Creative Computing II

Image Filtering

9th February 2011

This lab sheet covers the use of *Octave* for image reading, writing and filtering.

1. This part covers reading an image file, and converting a colour image to grayscale.

   (a) using `imread`, read in a colour image; ideally one with many different colours; be sure to save the image data to a variable (not called `i`!) and to suppress printing the data.

   (b) call the *Octave* function `size` on your image data. Check that the pixel resolution is what you expect.

   (c) use `imshow` to display your image.

   (d) one way of converting an image to grayscale is to average the red, green and blue channels for each pixel. Use the expression on p. 112 of Volume 1, Chapter 5 of the subject guide to evaluate this; display the result with `imshow`.

   (e) the sRGB primaries used in digital images and displays are not equally luminous, and the sRGB values are not linearly scaled, so computing the mean is not the correct way to compute the equivalent grayscale value. Instead, the weighted average of the linear C_l values (Volume 2, Chapter 1) should be used: weighted by the amount each channel contributes to Y (the luminance in CIE XYZ coordinates), and then gamma corrected:

$$C_{\text{gray}} = 1.055 \left(\begin{pmatrix} 0.2126 & 0.7152 & 0.0722 \end{pmatrix} \begin{pmatrix} \left(\frac{C_r+0.055}{1.055}\right)^{2.4} \\ \left(\frac{C_g+0.055}{1.055}\right)^{2.4} \\ \left(\frac{C_b+0.055}{1.055}\right)^{2.4} \end{pmatrix} \right)^{1/2.4} - 0.055$$

   Implement this transformation, and display the resulting grayscale image with `imshow`.

   (f) visualize the difference between the two methods of producing grayscale images.

   (g) save your two grayscale images using `imwrite`, and load them into an image processing program. Can you produce a visualization of the difference between the images? What about in *Processing*?

2. Just as with audio, an image filter kernel can be applied using convolution: in two dimensions, we use the `conv2` operator.

   (a) Taking your grayscale image from part 1, experiment with applying echo, blur and edge detection filters using `conv2`.

   (b) Verify that you get the same results by using the Fourier method (with `fft2` and `ifft2`) of applying filters.

Other Resources:

- *Images and Pixels* tutorial (from D. Shiffman, *Learning Processing*, Morgan Kaufmann, 2008). `http://processing.org/learning/pixels/`
- *GNU Image Manipulation Program User Manual*, Chapter 8.