scp.idct( fft_scp.idct( dct_thresh[i:(i+bitsize),j:(j+bitsize)], axis=0, norm='ortho' ), axis=1, norm='ortho' )
27
28 dft = np.zeros(size_image,dtype='complex')
29 dft_image = np.zeros(size_image,dtype='complex')
30
31 for i in rfc[:size_image[0]:bitsize]:
32     for j in rfc[:size_image[1]:bitsize]:
33         dft[i:(i+bitsize),j:(j+bitsize)] = npy.fft.fft2( original_image[i:(i+bitsize),j:(j+bitsize)] )
34
35 threshold = 0.031
36 dft_thresh = dft * (abs(dft) > (threshold*np.max(abs(dft))))
37
38 for i in rfc[:size_image[0]:bitsize]:
39     for j in rfc[:size_image[1]:bitsize]:
40         dft_image[i:(i+bitsize),j:(j+bitsize)] = npy.fft.ifft2( dft_thresh[i:(i+bitsize),j:(j+bitsize)] )
41
42 percent_nonzeros_dft = np.sum( dft_thresh != 0.0 ) / (size_image[0]*size_image[1]*1.0)
43 mse = npy.square(original_image - dct_image).mean(axis=None)
44 mse2 = npy.square(original_image - dft_image.astype('uint8')).mean(axis=None)
45 print(mse,mse2)
46 plt.figure("Assignment 2 PART B 2018MCS2140")
47 plt.imshow( npy.vstack( (abs(original_image.astype('uint8')), abs(dct_image.astype('uint8')), abs(dft_image.astype('uint8')) ) ), cmap='gray')
48 plt.show()
```