Digital Image Processing: Session 1

Introduction

Human beings rely mainly on visual skills to identify and determine the environment. We have developed visual skills to quickly process information obtained from images. We have to take into account that our perception of those images is very personal and not always objective.

Image processing will change an image in order to:

1. improve its pictorial information for human interpretation

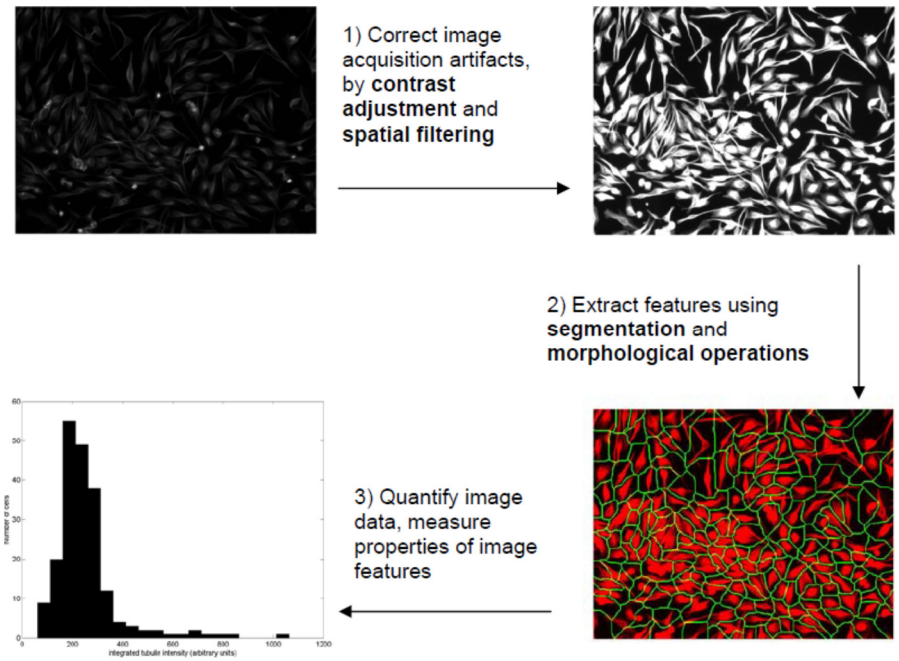
2. apply changes to enhance for autonomous machine interpretation.

Digital image processing will use a computer to change the appearance of a digital image. Changing a picture to make an image "look better" may be the worst decision to improve it for machine identification.

Image processing techniques

- Image enhancement: processing an image so that it is more suitable for a particular application.
  (sharpening or de-blurring an image, showing edges, improving image contrast, removing noise)
- Image restoration: reversing the damage done to an image
  (removing of blur caused by linear motion, removal of optical distortions)
- Image segmentation: subdividing an image into desired sections or isolating them
  (finding lines, circles, or particular shapes in an image)

For instance: the microscopy image above left shows a field of view of tissue-culture cells. One can ask: how many cells are there in this field of view? What is the average size? How much DNA is in each of the cells? (Image analysis for biology, HY Kueh)



Types of digital images

- **binary**: Each pixel is just black or white.
- **grayscale**: Each pixel is a shade of grey, normally from 0 (black) to 255 (white).
- **true color or RGB**: each pixel has a particular colour; that colour being described by the amount of red, green and blue in it. To transform a RGB-image to grayscale we can use the formula:

I = 0; 299R + 0; 587G + 0; 114B

You can of course use the matlab command RGB2gray to make the conversion
- **indexed**: the image has an associated colour map, which is simply a list of all the colours used in that image. Each pixel has a value which does not give its colour (as for an RGB image), but an index to the colour in the map.

Remember that digital images are represented using one or more matrices. **Vectorizing** your matrix operations will almost always replace loops which tend to slow Matlab down.

Use **Matlab help**! F.i. help imdemos

Conversion of images

Function	**Use**
ind2gray	indexed to grayscale
gray2ind	grayscale to indexed
rgb2gray	RGB to grayscale
gray2rgb	Grayscale to RGB
rgb2ind	RGB to indexed
ind2rgb	indexed to RGB
imbinarize	grayscale to binary

Function	Use	Format
ind2gray	Indexed to Greyscale	y=ind2gray(x,map);
gray2ind	Greyscale to indexed	[y,map]=gray2ind(x);
rgb2gray	RGB to greyscale	y=rgb2gray(x);
gray2rgb	Greyscale to RGB	y=gray2rgb(x);
rgb2ind	RGB to indexed	[y,map]=rgb2ind;
ind2rgb	Indexed to RGB	y=ind2rgb(x,map);

When using image conversions, make sure you understand the meaning of the different parameters.

