Uppsala University

Computer-Assisted Image Analysis I

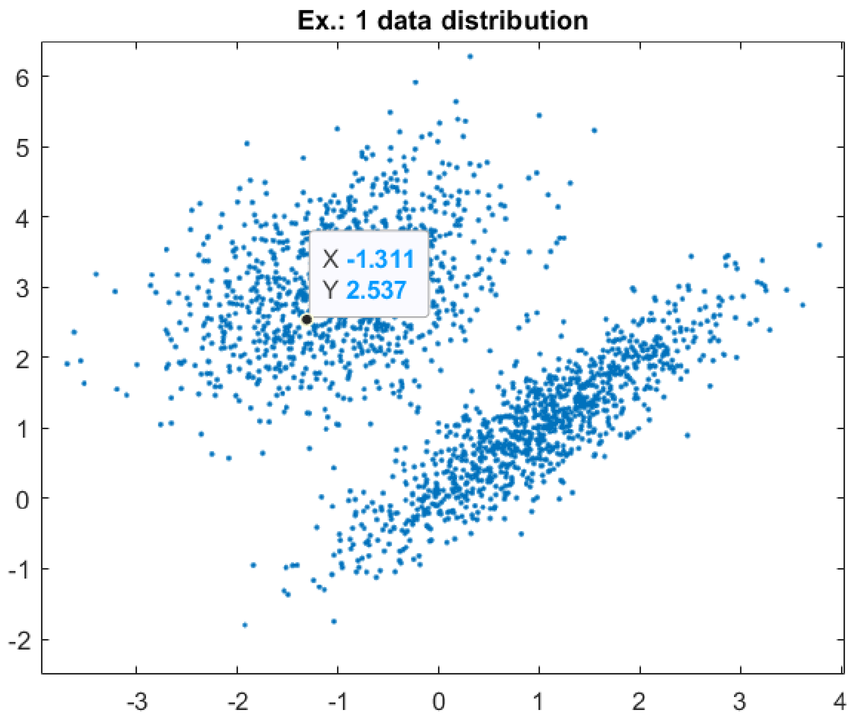
Lab 4

Authors:
Mareen Allgaier
Oskar Tegby

December 13, 2019

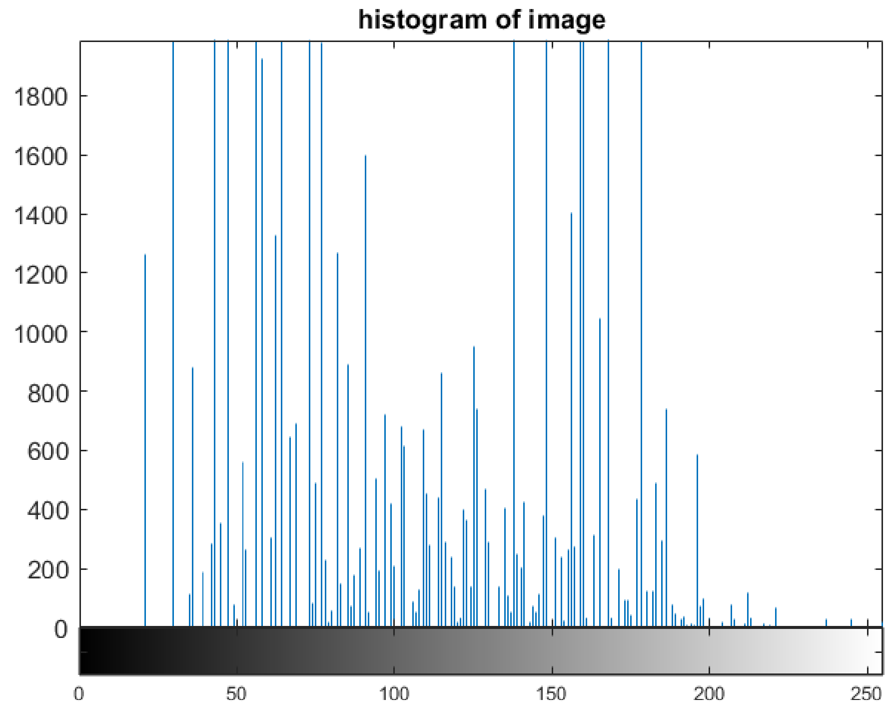
Task 1

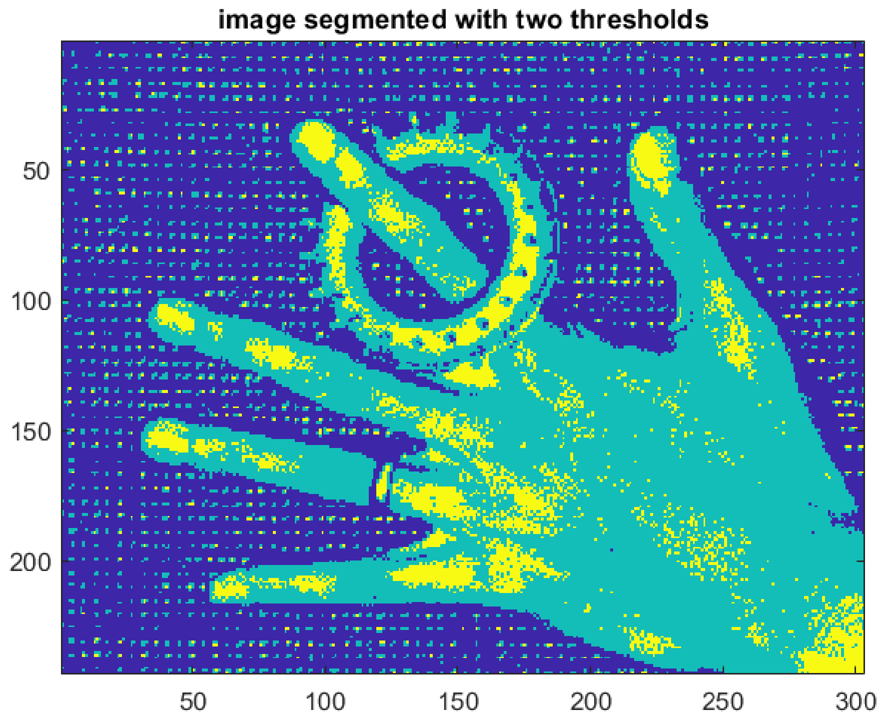
Neither of them would be a good discriminative feature, as seen in the picture below. A diagonal line (so both coordinates) are necessary.



Task 2

No, because the histogram (shown below) is not tri-modal. The object and the hand have nearly the same greyvalues, so there is no threshold to separate them. The segmentation result can be seen in the image below.


histogram of image
1800
1600
1400
1200
1000
800
600
400
200
0
0
50
100
150
200
250

