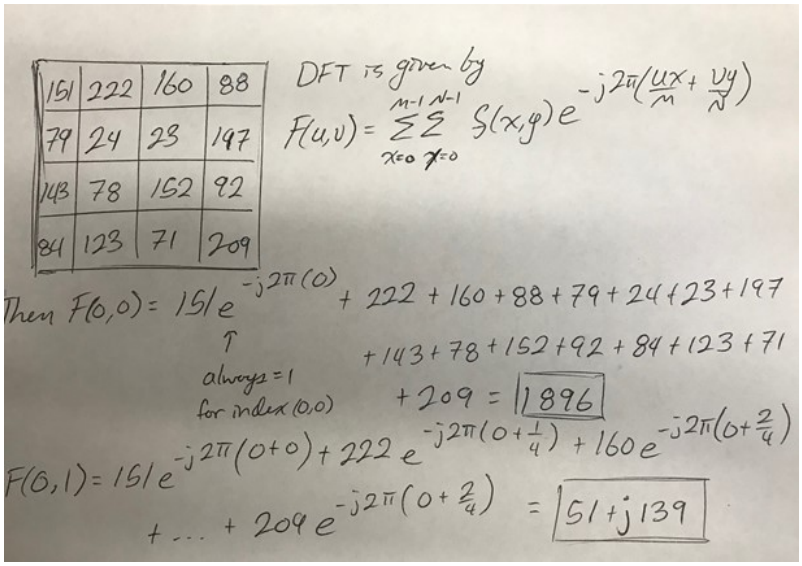
Connor McCurley

EEE 6512 **Homework 4 Due October 13, 2018** Fall 2018

Part I Questions

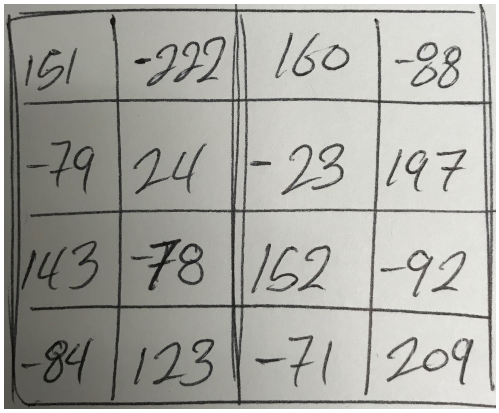
1. As seen below, the first component is worked by hand, while the rest were computed through Matlab.




```
>> a = [151 222 160 88;79 24 23 197; 143 78 152 92;84 123 71 209]

a =

   151   222   160    88
    79    24    23   197
   143    78   152    92
    84   123    71   209

>> A = fft2(a)

A =

   1.0e+03 *

   1.8960 + 0.0000i   0.0510 + 0.1390i  -0.1700 + 0.0000i   0.0510 - 0.1390i
   0.1560 + 0.1640i   0.0870 - 0.1910i  -0.1240 - 0.0580i  -0.0870 + 0.1050i
   0.2760 + 0.0000i  -0.0870 - 0.3790i   0.4220 + 0.0000i  -0.0870 + 0.3790i
   0.1560 - 0.1640i  -0.0870 - 0.1050i  -0.1240 + 0.0580i   0.0870 + 0.1910i
```

Figure 1: Left: First two components of the DFT computed by hand Right: Matlab Solution

