Department of Electrical & Computer Engineering
McGill University

ECSE-490 DSP Laboratory
Experiment 3

Image Processing

3.1 Purpose

In this experiment, you will become familiar with digital processing of two-dimensional signals – in particular, images. You will explore the 2-D DFT, as well as 2-D filtering, and the interpretation of the 2-D frequency response.

3.2 Background

A digital image usually represents a real-world image sampled on a rectangular spatial grid. Each sampling location is called a *pixel*, and is usually represented by its 2-D discrete spatial coordinates, say n_1 and n_2. The image content itself is represented by the value of the image signal at each pixel location $x[n_1, n_2]$. Different representations can be chosen for the actual values of the signal $x[n_1, n_2]$. In general, three components are sufficient to represent a colour image signal, so that each sample of $x[n_1, n_2]$ is a 3-dimensional vector, with each component representing the amount of red, green or blue colour in the image. For gray scale images, each sample $x[n_1, n_2]$ is just a scalar, and its value encodes the luminance at the given pixel location. In this experiment, we will concentrate on 8-bit monochrome images, where the possible levels are 0 (black) to 255 (white).

For the study of 2-D signals, one can also resort to the a frequency analysis, much like the Discrete-Time Fourier Transform. In this case, we have a 2-D DTFT (or DSFT, Discrete *Space* Fourier Transform) defined as

$$X(\omega_1, \omega_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x[n_1, n_2] e^{-j(\omega_1 n_1 + \omega_2 n_2)}.$$

Note that we now have two frequency variables, one vertical and one horizontal.

As in the case of 1-D signals, 2-D linear time-invariant (LTI) systems can be defined, and they have analogous properties. In particular, one can define 2-D convolution with a 2-D filter as

$$y[n_1, n_2] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h[k_1, k_2] x[n_1 - k_1, n_2 - k_2],$$

where $h[n_1, n_2]$ is the 2-D impulse response of the filter.

3.3 Preparation:

- Derive symmetry properties for the 2-D DTFT of a real signal (i.e. compute $X(\pm\omega_1, \pm\omega_2)$ as function of $X(\omega_1, \omega_2)$). Prove that $X(0, 0)$ must be real.
- Using the definition of 2-D DTFT, compute the impulse response of an ideal (rectangular frequency response) lowpass filter defined in the region $(-\pi, \pi] \times (-\pi, \pi]$ by

$$H_{LP}(\omega_1, \omega_2) = \begin{cases} 1, & |\omega_1| \leq \omega_{c,1} \text{ and } |\omega_2| \leq \omega_{c,2}, \\ 0, & \text{otherwise.} \end{cases}$$

  One could also define an isotropic low pass filter as

$$H_i(\omega_1, \omega_2) = \begin{cases} 1, & \sqrt{\omega_1^2 + \omega_2^2} \leq \omega_c, \\ 0, & \text{otherwise.} \end{cases}$$

  How would you compute the impulse response of $H_i(\omega_1, \omega_2)$?

3.4 Useful MATLAB commands

MATLAB has several useful commands to deal with images. In particular, the Image Processing ToolBox contains a number of functions for image manipulation.

In MATLAB, images will be represented as matrices. You must be careful when translating matrix indices from MATLAB (i.e. starting at 1 and from first row down) to spatial coordinates. In order to read an image file into MATLAB, use the `imread` command. Notice that if you read an 8-bit file, MATLAB will convert it to a matrix of `uint8` (8-bit unsigned) elements. You can convert this matrix to double-precision floating point representation with the `double` command.

In order to display an image, several possibilities are offered. We advise you to use the command `imagesc`, which adapts the colour range of the figure to the dynamic range of the image to display. The function `imagesc` should be used in conjunction with `colorbar` and `axis('image')`. You might also find it useful to set the colour-map to gray-scale by using `colormap gray`.

MATLAB contains an implementation of the 2-D DFT as `fft2`. Also have a look at `fftshift`.

3.5 Experiment

You will use two different files for this experiment: `checkerboard.tiff` contains an artificial image and `claire.tiff` contains a natural image.

3.5.1 2-D DTFT

- Using the artificial image `checkerboard`, compute its 2-D DTFT, and display it. Explain what you see and justify it. What happens if you use more points in your 2-D FFT?
- Now visualize the 2-D DTFT of the `claire.tiff` image. Characterize the frequency content of the image. Another way to view the local, spatial variations of the frequency content is to use a 2-D equivalent of the short-time Fourier-Transform (STFT). In order to do so, write a MATLAB program to subdivide am image into square blocks, and compute the 2-D DTFT of each sub-block. Explain how you can display the information gathered through this technique. Also devise a method to reconstruct the original image from the DTFT's of the individual blocks. Try blocks of size 16×16, 8×8, and 4×4 pixels.
- Generate an image whose pixels are samples of a white Gaussian noise process. Display its DTFT.
- Compute the 2-D DTFT of `claire.tiff`. Replace the phase of the DTFT by a random value (while still fulfilling the symmetry conditions of the phase of the DTFT for a real signal), and reconstruct an image by inverse DTFT. Repeat the operation with a random magnitude. Compare the two reconstructed images. What do you conclude?

3.5.2 Noise removal

In this part of the experiment, you will construct a noisy version of the image `claire.tiff` and try to diminish the effect of the noise using two different approaches.

- Build a corrupted version of `claire.tiff` by adding to it a random noise (zero mean and uncorrelated pixel-to-pixel) with standard deviation $\sigma = 20$.
- In order to remove noise, you will use the STFT implemented above to suppress all frequency components in different frequency bands. Assess the efficiency of your noise removal technique by computing the SNR before and after noise removal. Comment on the visual improvement and possible problems with this approach.
- A second denoising technique is to use a lowpass filter. Using the expression for the impulse response derived in the preparation, use implement a rectangular windowing technique to design an FIR lowpass filter. Choose the cut-off frequencies so as to match the signal and noise characteristics. Apply your filter (you may use the MATLAB function `conv2` to implement the filtering) to reduce the noise. Compute the SNR for different choices of the cut-off frequency and window size. Also comment on the visual reconstruction quality.

3.5.3 Edge Detection

A simple application of 2-D filtering is the problem of edge detection in images. A very simple way to detect edges is to filter the image with a suitable linear filter, then pass the absolute value of filtered image through a one-bit quantizer to threshold the filtered image,

$$y[n_1, n_2] = \begin{cases} 1 & x[n_1, n_1] > T, \\ 0 & \text{otherwise.} \end{cases}$$

where T is a suitable threshold. An example of a filter used for edge detection is the Sobel filter. You can access its 2-D impulse response in MATLAB by the command `fspecial('sobel')`. This filter is a 3×3 FIR filter that approximates a vertical gradient operator. Taking its transpose will approximate a horizontal gradient.

- Considering a simple 1-D model of an edge as a unit step, explain how filtering by an approximation of a gradient followed by thresholding can lead to edge detection.
- Use the Sobel filter to create a binary image representing the location of vertical edges in `claire.tiff`. In order to do so, you must choose a value for the detection threshold T. Explain the effects of changing T. Repeat the same operation, this time detecting horizontal edges.

3.6 Report

Include in your report all your results, as well as comments on the visual quality of images observed.