EE435: Biometric Signal Processing

Project 4: Spatial Image Processing

Assigned: Tues 2/7/12 Due: Tues 2/14/12

I. Image Display Functions

MATLAB has several image viewing functions available. These include *imshow*, *image* and *imtool*. The *imshow* function is designed for actual images: binary, grayscale or color. You have used the *imshow* function already in this course. The *image* function is used to visually display the values in a numeric matrix, and is less constrained in the types of data or values you wish to display. The *imtool* function is a very useful interactive function. It allows you to view an image, and while the mouse is moving over the image, it displays information such as pixel location the mouse is passing over, pixel value and size of the image.

Download the "iris.bmp" image from the course website and use *imtool* on it.

Under the "Tools" drop down menu, there are a number of interactive useful tools that can provide information about the image. Experiment using the "Pixel Region" and "Measure Distance" tools. Move the "Pixel Region" tool around the pupil in this image, and take a closer look at the values corresponding to an eyelash. When you do, the grayscale value appears in angled brackets, e.g. <157>, and the corresponding color value on the color cube (with values 0.0-1.0) are identified separately as R, G and B values.

Question: What is the approximate average grayscale value of the glare spot that appears below the pupil, close to the bottom eyelid? (i.e., look at the pixel values, then make an educated guess at what the average value is).

Question: The pupil looks almost circular: using the "Measure Distance" tool, what is its approximate radius?

II. Intensity Transforms

IMPORTANT NOTE: When using the MATLAB *imwrite* function to write out an image file, be sure that the values you're writing out are *uint8*, NOT *doubles*, or else when you double-click on the image in Windows to view it, you may not get what you expect. In any case, whenever you write out an image file in MATLAB, after using *imwrite*, double-click on the new image file in Windows to make sure it looks like you expect it to. If needed, you can convert *double* values to *uint8* using a command like (assuming variable *a* is a double array):

```
>> a = uint8(a);
```

Copy the files "iris10" and "iris10-bright," read them into MATLAB. The "iris10" image is an original image, and the "iris10-bright" image is one to which I've applied a brightening intensity transform.

1. In a 3x2 subplot, display: the original "iris10" image and its histogram, the "iris10-bright" image and its histogram, and the "iris10-bright" image's negative, and its histogram. TURN IN THIS PLOT.

   Question: When you compare the three histograms with the three images they come from, do the histograms make sense? Why or why not?

2. Perform a histogram equalization of the "iris10-bright" image using the *histeq* function and display the result.

   Question: Do you think that visually this is a better resulting image than it was? Why or why not?

3. Perform an adaptive histogram equalization of the "iris10-bright" image using the *adapthisteq* function.

   Question: Is this result any better than straight histogram equalization?

III. Spatial Filtering

1. Smoothing is typically used to reduce the amount of noise in an image. Copy the "OJ-noisy" image and read it into MATLAB. Smooth the image using a disk-shaped mask of radius 3, 21 and 33. These masks can be easily created with the MATLAB *fspecial* function with the 'disk' option. Once the masks are created, use *imfilter* to do the filtering. Compare the results. Create a 2x2 figure with the original and the three filtered images displayed, with each suitably labeled. TURN IN THIS PLOT.

   Question: Which size filter gives the best balance between minimizing the noise and losing the details of the image?

2. Download and bring the "Buster Keaton-jail-bars.jpg" image into MATLAB. Create the following matrices, then use the *imfilter* function with each matrix to filter the this image and display the result.

$$\begin{bmatrix} -0.5 & 1 & -0.5 \\ -0.5 & 1 & -0.5 \\ -0.5 & 1 & -0.5 \\ -0.5 & 1 & -0.5 \\ -0.5 & 1 & -0.5 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$$

   Question: When we are looking for some characteristic or feature in an image, we may filter the image with a special filter so that the feature we are looking for stands out (e.g., has high value). With that in mind, what does each of these two filters apparently look for?

IV. Correlation

Correlation can be used to automatically determine if and where "targets" are located in an image. A target could be a face, a tank, or even specific words or letters within text. In MATLAB, we use the *imfilter* command to perform correlation on images. There are two inputs: a target to look for, and an image in which to look for the target. It is important to convert each of the two input images to doubles (rather than uint8's) before performing correlation. Download the "VinceLombardiQuote" image, as well as the "V.bmp" and "g.bmp" images, and see what they look like. These are all images of text, which are ideal for demonstrating how correlation works.

1. Where is the "V" in the Vince Lombardi quote? Determine its location by the following means. Using the *imfilter* function, perform correlation between the "VinceLombardiQuote" image and the "V.bmp" image. There is a single capital "V" letter within the quote. View the correlation result (which has values in the range [918, 1804327]) using *imshow* with the "[ ]" option.

   Question: Look at the resulting correlation image. How can you use this correlation result to determine the location of the "V" within the quote image? Record the location of the "V" based on the correlation result.

2. Now perform correlation between the quote image and the "g.bmp" image. There is more than one occurrence of the letter "g" in the quote image.

   Question: How can you use the correlation result to tell where all of the g's are located? Using your *threshold* function on the correlation result, record the locations of the g's.

3. Question: Explain what you think the result would be if you performed correlation between the quote image and an inverted V (that is, Λ)? Try it: use the MATLAB *imrotate* command on the V variable to create the inverted V, and perform the correlation with the quote image. You'll probably find that you don't get a usable result.

   Question: What does this tell you about the importance of the orientation of the object you're looking for (i.e., its rotation) when performing correlation?

   Question: Describe how you'd go about locating an object in an image using correlation if you didn't know its orientation.

For a write-up, answer the above questions, and provide the plots called.