Digital Image Processing Laboratory: 2-D Random Processes

February 5, 2021

Introduction

This laboratory explores the use of 2-D random process models for images. You may implement your programs in Python. Make sure that all plots have accurate and clearly labeled axes and have titles that indicate what is being plotted.

In some of the following exercises, you will be asked to display images in the 8-bit range of 0 to 255. To do this in Python, perform the following steps.

- **Activate Anaconda Environment** - Use the file *environment.yml* to create and activate a Anaconda environment with the following commands:

```
conda env create -f environment.yml
conda activate ECE637
```

  If you have already created the environment, you can just use the second command.

- **Reading Images** - Read an image file, *img.tif*, into the Python by pillow using the following commands:

```
from PIL import Image
im = Image.open('img.tif')
```

- **Displaying Images** - Import Image Data into Numpy array. Display the image using plt.imshow(). Set 'cmap=plt.cm.gray' to display in gray.

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array(im)
plt.imshow(x, cmap=plt.cm.gray)
```

If you are producing an electronic version of your report, it is usually best to export 8-bit images directly to a file using *save*, as opposed to exporting through a figure window. For example, to export the image matrix x (with assumed range [0,255]), use the command

```
img_out = Image.fromarray(x)
img_out.save('img_out.tif')
```

Other output, such as mesh plots, can be exported through the figure window menu.

Questions or comments concerning this laboratory should be directed to Prof. Charles A. Bouman, School of Electrical and Computer Engineering, Purdue University, West Lafayette IN 47907; (765) 494-0340; bouman@ecn.purdue.edu

1 Power Spectral Density of an Image

In this problem, you will use Python to read and analyze the gray scale image *img04g.tif*. If you are unfamiliar with Python, please refer to the Usage of Anaconda listed on the main web page for this laboratory. Follow steps in that instruction to create a conda environment.

1. Download the Python py-file *SpecAnal.py* and the gray scale image *img04g.tif*. The m-file estimates the power spectral density by computing the logarithm of the normalized energy spectrum over a 64×64 window of the image. The comment lines in *SpecAnal.py* explain how the py-file operates.

2. Run *SpecAnal.py*. The py-file will display the image *img04g.tif* and show a mesh plot of the estimated log power spectral density. Export the plot for your report.

3. Run *SpecAnal.py* for 128×128, and 256×256 block sizes. Notice the power spectrum estimates remain noisy even when the block size is increased. Export the two mesh plots for your report.

4. Write a Python function, *def BetterSpecAnal(x)*, which returns a better estimate of the power spectral density of the 2-D array `x`. Your new py-file should:

   - Use 25 non-overlapping image windows of size 64×64. These windows should be selected from the center of `x`.
   - Multiply each 64×64 window by a 2-D separable Hamming window. You can create the 2-D Hamming window as the outer product of 1-D windows:
     `W = np.outer(np.hamming(64), np.hamming(64));` .
   - Compute the squared DFT magnitude for each window.
   - Average this power spectral density across the 25 windows.
   - Display a mesh plot of the log of the estimated power spectral density.

5. Use *BetterSpecAnal(x)* to compute the power spectral density estimate of *img04g.tif*, and export the mesh plot for your report.

Section 1 Report:
Hand in:

1. The gray scale image *img04g.tif*.
2. The power spectral density plots for block sizes of 64×64, 128×128, and 256×256.
3. The improved power spectral density estimate.
4. Your code for *BetterSpecAnal(x) function*.

2 Power Spectral Density of a 2-D AR Process

In this problem, you will generate a synthetic 2-D autoregressive (AR) process using Python, and analyze its power spectral density.

1. Use the Python function [numpy.random.uniform]. to generate a 512×512 image, x, with independent random numbers each uniformly distributed on the interval $[-0.5, 0.5]$. Display the image `x_scaled=255*(x+0.5)` using the plt.imshow command, as described in the introduction, and export the result for your report.

   Notice: Make sure the numpy array is uint8 before writing to file.

2. Filter the image x to produce the image y using an IIR filter with transfer function

$$H(z_1, z_2) = \frac{3}{1 - 0.99z_1^{-1} - 0.99z_2^{-1} + 0.9801z_1^{-1}z_2^{-1}} \ .$$

   Hint: Find the corresponding difference equation, and use that to implement the filter.

3. Display the image $y + 127$, and export the result for your report.

4. Theoretically calculate $S_y(e^{j\mu}, e^{j\nu})$, the power spectral density of y. Plot the magnitude of S_y using *mesh*, and export the result.

5. Use *BetterSpecAnal(y)*, your Python function from the previous exercise, to estimate the power spectral density of y. Plot the estimated power spectral density and export the result.

Section 2 Report:
Hand in:

1. The image $255 * (x + 0.5)$.
2. The image $y + 127$.
3. A mesh plot of the function $\log S_y(e^{j\mu}, e^{j\nu})$.
4. A mesh plot of the log of the estimated power spectral density of y using *BetterSpecAnal(y)*.