Exercises

1. Example:

```
IM=imread('baboon.jpg');
imshow(IM);
impixelinfo
imageinfo
imfinfo('baboon.jpg')
size(IM)
```

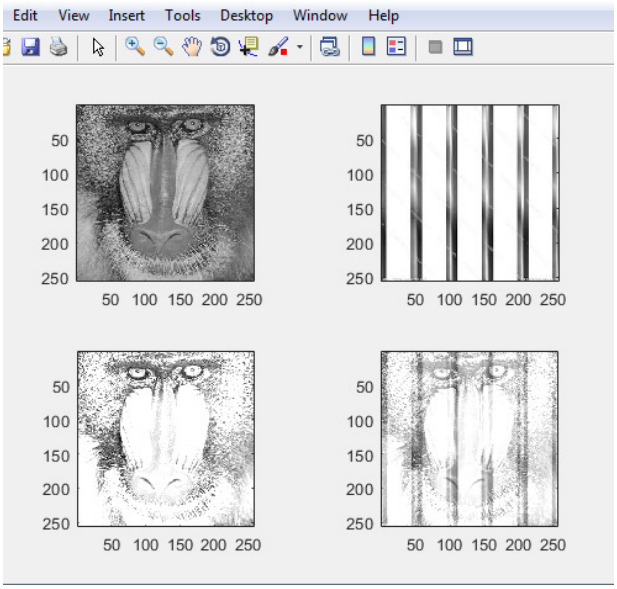
Try out these commands for the four different types of digital images and have a look at IM. Use the functions mentioned above to make the necessary conversions. Determine which commands are

suited for different tasks. Try to show an indexed image, using different colour maps. Have a look at values of the pixel (10,15) for each image.

A few operations on the grayscale image:
- what is the effect of following command?
  IM(10:50,30:150)=255
- create an image IM1 so all intensities of IM are halved
  create an image IM2 so all intensities of IM are tripled. Compare both images. What happens when the intensities exceed 255?

2. Create a 100x100 matrix M with random integer values in [0,255]. Transform to a second matrix N, so all values under 100 are converted to 0 and all other to 1. Use logic operations! Show both images (imshow). You have converted a grayscale image to binary. Use a matlab command to perform the same operation and compare (imshowpair).

3.
- load the image 'baboon.jpg' in IM1
- load the image 'jail.jpg' in IM2
- resize IM2 so both images have the same size
- subdivide (subplot) your graph window in 4 separate windows and show following images (subimage):



upper left: baboon in grayscale

upper right: jail in grayscale

bottom left: baboon in double intensity

bottom right: halve the intensity of jail and add to baboon. Why is the result brighter?

4. The command roiploy allows you to select part of your image and creates a "mask". This is a binary image with the same size as the original, allowing you to clip part of your original image.

Try out:

```
im1=imread('baboon.jpg');
im1=rgb2gray(im1);
imshow(im1);
masker=roipoly(im1);
figure;
imshow(masker);
```

Use your pointer to select the eyes of the baboon in a rectangle, double click to end

Exercise: use the images ‘moss.jpg’ and ‘squirrel.jpg’. Cut out the figure of the squirrel and insert it in the moss.

5. Make matrix X = [1 2 3;3 1 2;2 3 1]; X_map=[1 0 0;0 1 0;0 0 1];
Show the image using the following command:
imshow(X,X_map,'InitialMagnification','fit')
Try several colour maps.

6. Simple segmentation using a threshold:

Medical images often consist of illuminated objects in grayscale against a dark background. A very simple way to perform a segmentation, starting from an image f(x,y), uses a threshold T:

$g(x,y) = 1$ if $f(x,y) > T$

$g(x,y) = 0$ if $f(x,y) \leq T$

Use **imbinarize** to perform this operation.

6. Read the image ‘moon.tif’ and detect all craters with a circonference higher than a given value. Hint: look at the examples on help for **imbinarize** and also look up **regionprops.** Show the image with the detected circles.