# INF 4300 2010
# Object feature extraction – shape descriptors
Exercises for Friday 23.10.2010

## **Exercise 1.** Matlab exercise for recognition of round objects

**Step 1: Read Image**

Read in `pills_etc.png`.

```
RGB = imread('pillsetc.png');
imshow(RGB);
```

**Step 2: Threshold the Image**

Convert the image to black and white in order to prepare for boundary tracing using `bwboundaries`.

```
I = rgb2gray(RGB);
threshold = graythresh(I);
bw = im2bw(I,threshold);
imshow(bw)
```

**Step 3: Remove the Noise**

Using morphology functions, remove pixels which do not belong to the objects of interest.

```
% remove all object containing fewer than 30 pixels
bw = bwareaopen(bw,30);
% fill a gap in the pen's cap
se = strel('disk',2);
bw = imclose(bw,se);
% fill any holes, so that regionprops can be used to estimate
% the area enclosed by each of the boundaries
bw = imfill(bw,'holes');
imshow(bw);
```

**Step 4: Find the Boundaries**

Concentrate only on the exterior boundaries. Option 'noholes' will accelerate the processing by preventing `bwboundaries` from searching for inner contours.

```
[B,L] = bwboundaries(bw,'noholes');
% Display the label matrix and draw each boundary
imshow(label2rgb(L, @jet, [.5 .5 .5]))
hold on
for k = 1:length(B)
  boundary = B{k};
  plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)
end
```

**Step 5: Determine which Objects are Round**

Estimate each object's area and perimeter. Use these results to form a simple metric indicating the roundness of an object:

`metric = 4*pi*area/perimeter^2.`

This metric is equal to one only for a circle and it is less than one for any other shape. The discrimination process can be controlled by setting an appropriate threshold. In this example use a threshold of 0.94 so that only the pills will be classified as round.

Use `regionprops` to obtain estimates of the area for all of the objects. Notice that the label matrix returned by `bwboundaries` can be reused by `regionprops`.

```
stats = regionprops(L,'Area','Centroid');
threshold = 0.94;

% loop over the boundaries
for k = 1:length(B)
```

```
      % obtain (X,Y) boundary coordinates corresponding to label 'k'
      boundary = B{k};
      % compute a simple estimate of the object's perimeter
      delta_sq = diff(boundary).^2;
      perimeter = sum(sqrt(sum(delta_sq,2)));
      % obtain the area calculation corresponding to label 'k'
      area = stats(k).Area;
      % compute the roundness metric
      metric = 4*pi*area/perimeter^2;
      % display the results
      metric_string = sprintf('%2.2f',metric);
      % mark objects above the threshold with a black circle
      if metric > threshold
            centroid = stats(k).Centroid;
            plot(centroid(1),centroid(2),'ko');
      end
      text(boundary(1,2)-35,boundary(1,1)+13,metric_string,'Color','y',...
      'FontSize',14,'FontWeight','bold');
end
title(['Metrics closer to 1 indicate that ',...
'the object is approximately round']);
```

## **Exercise 2.** Matlab exercise for selecting grains with a given orientation

**Step 1: Read Image**

```
I = imread('rice.png');
imshow(I)
```

**Step 2: Use Morphological Opening to Estimate the Background**
Notice that the background illumination is brighter in the center of the image than at the bottom. Use the IMOPEN function to estimate the background illumination.

```
background = imopen(I,strel('disk',15));
% Display the Background Approximation as a Surface
figure, surf(double(background(1:8:end,1:8:end))),zlim([0 255]);
set(gca,'ydir','reverse');
```

**Step 3: Subtract the Backround Image from the Original Image**
Since the image and background are of class uint8, use the function IMSUBTRACT to subtract the background.

```
I2 = imsubtract(I,background);
imshow(I2)
```

**Step 4: Increase the Image Contrast**

```
I3 = imadjust(I2);
imshow(I3);
```

**Step 5: Threshold the Image**
Create a new binary image by thresholding the adjusted image.

```
level = graythresh(I3);
bw = im2bw(I3,level);
imshow(bw)
```

**Step 6: Label Objects in the Image**
The function BWLABEL labels all connected component in the binary image. The accuracy of your results depend on: the size of the objects, the accuracy of your approximated background, whether you set the connectivity parameter to 4 or 8, whether or not any objects are touching (in which case they may be labeled as one object). Some of the rice grains are touching.

```
[labeled,numObjects] = bwlabel(bw,4); % 4-connected
numObjects % Count all distinct objects in the image.
```

**Step 7: Examine the Label Matrix**

Each distinct object is labeled with the same integer value. Crop part of a grain that is labeled with 50s.

```
rect = [105 125 10 10];
grain = imcrop(labeled,rect) % Crop a portion of labeled
```

**Step 8: View the Whole Label Matrix**

A good way to view a label matrix is to display it as a pseudo-color indexed image. In the pseudo-color image, the number that identifies each object in the label matrix maps to a different color in the associated colormap matrix. Use LABEL2RGB to choose the colormap, the background color, and how objects in the label matrix map to colors in the colormap.

```
RGB_label = label2rgb(labeled, @spring, 'c', 'shuffle');
imshow(RGB_label)
```

**Step 9: Measure Object Properties in the Image**

The REGIONPROPS command measures object or region properties in an image and returns them in a structure array. When applied to an image with labeled components, it creates one structure element for each component. Use regionprops to create a structure array containing some basic properties for the labeled image.

```
graindata = regionprops(labeled,'basic')
% To find the area of the component labeled with 50s, use dot notation to
% access the Area field in the 50th element in the graindata structure
% array.
graindata(50).Area
```

**Step 10: Compute Statistical Properties of Objects in the Image**

Create a new vector allgrains which holds the area measurement for each grain

```
allgrains = [graindata.Area];
max_area = max(allgrains) % Find the maximum area of all the grains.
biggrain = find(allgrains==max_area) % Find the grain number that has this a
rea.
mean(allgrains) % Find the mean of the area of all the grains.
```

**Step 11: Create a Histogram of the Area of the Grains**

```
nbins = 20;
figure,hist(allgrains,nbins)
```

**Step 12: Measure the orientation of the Grains**

Now, measure the orientation of each grain
Find out how yourself ☺

**Step 13: Select the grains with orientation within 0 and +/-20 degrees**

Do it by yourself!

Good Luck!