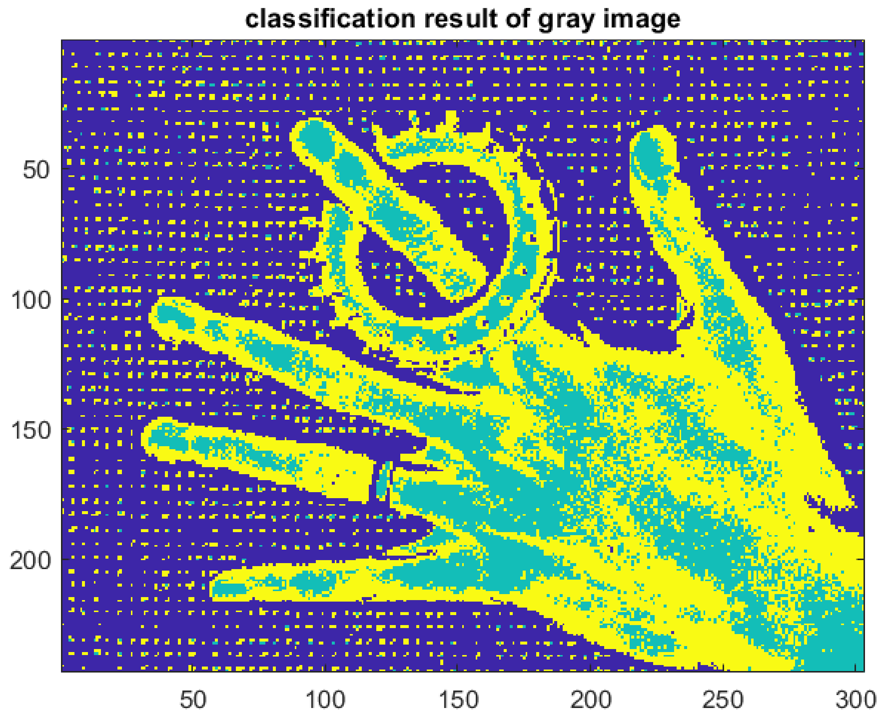


Task 3

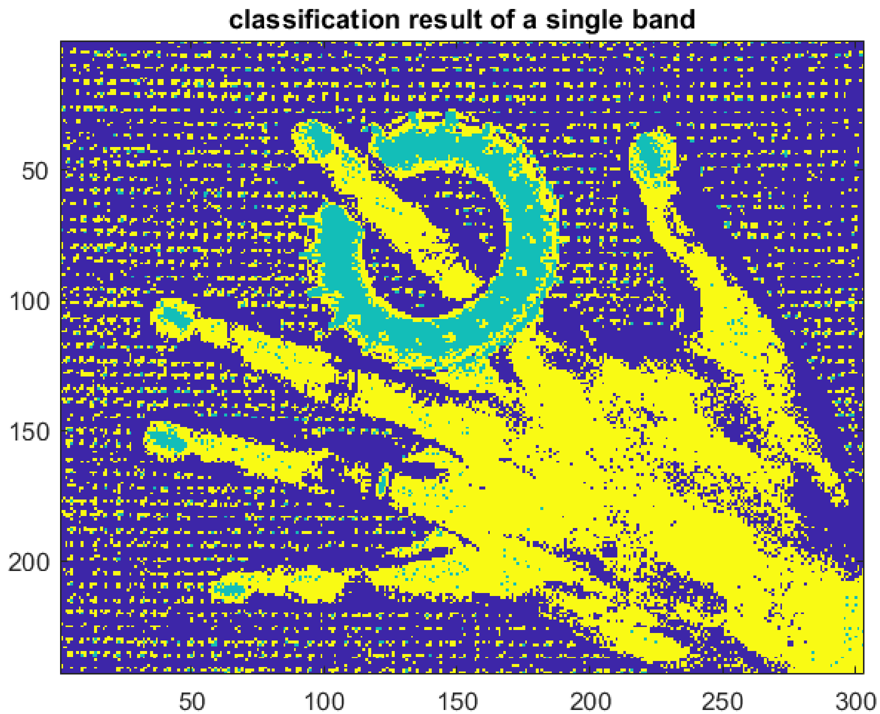
The assumption is, that the data can be separated by a linear function (line)

Task 4

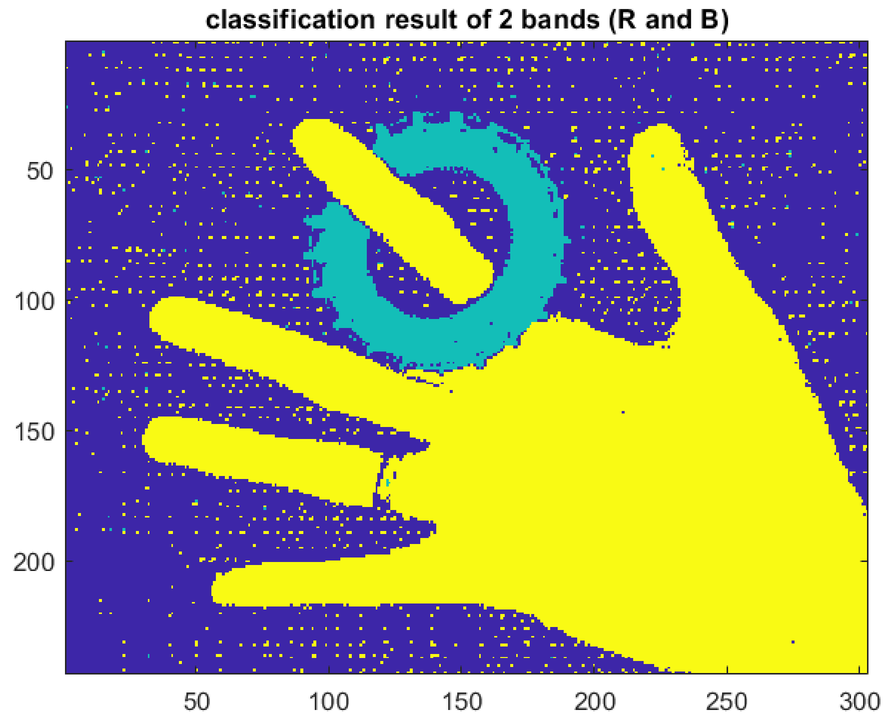
The results did not improve by using the classification on the greyscale image.



