Additional info to Computer Exercise 2

Anders Sjögren
Department of Mathematical Statistics
Chalmers University of Technology

January 21, 2006

Here are some (hopefully) clarifying remarks on computer exercise 2 for the Statistical Image Analysis course of spring 2006.

Part 0: Resampling

The notion of linear interpolation means that if you know the value of a function at two points, you approximate the function by a linear relationship between the function values at the points. In our example, we know the pixel intensity values at the columns of the original image, i.e. in the x-locations 1 to 512. When we stretch the image in the x-direction to 753 pixels, we have to guess the values of the new pixels that are placed in between the original ones.

Now, the task consists of two parts:

- *Figure out at which x-locations of the old image the new pixels will be located.* 1 should correspond to 1 and 753 should correspond to 512. It might be easier to first try a toy example of extending 3 pixels to 4 pixels.
- *Calculate the interpolated pixel values.* For each pixel in the new image, you now know the corresponding x-location in the old image. Using the pixels to the left and right of that location, figure out the linear formula describing the approximated intensity values between these pixels, as a function of the x-location. I.e. identify k and m in:

$$\text{intensity} = kx + m$$

Insert the x-location of the pixel in the new image and you get the interpolated intensity.

In the process above, you'll have to loop over the columns but you don't need to loop over the rows. Looping is time-consuming in matlab, so avoid it if you can.

In matlab, two commands that both can be used to solve this task if you get completely stuck are `imresize` and `interp2`. They will of course be much faster as well, but will not give you the same understanding.

Part I: Image filtering

Not much to add...

Part II: Thresholding and boundary detection

To start with, it is sometimes useful to know that matrices in matlab can both be indexed by pairs of column and row and by single indicies. This is for example useful if you want to set all the diagonal elements of a matrix to 1, or certain pixels in an image indexed by pairs of column and row to one. This can be performed by `M(sub2index(size(M),row_indicies,col_indicies))=1`. The single index goes from top left downwards and then to the next column, etc., and thus range from 1 to the column count multiplied by the row count.

Area filling

Hint: One way of performing the area filling is to start with one or several pixel(s) inside the contour, called a seed, and let it grow by iteratively using a morphological filter *(which one?)*. After each iteration, you have to make sure the filling does not enter the contour, as it would then pass outside the contour in the next iteration. Use the help browser to find out how to perform the filtering. Just search using the name of the morphological filter.

An idea is to first make the filling in a separate image, which begins with being completely black, and then superimpose the filling on the original image. This way, the morphological filter will not act on unwanted pixels in the image, such as the contour.

To identify a proper starting seed inside the contour, you can:

- try to do it completely generally and automatically (which might be hard)
- do it automatically but use the fact that the images look in a certain way, e.g. that if you take a point 100 pixels to the right and 100 pixels down from the first point in the contour, it will be inside the contour, or
- make the user report the location by a mouse click in the image, using the matlab function `ginput`.

An alternative is to fill the outside of the contour by using a seed outside the object, which might be easier to find. This can then be used to find the filling of the inside of the object.

If you want to improve the speed of the filling, you can perform the filling on a smaller part of the original image only just containing the object.

If you want, you can show the filling image after each iteration so that the user can watch the filling grow.

One more thing, you have to decide how long you iterate the filtering, i.e. when to stop. You either do this by stopping when the last filtering gave no change, or you do it by setting a fixed, sufficiently high number of iterations (which of course depends on the characteristics of the image, such as the size).