EE 40471 Project 3: Image/Multirate Processing

Due 28 April, 2017

Introduction: Multi-dimensional data is a fairly simple extension of what we've been doing in DSP. The most common data type is digital imagery, which is ubiquitous in all our experience. We'll try out a few of the things we've learned on pictures here. In Matlab, the command "imread()" will be most useful for image inputs, and "imwrite()" for output. For display, try "imshow()" or "imagesc()". Each digital image is read into, or written from, Matlab as a two-dimensional array. Many of the images you'll read in will be 8-bit unsigned integers, ranging from 0 to 255. As soon as you operate on them arithmetically, though, they are likely to be converted to double precision floating point. Keep this in mind when you write out your results. Under the Project 3 header on the website, you will find several monochrome, tagged image format (TIFF or TIF) digital images with which you may experiment. Feel free to use your own images for these exercises if you like.

1. Write a Matlab program to quantize your image values to arbitrary numbers of bits per pixel (do not use canned functions such as "quant()"). Use your program to quantize your favorite gray-scale image to 4 and then 2 bits per pixel and comment on the effects. Use quantization to the nearest level, not via truncation. This is a way of compressing the data, since it can be stored with fewer bits. Why is this a really bad way to do image data compression?

2. Finding edges is one of the most fundamental tasks in image analysis. Edges are usually defined as places where intensity changes rapidly as we move spatially. These locations of high gradients can be found though differentiation or its discrete approximation, differencing. Either a first- or second-difference operator is useful. A common approximation to a two-dimensional second derivative is the Laplacian operator, a filter whose coefficients are shown below:

0.08	0.17	0.08
0.17	−1	0.17
0.08	0.17	0.08

   Create an array of this form, and use the conv2() command in Matlab to filter your image to highlight its edges. Do the same with your coarsely quantized images and comment on the differences. Note that even if the original image is strictly positive, after this filtering, it will be zero-mean and you will need to contend with the negative numbers in such a way as to display them meaningfully.

3. As in the previous exercise, two-dimensional (2D) filters are often relatively simple generalizations of their one-dimensional (1D) counterparts. If we have a 1D filter we like, we may create a 2D "separable" version of it as follows. Let $\mathbf{h}_a$ and $\mathbf{h}_b$ be vectors representing the coefficients of 1D FIR filters, the first for manipulation in the vertical (y) direction and the other for the horizontal. Then let $\mathbf{h}_{2d} = \mathbf{h}_a \, \mathbf{h}_b^t$, the outer product of the two 1D filters (it could be that $\mathbf{h}_a = \mathbf{h}_b$). Use this format to create a 2D FIR highpass filter, and use it to enhance the high frequencies (particularly edges) in the

"barbara" image while maintaining response of 1.0 at zero frequency. Similarly to 1D, the (0,0) frequency output is just the sum of FIR filter coefficients. Submit a plot of the impulse response and frequency response of your 1D filter and a .tif file of your enhanced image.

4. Back in Chapter 4, we learned a bit about sampling rate changes, and multirate signal processing in general. We will not worry about polyphase implementation, but Section 4.7.6 may be most useful for us here. This is "sub-band" decomposition of a signal, essentially taking it apart into low- and high-frequency portions. You may recall from homeworks and examples that if we have bandwidth of no more than $\pi/2$ for our 1D signal, either high-pass or low-pass, we can downsample it by a factor of two and still recover the original by subsequent upsampling plus filtering that matches the original passband.

   In order to make this work as we've pictured it in exercises, the filters h_0 and h_1 in Figure 4.44 would have to be ideal lowpass and highpass, respectively. So if we're designing these two, you might immediately picture something like our windowing-based FIR design with filter of length 200 to minimize aliasing. However, using very sharp cutoff filters in sub-band decomposition is frequently counter-productive; for one thing, we know those sharp cutoffs in frequency give us big ripples in the time domain. As discussed in the text, if we co-design both our two "analysis" filters h_0 and h_1 along with our "synthesis" filters g_0 and g_1, we can erase aliasing in the final, reassembled signal, and achieve error-free reconstruction, with $y[n] = x[n]$ in Figure 4.44.

   If you look at equations (4.111) to (4.115) in the text, you'll see the conditions for aliasing cancellation and perfect reconstruction involve concepts that by now are old friends: the two analysis filters are related by our standard low-to-high transformation, the synthesis filter g_0 is equal to h_0 with scaling, and g_1 is a negative scalar times the high-frequency analysis filter. So the design of a single filter sets all four in Figure 4.44.

   We will use the very simple filter given in (4.116a) as our protoype. Sketch or plot all four filters' impulse responses, $h_0[n]$, $h_1[n]$, $g_0[n]$, and $g_1[n]$. Prove that these filters satisfy the aliasing cancellation property in (4.112) and also give us the perfect reconstruction condition of (4.115). Clearly, though, there is a *lot* of aliasing going on!

5. Now we will apply these concepts in 2D, on the "barbara" image (or suitable substitute - we can talk about this). In 2D, there are two frequency bands in each direction, meaning that we'll have four bands here. The first will be low frequency in the vertical direction and the horizontal, the second will be low frequency in the vertical but high frequency in the horizontal, and so on. For each sub-band, we need a filter akin to h_0 or h_1 in Figure 4.44, but now they will be h_{00}, h_{01}, h_{10} and h_{11}. Create these four filters and their g_{00}, etc., counterparts as in Section 3 and write their coefficients in 2×2 matrices (you may do this by hand).

   It's time to disassemble the image with your analysis filters. Convolve each analysis filter (conv2(), saving boundaries created) with the image and view each of the outputs; do they make sense in light of their approximate frequency content? Now downsample

each of them by factor 2 in each dimension, so each will be 1/4 as large. Place the four images into a single, larger image in the order of the indeces of the corresponding analysis filters, and save that clustered image as part of your report. These four images now contain essentially the same number of samples as before in the whole image, and each contains a single sub-band of frequency content. When images are transmitted, often the 00 band is transmitted first, and the refinement of detail is added as bandwidth permits. Try upsampling the 00 sub-band and then filtering by g_{00} and save the result. How does the quality compare with the original?

6. Let's try re-assembling the original image. Upsample (2×2) each of the sub-band images and pass it through its respective synthesis filter. Then add the four sub-band images, and trim off the borders. (The time-shift property you computed for (4.115) should tell you how much the image has been displaced.) Now, compute the maximum absolute error between the original image and your reconstruction. It should be a very small number, approaching machine precision. Submit the reconstruction as part of your report.