CSE 4/573 Fall 2010 v2 Sep 28 2010

Homework assignment #2 Due Wednesday Oct 6

1. In our database http://www.cse.buffalo.edu/courses/cse573/peter/dbs there is a file FindText.mat. Start Matlab, load this file, and you will find a uint8 image IM in your Matlab workspace. This file contains some hidden text. Submit an m-file with first line

```
function ReadIt=FindText(IM)
```

where

IM: the specific unit8 file loaded from FindText.mat
ReadIt: a unit8 image of the same size where the text contained in IM can be easily read using imshow(ReadIt)

Your code should perform a grayscale transformation of your own design which makes the text hidden in IM easily readable. Give some thought to how you can narrow down your search for where in the image, and at what grey levels, text may be written.

2. The method for affine geometric transformation described in lecture slides 04-08 to 04-14a can be extended to nonlinear geometric transformations as illustrated in this problem. Submit an m-file with first line

```
function ImOut=NLGT(InIm,Interp)
```

where

InIm: a 64x64 uint8 input intensity image
Interp: a string, either `nn` or `lin`
OutIm: a 64x64 uint8 output intensity image

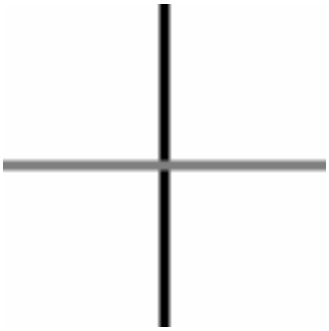
Assume that InIm has been created by sampling an original continuous input image with $\Delta x=\Delta y=1/32$ beginning with inital values x=y=-1. The geometric transformation we wish to produce is

$$x' = (x+y)^3$$
$$y' = (x-y)^3$$

where (x,y) are the original image coordinates and (x',y') are the transformed coordinates. That is, the transformed image should have at (x',y') the same grey level as the original image at (x,y). OutIm should be a sampled version of the transformed image with $\Delta x'=\Delta y'=1/32$ beginning with inital values x'=y'=-1.

Here is an example. The left image, when transformed and nearest-

neighbor interpolated using this geomentric transformation, becomes the right image. It is rotated and the bars thicken with distance from the center of the image.



Original Image

After geometric transformation

3. Write a script m-file (not a function m-file) hw2p3.m that prompts for the name of an input file, assumed to be a color jpeg file, and then prompts for three numbers:

- A magnification factor, any positive real number
- A vertical shift, any non-negative real number
- A horizontal shift, any non-negative real number

The image magnified by the magnification factor and then shifted by the specified shifts should be computed and displayed. Use bilinear interpolation for each color component separately,zero-padding as necessary. The size of the output image should be the minimum size required to display the magnified and shifted output.

So for instance if the input file InIm is say 128x256 (128 rows by 256 columns) and the magnificiation factor is 2.1, then the bounds for the magnified image are x'=2.1*128=268.8, y'=2.1*256=537.6. If the vertical shift is say 11.6 pixels and the horizontal 0.0 pixels, then the bounds become x'=268.8+11.6=280.4, y'=537.6+0.0=537.6. Thus the size of the output image needs to be 281 rows by 538 columns. To compute all the 281x538 output image's pixel values using bilinear interpolation you will have to zero-pad the input image, since some of the output pixel locations map back outside the original 128x256 input image.

4. Design a filter that works as follows. First specify an odd integer m, which sets the size of the mask to be mxm. Take the center pixel in this mask to be the origin of the mask. Next specify a weight vector w of dimension m^2. For each pixel (i,j) in the input image g(i,j), write down the pixel values g(i+k,j+l) which lie under the mask (ie. at k,l =-(m-1)/2...+(m-1)/2) in ascending

order, smallest to largest. Multiply the first (smallest) by w(1), the next by w(2) and so on. Add these m^2 products up to get the value of the filtered output image at the pixel location (i,j). Repeat for all (i,j) in the image. This is called a *weighted rank filter*.

Formally, submit an m-file WRF.m whose first line is

```
function OutIm=WRF(InIm,w,m)
```

where

InIm: a double normalized intensity image;
m: mask size (a scalar);
w: weight vector (an m2x1 column vector);
OutIm: a double normalized intensity image

Note that the weighted rank filter is a generalization of the median filter. For instance if we pick m=3 and we set w=[000010000] then OutIm will be exactly the median-filtered version of InIm (for a mask of the given size). But we can do more than median-filter with this filter: for example, with m=3 and w=[0.5 0 0 0 0 0 0 0 0.5] the rank-order filter will give us, for each pixel in the image, the average of the largest and the smallest pixels under the mask. For m=[0 0 .2 .2 .2 .2 .2 0 0] we have a filter that will reject the two largest and two smallest values under the mask while averaging the others. This is called an outlier rejection filter.

Note also that you have to zero-pad the original image to apply this filter. For instance, if m=7, you have to add 3 blank rows at the top, bottom, left and right or else the mask will run off the image.