CS 105 – Introduction to Scientific Computing

Assignment 3 – Manipulating Matrices

Objectives

After completing this assignment you should be able to:

1. Determine value(s) in vectors and matrices given indices
2. Determine the output of operations on vectors and matrices
3. Manipulate images by reading image data into matrices and using matrix operations.
4. Manipulate audio by reading audio data into vectors and using vector operations

Overview

In the real world we often want to manipulate large amounts of data and not just single values. That data can be stored in *vectors* or *matrices*. Two areas where we can have large amount of data are in image and audio processing. By using basic matrix operations on image and audio data we can manipulate the data to do interesting things, including cropping, down-sampling, and filtering.

Part I: Textbook Exercises (Section 2.15 Exercises, Page 79)

#2.1

#2.3

#2.6

Part II: Manipulating Images

In this part of the assignment you will practice loading images into Matlab as matrix data. You will then manipulate the data using matrix operations and display the results.

Background sub-sections A-D will provide some background information and functions for use in this part of the assignment.

You can download the 'monkey.png' image I put online to play with if you like.

Background A: Loading and Display an Image

Matlab has built in functions to load and display images. To load/read an image, you can use the command:

```
im = double(imread('filename'));
```

where 'filename' is the name of the image to load. If the file isn't in your current directory (either look at the top left panel in the Matlab window or type 'ls' to see what's in your current directory) then you must also include the path in the filename. The data is stored in variable im.

NOTE:

- Check out what *im* is, typically and NxMx3 dimensional matrix! Where NxM is the dimensions of the image, and 3 pertains to the Red, Green, and Blue channels (RGB).
- In the Workspace area look at the data type of *im*. Typically it will be uint8. Those of you who have done any website stuff may recall that you usually specify colors as 0-255. 8 bit unsigned integers can represent 2^8=256 values.

To display an image you can use function:

```
imshow(uint8(im))
```

where im is the variable to display as an image.

Background B: Grayscale

We can easily convert an RGB (color) image to grayscale. To do this we use the equation:

```
Gray = 0.2989 * R + 0.5870*G + 0.1140*B
```

Using our matrix im we can then write this as

```
Gray = 0.2989*im(:,:,1) + 0.5870*im(:,:,2) + 0.1140*im(:,:,3);
```

We can open another (different) figure window using **`figure(n)`** where **n** is the number of the figure window. Alternatively if you just type **figure** (without a number n) it just opens a new figure. The next drawing operation (plot, imshow, etc..) will then appear in this new window.

Background C: Brightening (or Darkining) Images

By increasing and decreasing the pixel values, we can brighten or darken the image. For instance:

```
Gray2 = Gray+100;
```

Background D: Cropping

In cropping an image we're creating a sub-image of it, this is akin to selecting a sub-matrix.

Take your original gray-scale image, use matrix selection intervals to select a sub-region, and display that sub-region in a new Figure

Tasks and Submission:

In your report list the commands (make sure they are in an order that will work) that will do the following:

1. Load a file as an image matrix
2. Display the image
3. Convert the image to a grayscale image
4. Display the grayscale image
5. Decrease the grayscale image's intensity by **half**
6. Display the image created in step (5)
7. Create a crop of the image in step (3) from the first 100 rows and first 50 columns of the image made in step (3).
8. Display the image created in step (7)

Part III: Manipulating Audio Data

In this part of the assignment you will practice loading audio into Matlab as matrix data. You will then manipulate the data using matrix operations and playback the results

Background sub-sections A-C will provide some background information and functions for use in this part of the assignment.

You can download the '000.mp3' audio file I put online to play with if you like.

Background A: Loading and Playing Audio Files

To read in an audio file we can use

```
[y,Fs] = audioread(filename)
```

To play audio we can use:

```
sound(y,Fs);
```

You can load the file's data and sampling rate using *audioread* and play it using *sound*.

Background B: Adding Audio Noise

One type of noise is "white noise", which is *random* noise.

In class we have discussed how to make a matrix consisting of random values:

```
A = rand(r,c);  %where r is the number of rows, c is the number of columns
```

Each value of A will then be in the range (0,1)

If you check out the maximum value in your audio array, you will see that this too has a maximum value near but typically below 1

Therefore if we just added the noise and the original audio together we have what's called a *signal-to-noise-ratio* (SNR) of 1:1

Steps:

1. Make an array of random values whose length is the same as your audio
2. Experiment with different SNR ratios by multiplying the noise by different factors. I.e. if y is our original signal and n is our noise we can do:
    a. y2 = y+0.5n

    to obtain a new signal with 2:1 SNR.

Background C: Changing Sampling Frequency

The function *audioread* can return two things: the audio data, and the *sampling* frequency:

```
[y,Fs] = audioread(filename)
```

where y is the data, and Fs is the sampling frequency.

To d*own-sample (or sub-sample)* by ½ means taking every other sample. The result is a new sampling frequency of Fs/2.

To select every other sample from a matrix (or vector) we can do:

```
y2 = y(1:2:end)
```

ASIDE: "High frequencies" (i.e. treble) can get lost if we don't sample often enough (if Fs is too low). Hopefully you see what I mean when you play with this.

Tasks and Submission:

In your report, list the commands (make sure they are in an order that will work) that will do the following:

1. Load an audio file into a vector (or matrix)
2. Playback the file loaded in step (1)
3. Create a version of the audio data that has 2:1 SNR ratio by adding white noise
4. Playback the data created in step (3)
5. Create a version of the audio data by downsampling it by ¼.
6. Playback the data create in step (5)

Submission

Your submission should include a PDF report document **as well as any audio or image files you *use*** inside of a *zip* file (you will submit the zip file).

This document will report your findings for each part. You may include screenshots in the document if you like. See the above "Tasks and Submission" sections for instructions on what you are supposed to do and show

This report should also state things like:

1. What you did
2. How you did it
3. Any additional things you tried or discoveries you made