Session 3: segmentation

Bit planes

Greyscale images can be transformed into a sequence of binary images by breaking them up into their bit-planes. Consider following examples, study what happens and try to find out how to recover the original image, using the different bit planes.

```
c=imread('trees.tif');
imshow(c);
cd=double(c);figure;imshow(cd);
c0=mod(cd,2);figure;imshow(c0);
c1=mod(floor(cd/2),2);figure;imshow(c1);
c2=mod(floor(cd/4),2);figure;imshow(c2);
c3=mod(floor(cd/8),2);figure;imshow(c3);
c4=mod(floor(cd/16),2);figure;imshow(c4);
c5=mod(floor(cd/32),2);figure;imshow(c5);
c6=mod(floor(cd/64),2);figure;imshow(c6);
c7=mod(floor(cd/128),2);figure;imshow(c7);
```

Use the same script for the images ‘staphylococco_aureo.tif’, ‘chest-xray.tif’ and ‘moon.tif’

Simple segmentation using a threshold

Medical images often result from illumination of objects against a black background. Segmentation of the image, using a threshold T, is a simple but effective method.

$$g(x,y) = 1 \text{ if } f(x,y) > T$$
$$= 0 \text{ if } f(x,y) \leq T$$

f(x,y) is transformed into the thresholded binary image g(x,y).

In Matlab, we can use the command: g = im2bw(f,T)

Histogram

The command imhist(f) enables us to inspect the histogram of the image f. Histogram uses 256 intervals as standard, but you can change that:

```
f = imread(‘chest-xray.tif’);
imhist(f);
figure;
imhist(f,20);
```

Checking the histogram, we can make an appropriate choice for the threshold value T. Afterwards we evaluate the effect of the contrast adjustments by looking at the histogram of the adapted image. As an example, we try out this technique using the image of the rice grains:

Execute following script and compare:

```
clear;
close all;
im=imread('rice.gif');
imshow(im);
figure;
imhist(im);
im=mat2gray(im);   % transform to set range to [0,1]
figure;
imshow(im);
figure;
imhist(im);
im=imread('rice.gif');
T=graythresh(im);  % matlab calculates the threshold value T (check matlab help)
im=im2bw(im,T);    % transforms into black white image according to threshold T
figure;
imshow(im);
figure;
imhist(im);
```

Try setting a different value for T.

T is calculated following the rule that the weighted distances between the average of the histogram and the averages of the back- and foreground pixels have to be maximized. This example uses a relatively simple image. Try out the same technique using 'rice.jpg'. The result will not be as good because of irregularities in the background. In later sessions we will see new image processing techniques to correct this problem.

Using the binary image 'rice.gif', we can have a look at the region properties of certain objects (such as area, diameter, …). An object within a binary image is a set of white pixels which are mutually connected. We can number the objects:

```
im=imread('rice.gif');
 [L,n]=bwlabel(im);
imtool(L);
```

L is the labelled image, n the number of objects. The last command is used to have a look at a number of pixel values. Adjust the contrast to see the range of intensity values in the image. Once the image has been labelled, use the **regionprops** command to obtain quantitative information about the objects:

```
D=regionprops(L,'P');
```

D is a structured array, P is the chosen property from the following list

'Area'	'EulerNumber'	'Orientation'
'BoundingBox'	'Extent'	'Perimeter'
'Centroid'	'Extrema'	'PixelIdxList'
'ConvexArea'	'FilledArea'	'PixelList'
'ConvexHull'	'FilledImage'	'Solidity'
'ConvexImage'	'Image'	'SubarrayIdx'
'Eccentricity'	'MajorAxisLength'	
'EquivDiameter'	'MinorAxisLength'	

TIP: Using structure arrays in MATLAB
Access an individual field within in the structure array by referring to its name after a dot (.) .
For instance:
>> D(1).Area

You can assign values to structure arrays using the assignment operator (=):
>> D(1).Area = 888; (n.b. we do not condone data manipulation)

And define new fields by using a new field name during assignment:
>> D(1).test = 1

To get an array of values from one field in the structure **D**, type:
>> w = [D.Area]

Try to understand following script:

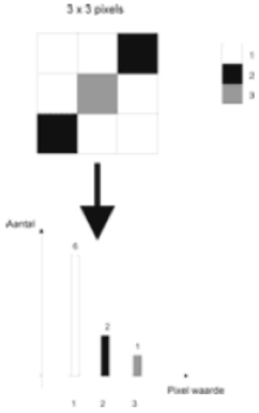
```
clear;
close all;
im=imread('moon.tif');
imshow(im);
[L,n]=bwlabel(im);
imtool(L);
D=regionprops(L,'Centroid');
centroids = cat(1, D.Centroid);
plot(centroids(:,1), 'b*')
D(1)
w=struct2array(D);
meancentroid=mean(w);
Stdcentroid=std(w);
hist(w)
```

What is the difference between the commands imhist and hist? Try to show the perimeter of the objects.

