Centre for Image Analysis
Swedish University of Agricultural Sciences
Uppsala University

Computer Assisted Image Analysis

Exercise 1, VT2009: Introduction

The aim of this exercise is to familiarize you with some common algorithms of computerized image analysis, e. g. pixel-wise operations, filtering, image arithmetics, and the fast Fourier transform. As long as you attend the lab session the exercise can be corrected orally at the lab session without any written report. If you don't finish all the questions or doesn't attend the lab session a written report with all the remaining answers has to be handed in as a pdf.

1 Getting started

Log on to one of the workstations. Matlab is started by giving the UNIX-command:

```
matlab-7 &
```

Remember to start Matlab from a directory where you have write permission. After a while you will be presented with a graphical interface to Matlab. Add the necessary paths by using the Matlab commands:

```
addpath('/it/kurs/bild1/matlab');
addpath('/it/kurs/bild1/images');
```

This gives you access to some custom made functions that we have made to decrease the complexity of this exercise. The files can also be found as a zip-file on Studentportalen. These functions are ordinary text files and can be viewed in an ordinary text editor (or Matlab's m-file editor). E.g., this UNIX command will print the function `filterimage` in the terminal window:

```
cat /it/kurs/bild1/matlab/filterimage.m
```

A useful Matlab command is the `help` command. If you type `help` followed by the name of a standard function in Matlab, valuable information on how to use that function will be displayed. Try for example:

```
help find
```

First you get the syntax, and at the bottom there is a small example of how to use it.

Since the paths are set (using `addpath` above), you can start by reading the image **napoleon.pgm** by using the Matlab command:

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

The image is now stored as intensity values in the 2D matrix `I`. Since an image is stored as an ordinary matrix in Matlab, all matrix operations available can be used on the image (e.g., addition, subtraction etc).

2 The Matlab Image Tool and the Histogram

We will now examine the image using Matlab's built in image tool **Image Tool**. Open the image in **Image Tool** by using the Matlab command:

```
imtool(I)
```

where `I` is the image. **Image Tool** will display two windows; the **Overview** window and the **Image** window. The **Image** window has a menu bar and a toolbar with various buttons at the top, a window pane displaying the image in the middle, and a status bar showing various information at the bottom.

To measure pixel values in the image you can move the pointer over the pixel. The status bar shows the current graylevel as:

```
Pixel info: (x, y) f
```

where `f` is the graylevel at the position (`x`, `y`) in the picture. You can also choose **Tools→Pixel Region** in the menu (or by pressing the Inspect pixel values button in the toolbar). This will open the **Pixel Region** window and show a cross hair marker in the image. The **Image Region** window will show the pixels in the part of the image covered by the cross hair marker.

1. ***Where in the image is the pixel*** $(1, 1)$ ***located and what is the graylevel value?***

Open the images **naopleon.pgm**, **napoleon_light.pgm**, and **napoleon_dark.pgm** and display their graylevel histograms by selecting **Tools→Adjust Contrast** in the menu (or by pressing the Adjust contrast button in the toolbar). The histogram of the image will appear in the new **Adjust Contrast** window with a pink interval defining the graylevel interval of the image file that is displayed on the screen.

The contrast and brightness of the displayed image can be changed by moving the red handles and changing the length of the pink interval in the histogram. This is only a display function and will not change the values in the image file. Try to change the contrast and brightness of the three Napoleon images so that they all look the same.

2. ***Explain what contrast and brightness are. What change in the histogram is related to the contrast and what change is related to the brightness? Illustrate this by drawing graphs showing how the graylevel transform changes as the contrast and brightness are varied.***

Try setting **Eliminate outliers** to 10% in the **Scale Display Range** section and press the **Apply** button.

3. ***Explain what*** **Eliminate outliers** ***does.***

3 Viewing Images and Saving Images or Figures

There are other ways of displaying the image in Matlab than using the image tool. Aside from `imtool` you have the functions `imshow`, `image` and `imagesc`, which all display the image in different ways. `imshow` shows the image, using the right axis ratio and original graylevel scale of the image. `image` uses Matlabs method for displaying matrices as images. It uses the graylevel scale of the image, but it does not use the right axis ratio. `imagesc` works just as `image` but rescales the graylevels to use the full colormap. To see the color map you can give the Matlab command `colormap` after you have displayed the image, or you can choose **Insert→Colorbar** in the menu of the figure window. To get a notion of the difference between `image` and `imagesc` you should try the following Matlab commands:

```
A = [1:10]' * [1:10] .* 0.2;  % create a 10x10 matrix of values from 0 to 20
image(A)                      % show matrix A using 'image'
colorbar                      % add a colorbar showing the colormap
figure                        % open a new figure window
imagesc(A)                    % show matrix A using 'imagesc'
colorbar                      % add a colorbar
```

The text written after the `%` is just a comment and will not be interpreted by Matlab. You can of course also see the pixel values of an image directly in the matlab console by just typing the matrix name without a semicolon, e.g., `A`.

I recommend using `imshow` if you just want to view an image, `imtool` if you want to examine an image thoroughly, and `imagesc` if you want to display a matrix which is not necessarily an image. This will come in handy both during the computer exercises and the project later on in the course.