2. Transforming the matrix results in the following:




```
>> a = [151 222 160 88;79 24 23 197; 143 78 152 92;84 123 71 209]
aAlt = zeros(size(a,1),size(a,2))

a =

   151   222   160    88
    79    24    23   197
   143    78   152    92
    84   123    71   209


aAlt =

     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0

>> for x=1:4
for y = 1:4
aAlt(x,y) = a(x,y)*((-1)^(x+y));
end
end
>> aAlt

aAlt =

   151  -222   160   -88
   -79    24   -23   197
   143   -78   152   -92
   -84   123   -71   209

>> AAlt = fft2(aAlt)

AAlt =

   1.0e+03 *

   0.4220 + 0.0000i  -0.0870 + 0.3790i   0.2760 + 0.0000i  -0.0870 - 0.3790i
  -0.1240 + 0.0580i   0.0870 + 0.1910i   0.1560 - 0.1640i  -0.0870 - 0.1050i
  -0.1700 + 0.0000i   0.0510 - 0.1390i   1.8960 + 0.0000i   0.0510 + 0.1390i
  -0.1240 - 0.0580i  -0.0870 + 0.1050i   0.1560 + 0.1640i   0.0870 - 0.1910i
```

Figure 2: Top: Image transformed by $(-1)^{x+y}$ Bottom: DFT solution provided by Matlab

The computation of the DFT was provided by Matlab's fft2() function.

3. As seen by the final solutions, the resulting DFTs are just shifted versions of each other. The matrix transformed by the (-1) term shifts the frequencies so that the DC is centered and the other frequencies expand radially from there.

4. Question 7.25:

   a In the HSI space, each pure color region demonstrates a Saturation of 1 and an Intensity of 1/3 (given by equations 7-18 and 7-19 in the textbook). However, each converted color demonstrates a different hue, the R at 0, G at $\frac{1}{3}$ and B at $\frac{2}{3}$, which are the normalized hue values given by equations 7-17 and 7-17 in the textbook.

   b Smoothing the resulting HSI image over the saturation component will not affect the image visually, the only would be in intensity.

   c If smoothing is applied in just the hue channel would not be altered by a kernel smoothing. Only intensity would be affected.

Part II MATLAB Programming

1. This question asked to identify and remove an anomalous frequency in a corrupted image. This was done by observing the DFT of the image and multiplying the frequency image by a gaussian kernel to effectively smooth the high frequency components. The bandwidth was chosen subjectively. The image was then reconstructed using the inverse DFT. While this method was adequate in removing the frequency interference, it resulted in loss of high frequency detail in the spatial domain.

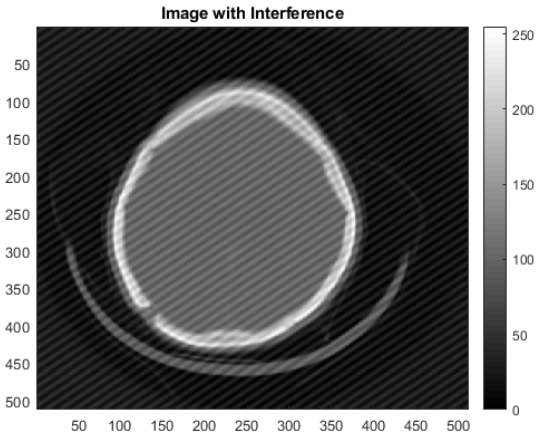


Figure 3: 8-bit intensity image corrupted by frequency noise

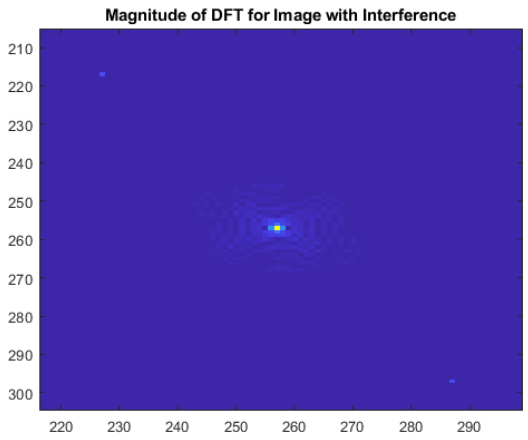


Figure 4: Zoomed in DFT of the corrupted image. Anomalous frequencies can be seen in the corners of the zoomed image.

