EN1610 Image Understanding

Lab # 1: PointWise Image Processing

The goal of this first lab is to

- Familiarize you with the MATLAB image processing toolbox
- Implement some pointwise image processing operations specifically thresholding, sampling, quantization, contrast enhancment

All the images you will need for the lab can be downloaded off the class website, `http://vision.lems.brown.edu/engn161/fall2011/Labs`.

Introduction to Matlab Image Processing Toolbox

Getting Started Viewing Images To load an image into MATLAB, you should use `imread()`. Save the images needed for the lab to the computer you are on and load it into MATLAB. To display a matrix/image you are working on, use `imshow()`. To write the image to a file, use `imwrite()`. In summary, you will need to load an image into MATLAB, which treats the image as a large matrix. Perform your tasks on the image/matrix, and view the results use `imshow()`, then when you are finished, use `imwrite()` to write your data to an image, which you can put into the document you hand in.

Example code: Put this in a file called testfunc.m:

```
function testfunc()
DATA = imread('quart.jpg');
imshow(DATA);
imwrite(DATA, 'myfile.jpg');
```

Conversion of color images Many images come in color and certain operations in MATLAB are only defined on grayscale Images. Use the function `rgb2gray` to covert them to grayscale. You will also have to change the class of the image from uint8 to double in order to perform operations with the data. You will then have to change it back in order to view it with imshow. For example:

```
myImage = imread('myPhoto.jpg');
grayImage = rgb2gray(myImage);
doubleImage = double(grayImage);

...perform operations with doubleImage...
processedImage = someFunction(doubleImage);

finalImage = uint8(processedImage);
```

```
imshow(finalImage)
```

One common mistake users make with the image processing toolbox, and using images, is that some operations are not defined on unsigned integers, so please make sure to convert your images to double using the command `double()`.

PointWise Image Processing

Problem 1. Thresholding

1. Using the white blood cell image1 as seen below, threshold it to make any non-cell area black. To do this, you will need to check each pixel in the image, and if it is over T_0 (where T_0 is the threshold you have chosen) set the pixel's intensity to 0, otherwise, set to 255. You will need to play with T_0 to find its optimal value. You will probably want to pass it into the function as a parameter.

   Sample Code:

```
for (i=1:height)
for (j=1:width)
CHECK pixel value of I(i, j)[I is the image] versus the threshold...
 and act accordingly
end
end
```

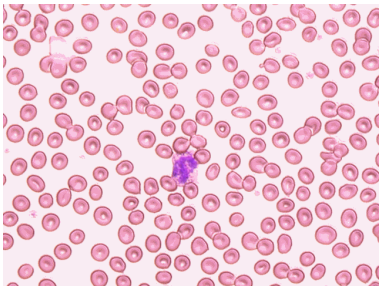

Figure 1: White Blood Cells

2. Using the snowman image 2, perform bilevel thresholding. i.e. set all pixels with intensity between I_0 and I_1 to 255 and set all pixels less than I_0 to 0 and all pixels greater than I_1 to 0. The snowman is the region of interest, you are thresholding for (where bob is, should be all white and the rest should be black).

Problem 2. Contrast/Inversion

1. Increase the brightness of the image3 by 10% or decrease the brightness of the original by 10% by adding/subtracting a value (I_0=25) from each pixel intensity. Try different values of I_0.
   Hint: Make sure you deal with values that fall outside the range of the image.

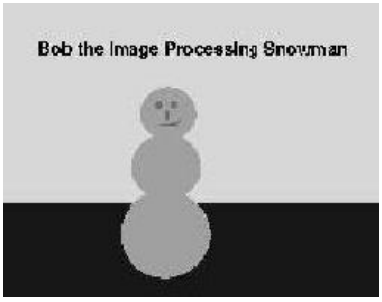

Figure 2: Snowman

Figure 3: Poor Contrast Image

2. Invert the white blood cell image 1 to get something similar to Figure 4.

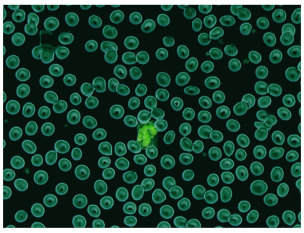

Figure 4: Inverted White Blood Cells

Hint: Make sure pixel intensities are not negative, you may need to shift up the intensities to ensure this does not happen.

Problem 3: Quantization The current cell image 1 has 256 possible values of pixel intensity, reduce this to 8 levels. Make some observations about the quantization levels and image quality. Do this by grouping intensity regions, i.e. set the intensity of all pixels with current intensity between 0-31 to 16, between 32-63 to 48, 64-95 to 80,...etc.

How would you do this for 16 levels?

Problem 4: Sampling Reduce the sampling of the white blood cells image 1, by skipping every other element:
Sample Code:

```
for i from 1 to width in steps of 2
   for j from 1 to height in steps of 2
      outputimage (i, j)= inputimage(i, j)
```

How would you reduce the resolution by a factor of 7?

Playing with a Camera

This lab will also give you a chance to become familiar with a digital camera. Use your own digital camera, smartphone, (or come by the lab (B&H317) to set up a time to use the lab's digital camera) to take a few pictures, and upload them to the computer. You will use these images for the next set of problems.

Sensor Noise

Problem 5: Take 20 images with your digital camera of the same scene under the same illumination condition. Make sure nothing changes in between taking these images, so using a tripod is best. We have tripods in our lab if you do not have one. If you do not have a tripod rest on a set of books or something that has a steady surface.

- Show the mean image and the standard deviation image. Matlab functions `mean()` and `std()` are available
- Show the maximum difference from this mean image. How big is this? Does this depend on the mean?
- Pick one pixel across the 20 images, and plot the histogram `hist()`. Discuss what the distribution of pixel intensity looks like. Why does it look like that?