Armstrong Atlantic State University
Engineering Studies
MATLAB Marina – Image Processing Exercises

1. Write a MATLAB program that will:
   - Load the JPEG image `hibiscus.jpg`.
   - Display the hibiscus image in a figure window.
   - Use MATLAB's array indexing to extract the middle third of the hibiscus image (crop out the left and right hand sides leaving the flower). Display the cropped hibiscus image in a second figure window
   - Use MATLAB's array indexing to extract the hibiscus pistils and stamens (the pistils are the red rods with circles on the ends at center right of the flower and the stamens are the smaller yellow rods just to the left of the pistils). Display the cropped pistils and stamens in a third figure window.
2. For your program of Exercise 1:
   - How is the hibiscus image represented in MATLAB?
   - What is the size of the data structure representing the hibiscus image?
   - What values do the pixels of the hibiscus image take on?
   - What is the size of the data structure representing the cropped hibiscus image?
   - What is the size of the data structure representing the cropped pistils and stamens image?
3. Write a MATLAB program that will:
   - Load the JPEG image `hibiscus.jpg`.
   - Down sample the hibiscus image by a factor of four in both the x and y direction. Display the down sampled image and original image in separate figures and compare the two images.
   - Save the down sampled image of the hibiscus as a bitmap image (`.bmp`).
4. Write a MATLAB program that will:
   - Load the JPEG image `hibiscus.jpg`.
   - Down sample the hibiscus image by a factor of eight in both the x and y direction.
   - Using linear interpolation (MATLAB `interp2` function), up sample by a factor of eight the image obtained by down sampling by a factor eight.
   - Display the original image and the up sampled image in separate figures and compare the two images.

   The `interp2` function can either take row and column vectors or 2D arrays to specify the points of the given data and the points to interpolate values at. The `interp2` function is defined for single and double precision real numbers. The image will need to be converted to type `double` for the interpolation and the result of the interpolation will need to be converted back to type `uint8` (8-bit unsigned integer) for display and storage. Color images are 3D arrays so the interpolation using the `interp2` function will need to be performed on the 2D arrays of red, green, and blue pixels separately and combine the results into a 3D array.

5. Write a MATLAB function named `lighten` that will lighten (or darken) an image by some amount delta. Use the function definition of Figure 1 as a starting point. The lighten function should perform the lightening using array operations (no loops).

```
function outimage = lighten(inimage, delta)
% -----------------------------------------------------
% lighten.m
% -----------------------------------------------------
% lighten lightens or darkens an image
% -----------------------------------------------------
% Syntax: outimage = lighten(inimage, delta)
%   inimage = original image
%   delta = amount to lighten or darken by (-255 to 255)
%   outimage = lightened (darkened) image
% -----------------------------------------------------
```

Figure 1, lighten Function Definition

6. Write a MATLAB test program that uses the `lighten` function written for Exercise 5 to lighten and darken the `hibiscus.jpg` image. Experiment with various values for delta and determine what happens when delta is very large or very small.
7. Modify your `lighten` function from Exercise 5 so that if the function is called without specifying a value for delta, the unaltered image is returned. Hint: modify the function so that the delta parameter is an optional argument. Using your test program developed for Exercise 6, verify that the `lighten` function works for lightening, darkening, and leaving the image unaltered.
8. Rewrite the `lighten` function from Exercise 7 to use iteration (loops) rather than array operations. Test the iterative version of the `lighten` function using the test program from Exercise 6.
9. When values are outside the range that can be stored in variables of type `uint8` (overflow), the value is mapped to 0 if it is less than 0 and the value is mapped to 255 if it is greater than 255. This creates a potential undesirable nonlinearity in operations such as the `lighten` function written for Exercise 5. Modify your `lighten` function from Exercise 7 so that the function ensures that no image intensity values are saturated (values that overflow that are then mapped to the lower and upper limit of the data type). The `lighten` function should ensure that lightened/darkened intensities do not go outside of the 0 to 255 range. Hint: determine the minimum and maximum color intensity and then adjust delta so that after lightening/darkening, no pixel intensities go out of the 0 – 255 range. Test the modified `lighten` function using the test program from Exercise 6.
10. An alternative to adjusting the lightening/darkening amount delta to prevent saturation is to convert the image to type `double`, lighten/darken the image (since data is type `double` saturation is very unlikely), map the intensities back to the range of type `uint8`, and convert resulting image back to type `uint8`. Rewrite the `lighten` function of Exercise 9 so that the lightening operation is performed on an image of type `double` and

the intensities are mapped back to the appropriate range for type `uint8` using either a linear mapping of setting all values that exceed 255 to 255 and all values lower than 0 to 0 before being converted back to type `uint8` and returned. Test the modified `lighten` function using the test program from Exercise 6.

Last modified Thursday, November 07, 2013