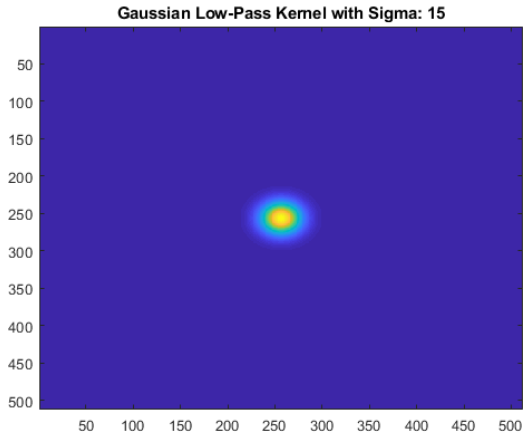


Figure 5: Gaussian kernel with bandwidth $\sigma = 15$

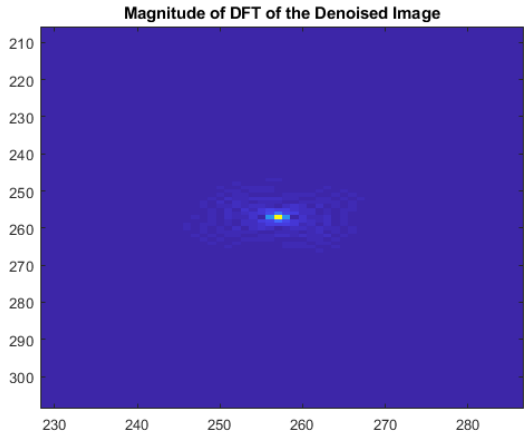


Figure 6: "Cleaned" dft image zoomed in. This image is a result of the multiplication of the gaussian kernel and corrupted dft image. It can be observed that the anomalous frequencies seen in fig 4 have been attenuated.

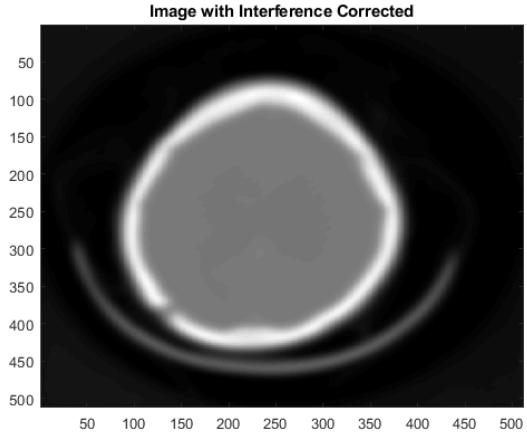


Figure 7: Filtered image of fig 3. Smoothing was performed in the frequency domain.

2. This question asked to convert the gray-scale images from homework 2 into 8-color pseudo-color images with the colors: red, blue, green, yellow, orange, purple, brown, and black. Results of the conversion using intensity slicing are presented below.