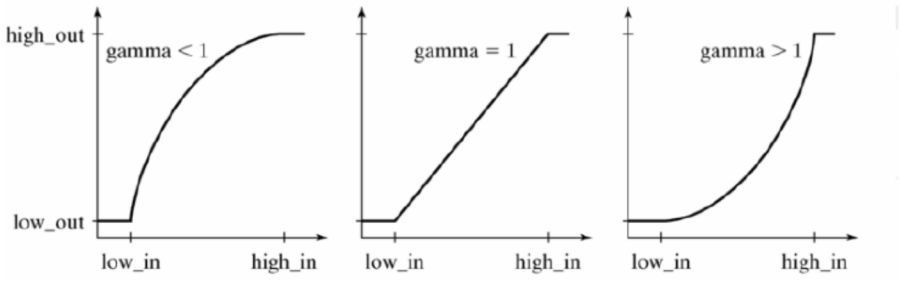
Adjusting the contrast of an image

Often image shave a limited dynamic range, which makes it difficult to detect certain image properties. We can use intensity transforms to enhance the image. This often results in higher performance for segmentation and feature recognition. Histogram determines for all colours of an intensity of an indexed image, how many
pixels within the image have that specified colour.

Looking at the histogram of a low contrast image, we notice that the larger part of the pixel values are situated in a small interval of intensities. Histogram stretching allows us to stretch this limited interval to the entire range [0,1]. The intensities according to the original range, are matched to intensities within [0,1] according to a specified transformation algorithm. For the histograms at the right, the transformation is linear.



Underneath we show the frequently used gamma contrast transform, for a number of possible gamma values:
low_in and **high_in** are the low and high grayscale intensity values which are used for the contrast transformation.



Matlab uses the function **imadjust**:

```
g=imadjust(f,[low_in high_in],[low_out high_out], gamma);
```

Using the function **stretchlim**, Matlab calculates the limits for you (check Help). Try following commands:

```
f=imread('chest-xray.tif');
imshow(f);
g1=imadjust(f);
imshow(g1);
g2=imadjust(f,[0,1],[1,0]);
figure,imshow(g2);
g3=imadjust(f,[0,0.2],[0,1]);
figure,imshow(g3);
g4=imadjust(f,[],[],0.2);
figure, imshow(g4);
```

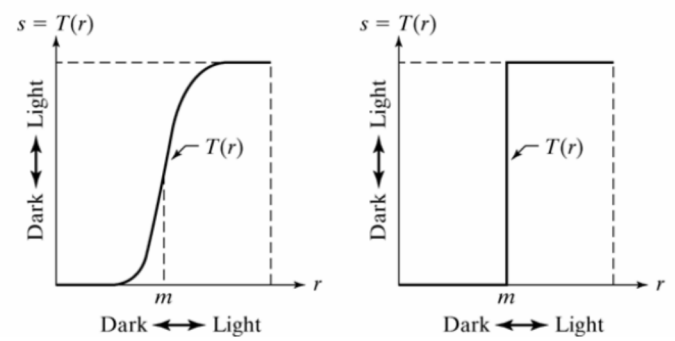
Try to explain what happens.
The negative of an image can be obtained:

```
g=imcomplement(f);
```

We can use another contrast stretching transform, which follows the formula (in Matlab):

```
g = 1./(1+(m./double(f)+eps)).^E)
```

Following images show this transform, which is determined by the threshold m and the steepness of the threshold E.



In the limiting case of E>>1, the image is limited by the intensity m.

The second important intensity transformation is the logarithmic transformation. In Matlab:

```
g=c*log(1+double(f))
```

We look at the effect of this transform:

```
f=imread('spectrum.tif');
imshow(f);
g=im2uint8(mat2gray(log(1+double(f))));
figure,imshow(g);
```

Exercises

1. Load the images 'pollen.tif' and 'bone-scan-GE.tif'. Show both images using imtool. Check the pixel values for a number of pixels. Transform the images to double and compare the corresponding pixel values.

2. Eliminate the "bad" (area below a certain limit) rice grains (incomplete, little specks, …) from 'rice.gif' using regionprops. You can look up the desired grains using **find** and delete them. Use a loop to check all objects. Compare the resulting image with the original. How many "bad" grains are left?

3. Write an m-file which shows the effect of the last contrast stretching transform for different values of m and E, applied to the image 'spectrum.tif'. As an extra try out the logarithmic transform and explain the differences.

4. Load the images 'pollen.tif' and 'bone-scan-GE.tif' and show them, using imtool. Transform the images using contrast stretching and also the logarithmic transform and show the results. Look into a few pixel values.