To save an image you can use the `imwrite` command, but make sure that you are in a directory where you have writing rights.

```
imwrite(I,'my_napoleon.pgm')        % write image I to file 'my_napoleon.pgm'
```

use the Matlab `help` if you want more information on how to write images to different file formats. If you want to save an entire figure (including the parts around the image like, e.g., the colorbar) you can use the `print` command.

```
imagesc(I)                          % show image in figure
colorbar                            % add colorbar
print(gcf, '-dpng', 'nap_fig.png')  % write current figure to file 'nap_fig.png'
```

If you prefer using the menu for saving the contents of a figure you can use **File→Save As...**, and choose a suitable file format and filename, to save the contents of the figure.

A command which can be used to save the contents of any window as an image is the UNIX command `import`. In a UNIX console you just type `import` followed by a filename. When executing the command (i.e., hitting <return>) the pointer will change into a cross hair and you can select which window to save. The file format will be determined by the filename extension. E.g., if you have the figure from the previous Matlab code still open, just write the following in your UNIX console.

```
import nap_fig.png
```

When you hit <return> you can choose the figure window and the contents will be saved as a PNG image to the file `nap_fig.png`.

The commands for saving a figure or image will come in handy when putting together your reports for the computer exercises and project later on in the course.

4 Pixelwise Transforms

Start by creating an image to test some standard transformations on by using the Matlab command:

```
I = uint8( ones(256, 1) * [0:255] );
```

You now have an image which you will transform by using four different pixelwise transformations. Now, try NEUTRAL, INVERT, EXPONENTIAL and LOGARITHM on the ramp image by using the following Matlab commands:

```
J = neutral(I);
K = invert(I);
L = exponential(I);
M = logarithm(I);
```

4. ***Explain the resulting images, describe and draw a graph of each transform. What happens to a "normal" image, for example* napoleon.pgm*, when the transforms are applied?***

Histogram equalization is also a pixelwise graylevel transformation. Continue to work with the three Napoleon images. Perform histogram equalization on them by using the Matlab command:

```
J = histeq(I);
```

5. ***Explain how histogram equalization works in theory. Compare the histograms of the original images and the output images. Do the changes to the histograms and the images agree with the theory of histogram equalization?***

5 Aliasing when Sampling

Open the file **zebra.pgm** in **Image Tool**. Why doesn't the overview look the same as the image? Alter the size of the **Overview** window with the mouse by moving the lower right corner. Make the image smaller as well as larger. Can you make the overview completely white just by resizing the window?

6. ***Explain why the contents of the overview image seem to differ very much depending on the size of the overview window.***

The effect in the overview window is the same as the effect that occurs when resizing an image. Try to resize the zebra image to 400x400 by using the Matlab commands:

```
J = imresize(I, [400 400], 'nearest');
K = imresize(I, [400 400], 'bilinear');
```

where `I` is the zebra image. Open the resized images in **Image Tool** but close the **Overview** windows for both images this time.

7. ***Explain why the contents of the images differ.***

6 Local Filtering

Load the predefined smoothing- and sharpening-filters by using the Matlab command:

```
loadfilters
```

This loads the predefined filters found in the table below.

Filter	Description	Filter	Description
`mean3x3`	Mean 3x3	`gaussian3x3`	Gaussian 3x3
`mean5x5`	Mean 5x5	`gaussian5x5`	Gaussian 5x5
`mean7x7`	Mean 7x7	`gaussian7x7`	Gaussian 7x7
`mean11x11`	Mean 11x11	`lp3x34n`	Laplace 3x3 4n
`mean15x15`	Mean 15x15	`lp3x38n`	Laplace 3x3 8n
`mean25x25`	Mean 25x25	`lp5x5`	Laplace 5x5
`meancircular5x5`	Flat Circular 5x5	`crisp3x34n`	Crisp 3x3 4n
`meancircular7x7`	Flat Circular 7x7	`crisp3x38n`	Crisp 3x3 8n
`median3x3`	Square 3x3	`crisp5x5`	Crisp 5x5
`median5x5`	Square 5x5	`sobelvr3x3`	Sobel vertical right
`median7x7`	Square 7x7	`sobelvl3x3`	Sobel vertical left
`median11x11`	Square 11x11	`sobelhb3x3`	Sobel horizontal bottom
`median15x15`	Square 15x15	`sobelht3x3`	Sobel horizontal top
`median25x25`	Square 25x25		

Open the image **wagon.pgm** and filter it using the Matlab command:

```
J = filterimage(I, F);
```

where `I` is the image and `F` is a filter from the table. Test at least **three** different kinds of filters, among which there should be at least one sharpening- and one smoothing-filter. Also examine the weights in the masks by using the Matlab command (note: no semicolon):

```
F.filter
```

where `F` is one of the filters in the table.

8. ***For each filter, examine the effect of the filter and explain what the filter does to the image.***

9. ***For each filter, how does the size of the filter mask affect the result?***

Open the image **wagon_shot_noise.pgm**. Perform several median filterings on the image using different sizes of the filter masks.