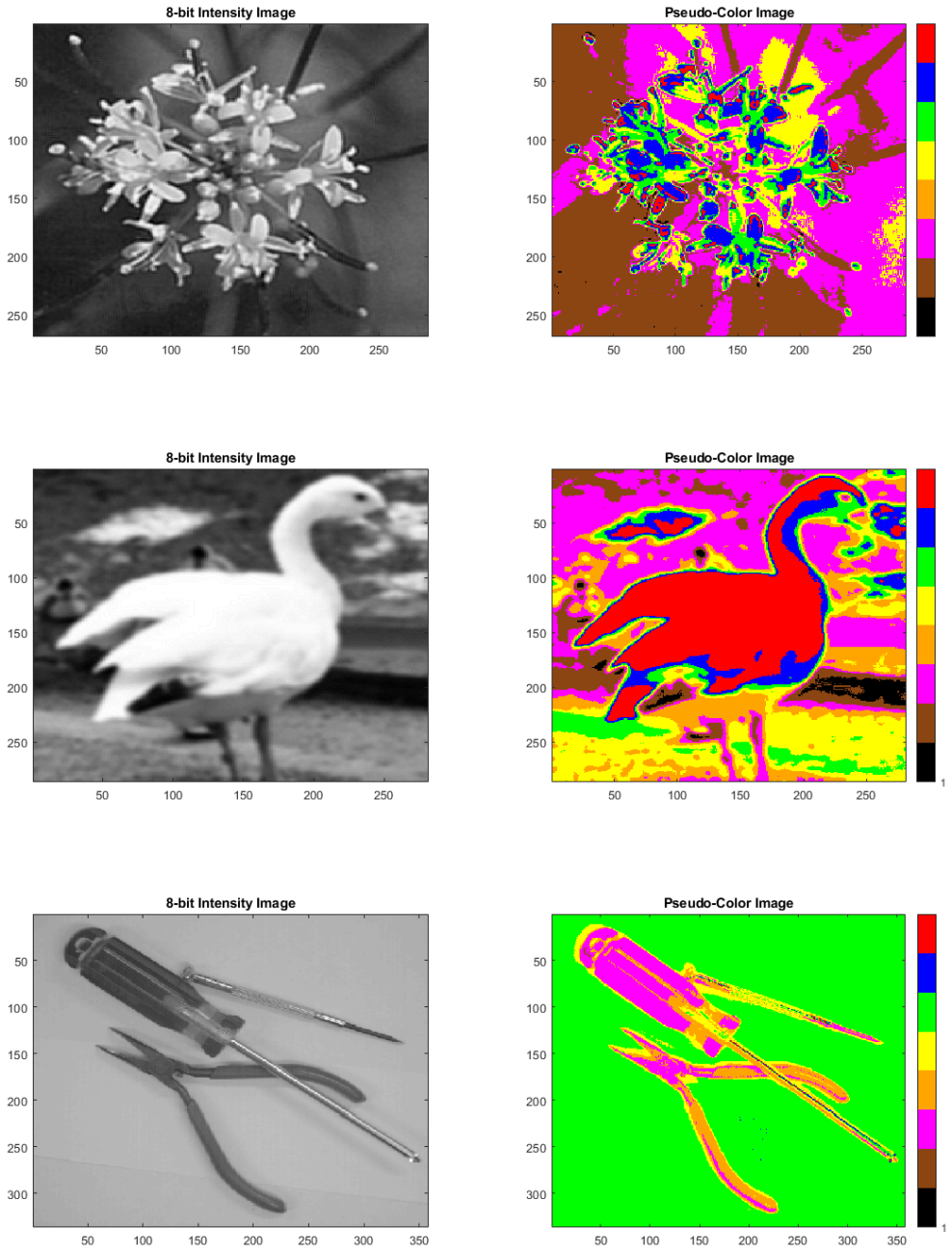


Figure 8: Left Column: 8-bit intensity images Right Column: Corresponding 8-color pseudo images

My code generated the pseudo-color images by first creating a look-up table of integer values. The table contained 8 rows representing the 8 color bins. Each row contained 32 sequential integer values. For example, the first row contained integers 0:31, the second row 32:63, and so on. Each element of the gray-scale image was compared to the look-up table and a label was provided as the corresponding row in the table. This effectively binned all 256 gray-scale values into the 8 color bins. A single color was applied to each bin to get a pseudo-color image.

Part III Extra Credit

For this portion of the assignment, my code converted the scene provided into 3 separate images of colorspace RGB, HSI, and LAB, respectively. Each image was then segmented using color slicing. Prototype regions were selected from areas of each image to include sky, cloud, water, sand, rock, and vegetation. Each prototye region was averaged to provide a single feature vector prototype. Euclidean distance was calculated between each pixel in an image and the set of prototypes. The pixel was assigned the label of the closest prototype. This process was repeated for the RGB, HSI, and LAB color spaces.

