**EE168** **Lab/HW #2 Hints & Comments**
**Introduction to Digital Image Processing**

*General Hint: Start Early (!)*

## Problem 1

### (i)

- As before, the image type is 'uint8'.
- As you realized in lab #1, the reason for transposing the image between reading and displaying it is that the raster format is *row*-wise, while Matlab reads in data *column*-wise.
- To find the number of bytes in a row: while brute force will work, another method might be found by considering the typical similarity between adjacent rows in an image.

### (ii)

- An example of scaling an image by 2 and inserting zeros:
  num_row = size(im, 1);
  num_col = size(im, 2);
  imzoom2 = zeros(2*num_row, 2*num_col);
  imzoom2(1:2:2*num_row, 1:2:2*num_col) = im;
- Often it is a good idea in general when zooming images to first create a new "canvas" of the zoomed size and then fill in its values as appropriate.
- The syntax of "a:b:c" means a vector of values starting from a, incrementing by b, and going up to a maximum of c.

### (iii)

As always, before you dive into writing code, take a moment or two to think more abstractly about what you want to do and get a feel for what an efficient way to do this would be. Once you understand that clearly, begin to write your program.

### (iv)

In a 1st order hold, interpolated pixels are given values by calculating a weighted average of the pixels' nearest neighbors. The weights are chosen proportionally, in a linear manner, from the pixel element's distance to its neighbors. Ex: [a 0 0 b] becomes [a (a*2/3 + b*1/3) (a*1/3 + b*2/3) b]. The concept is easy, and any difficulty you have might be in implementing it (read (iii) hint).

## Problem 2

**(ii)**

- There are various ways of implementing the input coordinate ←→ output coordinate mapping, using the rotation matrix described in class. Different implementations will differ in their efficiency and whether the output image has missing (unassigned) values resulting from input image coordinates being mapped to output image coordinates that are fractional and need to be rounded. In this case, to present a respectable output image, you should fill in these values in a reasonable manner and not just leave them blank. *Explain your method of filling in these missing values.*
- You will need to create a large canvas matrix in which to insert your rotated images, so that the whole image is seen and nothing is cropped.
- The rotation matrix (as described in lecture) is defined relative to the origin. A good strategy is to rotate the original image around the origin in an appropriate way and then translate it to the center of the aforementioned large canvas image.

**Problem 3**

No comments.

**Problem 4**

No comments.

**Problem 5**

No comments.

**Problem 6**

No comments.