Image Segmentation

1. Point and line detection

(a) Create a synthetic image of a line at angles 30^0, 45^0, 60^0 with the *y*-axis (see Appendix). Use the line filters (slide 6 of the computer class presentation on segmentation) on these images, and comment on the response. Is the behavior different for angles that are not multiples of 45^0, and why?

(b) Now make another image of lines at an arbitrary angle. Modify the method of the Appendix to make lines of various thicknesses (2, 3, and 4 pixels), and repeat the previous exercise. What do you observe?

2. Thresholding

Find the optimal threshold for image *dodecahedron_example.png* using the optimal thresholding method of Gonzalez. Show the results of segmenting the image.

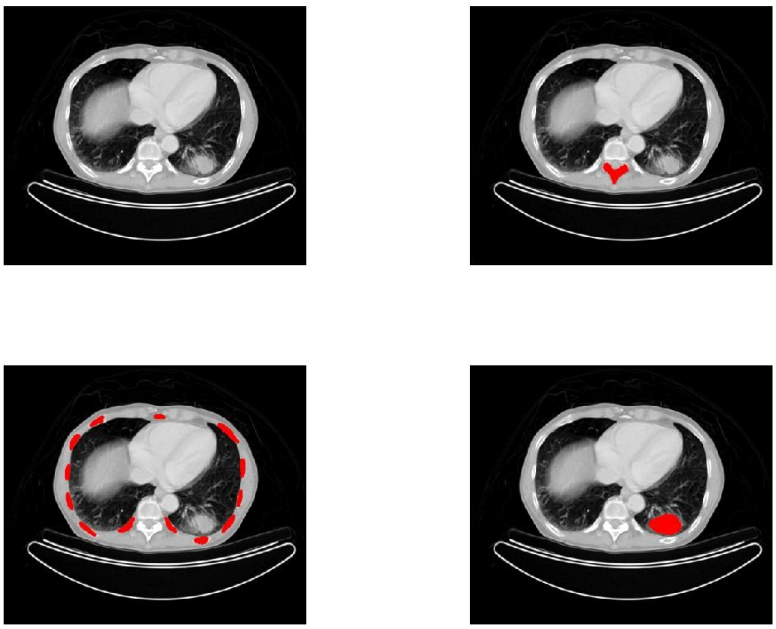

Figure 1: Image medtest.png and the regions selected for segmentation.

3. Region growing

(a) Perform region growing segmentation on image *medtest.png*. By using different seed points, try to segment the regions shown in Figure 1.

(b) Calculate and show the histogram of the image. Try to segment the image (for example, the first region in Figure 1) by thresholdiing. Why is region growing fundamentally different than thresholding techniques?

Appendix

Manipulating matrix indices in MATLAB The function `find` can be used to access matrix indices that satisfy a certain property, e.g. `find(A>10)` returns the indices of all elements that are larger than 10. These indices can then be used to access elements of the array or in themselves, as the coordinates of points that satisfy a certain property. This is a very powerful function in MATLAB and in this example we will show how to use it to make a synthetic image of a line of a certain slope.

One way of expressing a line is by $y = \lambda x$ + c, where $\lambda = tan(\theta)$ is the slope of the line. We want to find indices (coordinates) in the matrix A that satisfy this equation. We start by first expressing all indices of the matrix, by using `find` with a property that is always true, e.g.:

```
A = zeros(100); [j, i] = find(A<Inf);
```

This will give us all the indices of all the elements of A (we first define A as an empty image of 100^2 pixels). Index j is the y-coordinate and i is the x-coordinate. Now we will enforce the line equation $y = \lambda x + c$ or $y - \lambda x - c = 0$. In fact, due to discretization we want to find all points that are within one pixel of this line, so all pixels from which the line passes. This can be written:

```
q = find( abs(j-tan(theta*pi/180)*i)<=1);
```

Note that here we used `find` to give us one linear index, instead of two x- and y-coordinates. This is because we can also access matrix elements with this linear index. We make the line by giving some non-zero value to the elements of A to which q points:

```
A(q) = 100;
```