Technische Universität München
Fakultät für Informatik
Prof. N. Navab, Ph.D.

SS 2013
Exercise Sheet 1
April 18, 2013

Exercises in Computer Aided Medical Procedures II

Exercise 1 (P) MATLAB - the MATrix LABoratory

Go through `FingerExercises.m` in order to refresh the very basics of MATLAB if necessary. Otherwise you can proceed with exercise 2.

Exercise 2 (P) Basic image processing

MATLAB stores every image (volume) in a two-dimensional (three-dimensional) matrix. The goal of this exercise is to become familiar with the basics of image processing taking the matrix-oriented storage into account.

a) Use the function `imread` to load the image *mrihead.jpg* and store it in the variable im.

b) Use the function imshow to visualize this image. Next open a new figure-window (using `figure`) and try to obtain the same visualization with the function `imagesc` - you will also need the functions `colormap` and `axis`.

c) Now visualize only the upper left quarter of the image in the range [100,150].

d) The MATLAB image processing toolbox provides you with some functions for histogram equalization which can improve the contrast of the image. Use the function `adapthisteq` in order improve the contrast of your image.

e) Unfortunately the histogram equalization enhances also the noise. Find out how to smooth your images with a Gaussian filter by typing `help imfilter`. Don't forget to use to option `'replicate'` in order to avoid black boundaries after filtering! The precise meaning of this option will become clear in the following exercise.

f) Open a new figure and visualize the difference between the original image and the enhanced image. Do not forget to convert both images to double (`double( )`). Please keep in mind that all MATLAB image processing functions might behave different for `double` and all integer data types like `uint8, int16` and so on!

Exercise 3 (P) Finite Differences

Many advanced algorithms for filtering, segmentation, and registration are inspired by physical models. For this reasons many of these algorithms require the computational solution of partial differential equations (PDEs). One possibility for the computational solution of PDEs is the approximation of differential operators using finite differences. The aim of this exercise is to implement functions for computing approximations to the following differential operators:

a) gradient: Let $u : \mathbb{R}^2 \to \mathbb{R}, (x,y)^T \longmapsto u(x,y)$ be a real-valued function, then the gradient of u is defined as

$$\nabla u(x,y) := (\frac{d}{dx}u(x,y), \frac{d}{dy}u(x,y))^T \tag{1}$$

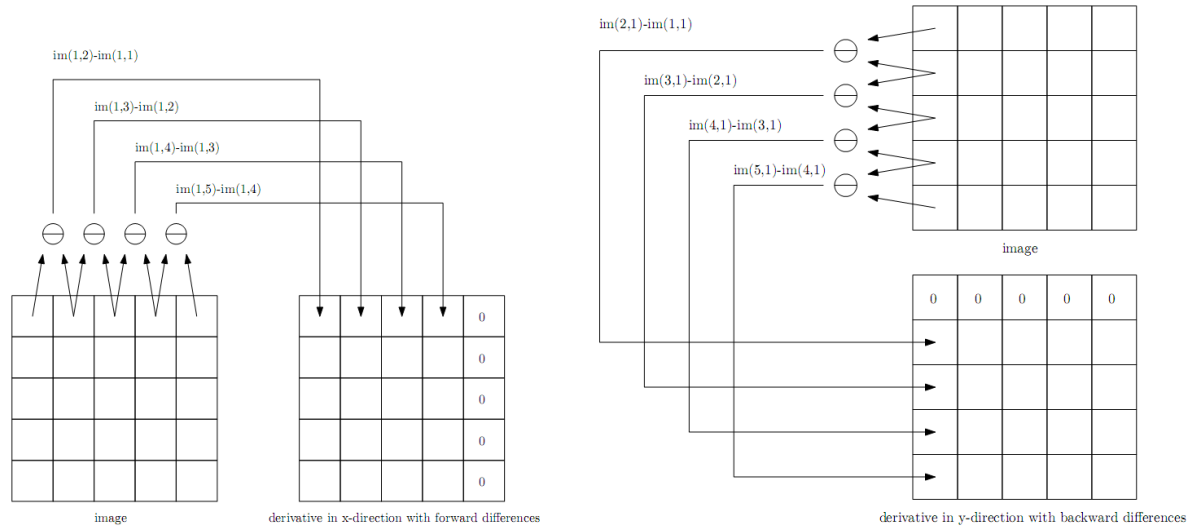

Figure 1: (a) forward differences in x-direction $\frac{d}{dx}im(i,j) \approx (im(i,j+1) - im(i,j)/1$, (b) backward differences in y-direction $\frac{d}{dy}im(i,j) \approx (im(i,j) - im(i-1,j)/1$

b) divergence: Let $v : \mathbb{R}^2 \to \mathbb{R}^2, (x,y)^T \longmapsto (v_1(x,y), v_2(x,y))^T$ a real-valued vector-field, then the divergence of v is defined as

$$\text{div}v(x,y) := \frac{d}{dx}v_1(x,y) + \frac{d}{dy}v_2(x,y))^T \tag{2}$$

Please complete the two functions grad and div which should compute approximations to these differential operators. The gradient should be computed with forward differences and the divergence should be computed with backward differences. At the border you have the problem that you cannot compute the finite difference approximations. It is an usual assumption in image processing that the derivative across the boundaries of an image are 0 -mathematically speaking we are assuming Neumann boundary conditions.

Remarks:

- It is not necessary to use a for-loop in order to compute the finite differences! If you have done exercise 1 carefully, you should know how to do it!
- You can test your functions by computing the Laplacian ($\Delta := div(\nabla)$) of the image `im` with:

```
[dxim dyim]=grad(double(im)); D = div(dxim,dyim);
```

Now visualize the difference between D and the outcome of

```
imfilter(double(im),[0 1 0; 1 -4 1; 0 1 0],'replicate').
```

Please note, that the Laplacian serves as a good edge-indicator!