Image Processing Practical for Chapter 3

Department of Bioengineering

Version 15-02-2015

DISCLAIMER: This is the second generation of a practical for Chapter 3. Debugging of this practical might still be expected. Your feedback is vital and welcome.

Begin this session by reviewing the Practical on image display and colourmaps; this will be useful to interpret the results of transform spaces. Also ensure that you have completed the practical of Chapter 2 (the peas on desk one), which is needed for part of this practical.

For the exercises below, save the commands in `.m` files where appropriate.

1 Image Transforms and Basis Images

Read the section of notes related to basis images, and carefully go though the exercise involving the transformation of the 2×2 image $\mathbf{F}$ into the transform domain $\mathbf{V}$. Ensure that you understand the operation of the *matrix inner product* and how it relates to the exercises in Chapter 3 on Page 51.

- *On paper*, ensure that you can go from the image to the transform domain and back again. Using Equations (3.16)/(3.18) and Equation (3.22).
- Try the following calculation:

  $$\langle \mathbf{V}, \mathbf{B}_{m,n} \rangle$$

  for $m = 0, 1$, $n = 0, 1$. What is the result that you get ? Can you write an equation similar to 3.18 that expresses this operation ? What does it mean (write it in words!)
- The basis images that we are using for the exercise on page 51 have simple patterns, consisting of +1 and -1 (up to a scaling of 1/2, which can be applied once, at the end of a calculation). Assuming that these numbers *could* be arbitrary floating point numbers, what is the number of arithmetic operations (additions and multiplications) needed to transform an image $\mathbf{F}$ into its transform domain representation $\mathbf{V}$? Can you figure out roughly how the number of operations increases as the size of a square image, containing N rows and $M = N$ columns, increases ?

2 The 2D-DFT

For this exercise, you will be using the "head.128" image previously used in Practical 1 (on colourmaps). Load this image into *Matlab* again, and ensure that you can display it with a grey-scale colour map.

- Using the two-dimensional Fourier transform of *Matlab*, transform head image, which we shall refer to as `x`, into the two-dimensional discrete Fourier domain. You will have to convert the image to double precision to do this. Write down your observations about the size of the 2D DFT representation as a *Matlab* variable
- Display the *absolute* values of the 2D DFT result. Can you see where the significant coefficients are ? Explore the effect of the `fftshift` command.
- Ensure that you can apply the *inverse* 2D-DFT to the discrete Fourier space already calculated to yield a *recovered* image `xdash`. Note that the result will be complex-valued on older versions of *Matlab*. If you get a complex-valued result, examine the values in the imaginary part of the array `xdash`. They should be very close to 0. Newer versions of *Matlab* should return a real-valued result on taking the inverse DFT of a forward DFT of a real signal: the correct result. If you modify the values in transform space, you *may* get a complex-valued result on taking the inverse.

- Calculate the *difference image* between the original `x` matrix and the reconstructed `xdash`. Find the sum-of-squares of the error image (SSE). You may express this as a percentage SSE relative to the original pixel values.

2.1 Visualising the basis images

We will now explore the effect of modifying coefficients in 2D DFT space. BUT first, we need to understand the implications of each coefficient. In the following little exercise, you will work on how to derive the basis images behind the transform, and therefore learn what each coefficient in discrete Fourier space represents. You will NOT be using the head image for this, but will be working from an empty 2D Fourier space, and placing values directly into that space yourself. Then, you will apply the *inverse* 2D-DFT to come back into the image domain.

- Generate an empty 16×16 array using the `zeros` command. This matrix will be treated as an empty 2D discrete Fourier space. Now, set one of the elements (say, $row = 5$, $col = 10$) to 1, and apply the *inverse* 2D-DFT. Display the real and imaginary parts of the result as an image (use `imagesc` to do this). The pair of images that you get (real and imaginary) represent the corresponding basis image for that coefficient.
- Confirm that the basis images of the 2D-DFT are orthogonal to each other.

2.2 Energy Compaction Property

We will now look at the effect of gradually copying the coefficients from the 2D-DFT of the head image into an empty matrix. This will be done by starting from the biggest coefficient and running to the smaller ones. So, you should put aside your basis images now, and go back to the head image, and it's associated discrete Fourier space.

- Explore the use of the `sort` command in *Matlab*. Sort the coefficients of the 2D-DFT into *decreasing* order of magnitude. You will need to pay careful attention to the syntax of the `sort` command. I suggest you try it out first on a small 3×3 matrix to understand how it works !
- Copy the coefficients of top 10 *magnitude* 2D-DFT space of the head image into an (initially) empty matrix, which is denoted `Ydash`, representing a new 2D-DFT space. Apply the inverse 2D-DFT to `Ydash` to generate an approximation `ydash` to the original image `x`. What do you see ? Add the next top 10 components, and repeat. You will need to be sorting coefficients according to magnitude, but copy the complex values into the new matrix, so be careful! You may wish to take coefficients in chunks of 100 elements, rather than in groups of 10 elements at a time. Explore !
- Create an animation in which the head image is gradually refined according to the top magnitude coefficients; you may find that the `pause` command is useful to display this properly.
- For each group of 10 (or 100) coefficients, compute the error in reconstruction between the recovered image `ydash` and the original image, `x`. How does the error change with iteration (i.e every 10s or 100s of coefficients in order to decreasing magnitudes) ? Plot a graph.

3 The Haar Wavelet Transform

Read carefully the help pages on the *Matlab* commands `dwt2` and `idwt2`.

- With the help of the `meshgrid` command, create an image of a solid white circle against a dark background. You can do this by a boolean command, but need to create a spatial coordinate system first. This is where `meshgrid` comes in.....
- Once you are happy with your image, apply a 2D Haar wavelet transform using something like:
  `dwt2(x,'haar','mode','sym')`
  where `x` represents your test image.
- Visualise the *channels* of the Haar wavelet transform. Note that these channels correspond with nothing more than a sorting of the basis image into certain pattern types. What is the nature of the pattern sorting ?
- Try to visualise the basis images of the Haar transform.

4 Learning Outcomes

What you have done today is to (I hope!) connect aspects of Chapter 3 with the equivalent commands in *Matlab*. I also hope that this material serves to demystify some of the mechanics of image transforms.