10. ***Compare visually the effect of median filtering to the effect of mean filtering and explain the differences.***

11. ***What is the advantage and disadvantage of each filter when it comes to the result of the filtering? In general the median filter is more time consuming, why?***

7 Image Arithmetics

Another way of comparing images, that is not only performed visually, like in question 10, is by subtracting two images and examining the difference image. Use the Napoleon image and perform a subtraction of a mean-filtered and a median-filtered version of the image. For a relevant comparison, the images should be filtered using the same original image with the same size of the filter mask.

12. ***Describe the difference image. This is easier if first adjusting the contrast and brightness. Is the difference coherent with the answer to question 10? Give cause!***

A SPECT image shows the activity in the brain. When evaluating patient data it is often of interest to make a comparison with an image of a "standard" healthy brain. Standard data is created by averaging over a large number of images of healthy brains. The difference between the standard data and the patient data is found by subtraction.

Images **brain1.pgm** and **brain2.pgm** show two SPECT images of healthy brains. Image **brain3.pgm** shows a SPECT image of a brain from a patient with a stroke.

13. ***How can a "standard" healthy brain, or a mean image, of the two images* brain1.pgm *and* brain2.pgm *be constructed?***

14. ***Find the difference between the "standard" brain and the image from the stroke patient (*brain3.pgm*). Where in the brain is the change located?***

Another form of arithmetic for images is letting a constant value affect the image. Try to add or subtract a constant from an image.

15. ***What happens when a pixel gets a value less than 0 or a value greater than 255? Are there other ways this can be handled?***

8 Geometric Transforms

It is sometimes necessary to geometrically correct images. In object recognition a first step can be to rotate the image so that the object is in a standard position, for example along the vertical axis. Open the image **wrench.pgm** and rotate it 20 degrees, with and without interpolation, using the Matlab commands:

```
J = imrotate(I,20);
K = imrotate(I,20,'bilinear');
```

where `I` is the wrench image.

16. ***Compare rotations performed with and without interpolation. It is easiest to see differences along lines and edges of the images. What does interpolation mean in this case?***

17. ***In general it is faster to rotate the image by a multiple of 90 degrees than by some arbitrary degree. Explain why.***

9 Fast Fourier Transform

Open the image **cameraman.pgm**. Fourier transform the image using FFT and display the Fourier domain by using the Matlab commands:

```
f = imread('cameraman.pgm');
F = fft2(double(f));
displayfft2(F);
```

This shows the FFT of the image. The `double()` command converts the image matrix from the `uint8` format (8-bit unsigned integer) to the `double` format (64-bit floating point). This is necessary since the function `fft2` does not operate on matrices in `uint8` format.

You will now filter the image by extracting the central disk of the FFT. This is done by first creating an ideal filter,

```
H = idealfft2filter(R,S);
displayfftfilter(H);
```

then we use the filter for filtering the FFT:

```
G = F.*H;
displayfft2(G);
```

where `R` is the filter radius ($0 \leq$ `R` ≤ 1) and `S` is the size of the filter (must be `size(F)`). The filtering is done by multiplying the image with the filter in the frequency domain. You have now filtered the image in the frequency domain. To get get your filtered image from the frequency domain you need to inverse transform your filtered image using the inverse FFT:

```
g = abs(ifft2(G));
```

The inverse FFT result contains complex values, therefore, the `abs()` function is used to get the magnitude of the complex values. Open the filtered result in **Image Tool**, but this time you need to pass an empty matrix as a second argument since the resulting image is in `double` format:

```
imtool(g, [])
```

The `[]` tells imtool to use the image's graylevel range (called dynamic range) as display range. If the image does not look filtered, try a different filter radius.

Repeat the above but use the a Butterworth filter and a Gaussian filter instead of the ideal filter using the commands:

```
H = butterworthfft2filter(R,S,N);
H = gaussianfft2filter(STD,S);
```

where `N` is the order of the Butterworth filter, and `STD` is the standard deviation of the Gaussian filter.

18. ***How do the results differ? What causes the differences?***

To create a high-pass (HP) filter it is necessary to obtain the complements of the created low-pass (LP) filters. This is done by simply inverting the filters above. This can be done in the filtering step, which then becomes:

```
G = F.*(1 - H);
```

Try a few different HP-filters and see if the results meet your expectations.

19. ***Which filters in the spatial domain correspond to LP- and HP-filters in the frequency domain, respectively?***

Open the image **freqdist.pgm**. There is a pattern present in the image that should be filtered out. To your aid you have a function which can create an ideal filter which is not necessarily centered:

```
H = idealfft2filterpos(R,S,X,Y);
```

It works just as the ordinary `idealfft2filter`, but `X` and `Y` specifies the disk center. Note that you can combine many ideal filters into one filter by using the `|` operator. An example:

```
H1 = idealfft2filterpos(...);
H2 = idealfft2filterpos(...);
H = H1 | H2;
```

20. ***Create a filter in the frequency-domain that suppresses the pattern in* freqdist.pgm*, but leaves the rest of the image as intact as possible. What does the filter look like? What do you see in the filtered image?***

That's it. The exercise is done!

21. ***Any comments on the exercise?***