1 Edge Detection

In this session we will study various edge enhancement methods and edge detectors using MatLab. In MatLab, we can import images using

```
I = imread('chest.pgm');
```

which yields an array of type `uint8`. This does not support signed values or the square root operation, so you will need to convert this to doubles using

```
F = im2double(I);
```

Alternatively, we could of course have typed:

```
F = im2double(imread('chest.pgm'));
```

Once imported, we can convolve images using

```
F2 = imfilter(F,h,'replicate','conv');
```

in which `h` denotes the convolution kernel. The `'replicate'` parameter reduces image boundary problems, by replicating border pixels beyond the image domain. The kernel `h` can be specified by

```
h = [-1,0,1];
```

for simple x derivatives. The y derivative can be obtained either by using the transpose `h'`, or more explicitly

```
h = [-1;0;1];
```

Various special filter kernels can also be obtained using the `fspecial` method (see MatLab help).

Dilation and erosion can be performed using the `imdilate` and `imerode` functions:

```
F2 = imdilate(F,h);
```

with `h` a structuring element specified by an array such as

```
h = [0,1,0;1,1,1;0,1,0];
```

for a 4-connected dilation. Function `imerode` has the same syntax. See MatLab help for details.

2 Excercises

Exercise 1.

Implement MatLab functions to compute gradient magnitude, using simple differentiation, and the Sobel, Prewitt, and Roberts' cross methods. Note that we want a grey-scale output, and not a binary image containing just the detected edges (this means you must not use the `edge` function).

Apply them to the `chest` image. We can add noise of various kinds using `imnoise` (see MatLab help for details). Explore how the performance of the different filters degrades as a function of noise level, by creating different copies of the `chest` image, with different levels of Gaussian noise, and observe how the edge detection is degraded. Which methods are most sensitive to noise?

Exercise 2.

Implement the morphological edge detectors discussed in the lecture. Compare them to eachother and to the previous methods in the same way as above.

Exercise 3.

MatLab also implements the Canny edge detector, implemented through the `edge` function:

```
BW = edge(F,'canny');
```

In this simplest form it chooses default values for the thresholds (high and low), and the smoothing parameter σ. Look in MatLab help to see how this is done. Using the same methods as above, test how the Canny edge detector performs as a function of added noise. Also test how parameter settings affect the result. Discuss your results qualitatively, and indicate which settings at different noise levels you think are best and why.

Exercise 4.

Study the paper "Contour and boundary detection improved by surround suppression of texture edges" by Grigorescu, Petkov, and Westenberg provided on Nestor. This is the predecessor of the methods described in the lecture. The method can be accessed through a web implementation found at:

```
http://www.cs.rug.nl/~imaging
```

Click on "Canny edge detector (Canny filter) for image processing and computer vision:", which will lead you to a web application which can perform both the Canny detector and two surround inhibition methods. Select the `kanizsa.png` image as the target and update the view. Try various settings of surround inhibition, and explain the differences in output. Do the same for `popout.png`. What is the difference with `kanizsa.png`?

Hint: Set the sigma (around 4 seems to work) and $K2$ (around 8) parameters high enough to get the expected inhibition effects.