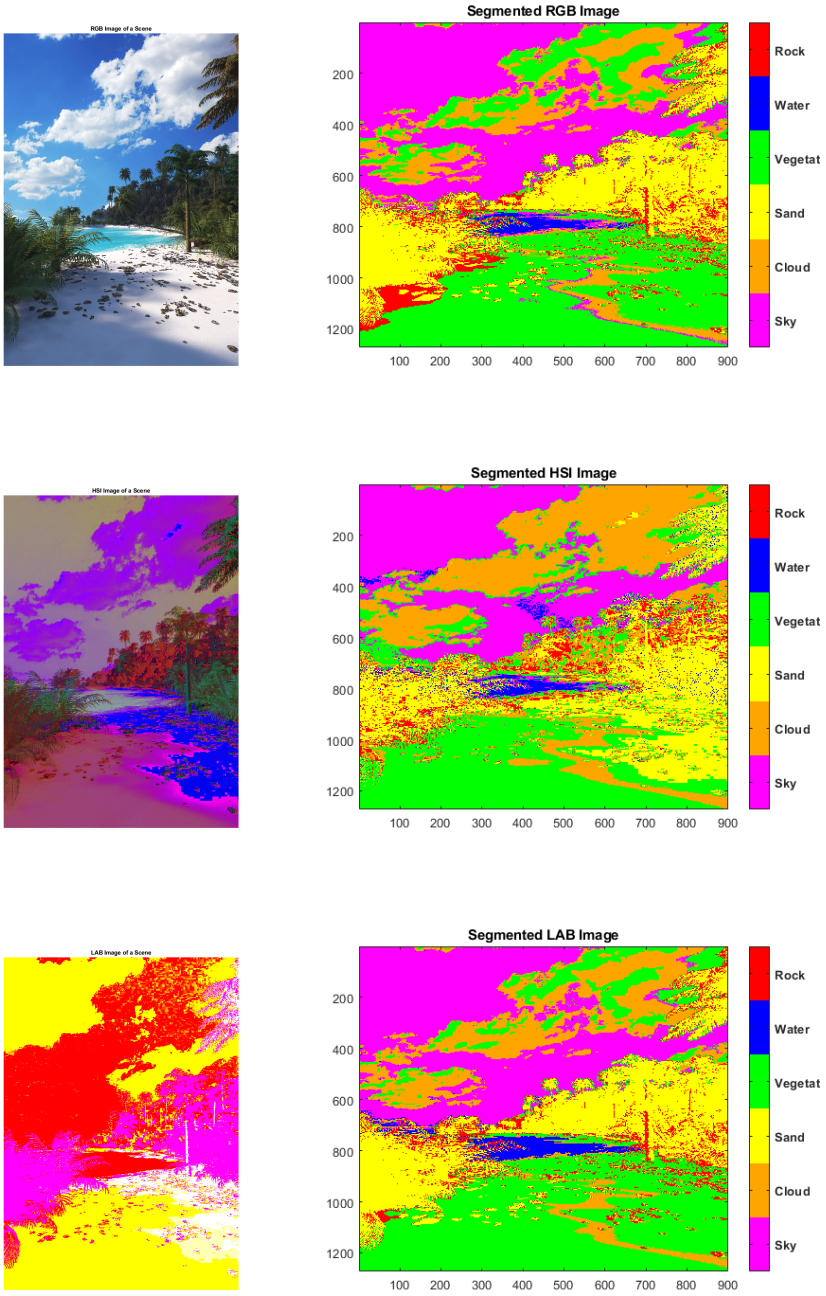


Figure 9: Left Column: Original in rgb,hsi, and lab color spaces from top to bottom, respectively. Right column: Corresponding segmented image

It is worth noting that th
Accompanying code is provided in $mccurleyHW04.m$, $removeInterference.m$, $pseudoColor.m$, and $colorSegmentation.m$