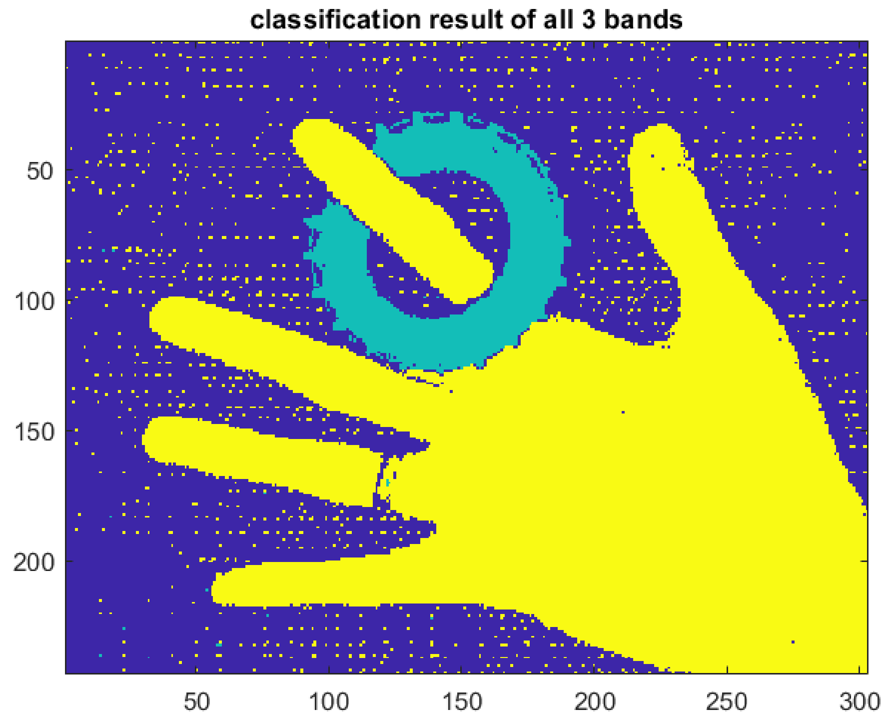
The result of the single band classification is better than the greyscale result, because the blue channel is used and with this feature the object and the hand can be separated. The single band is more successful than grayscale, as the hand and the object have same grayvalues, but different rgb values (object blue, hand more red), so the single band is easier to discriminate.



The classification can be improved even more by using two bands. The red and blue channels are used as the discriminative features.

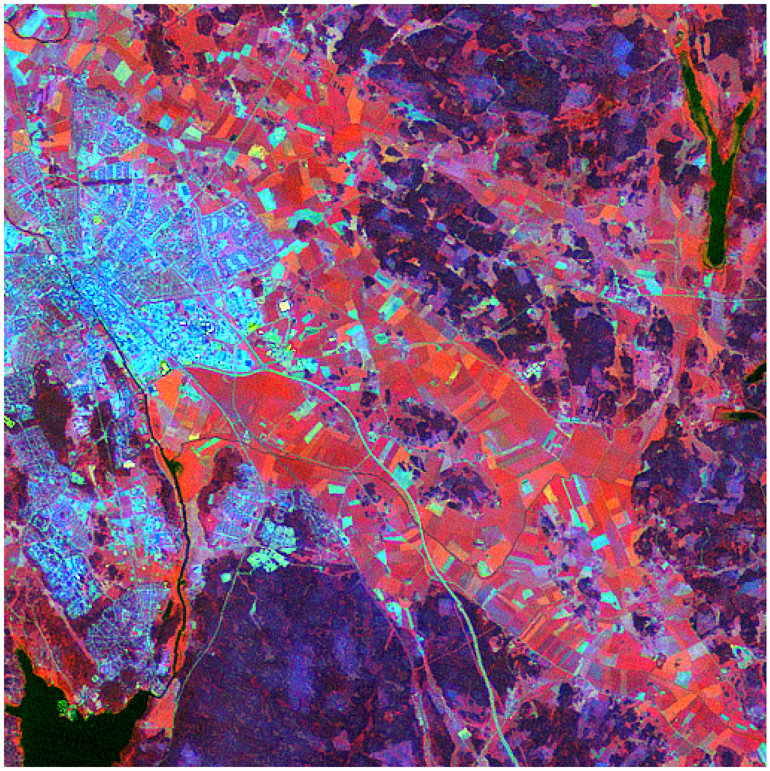


The image below shows the result using all three bands.



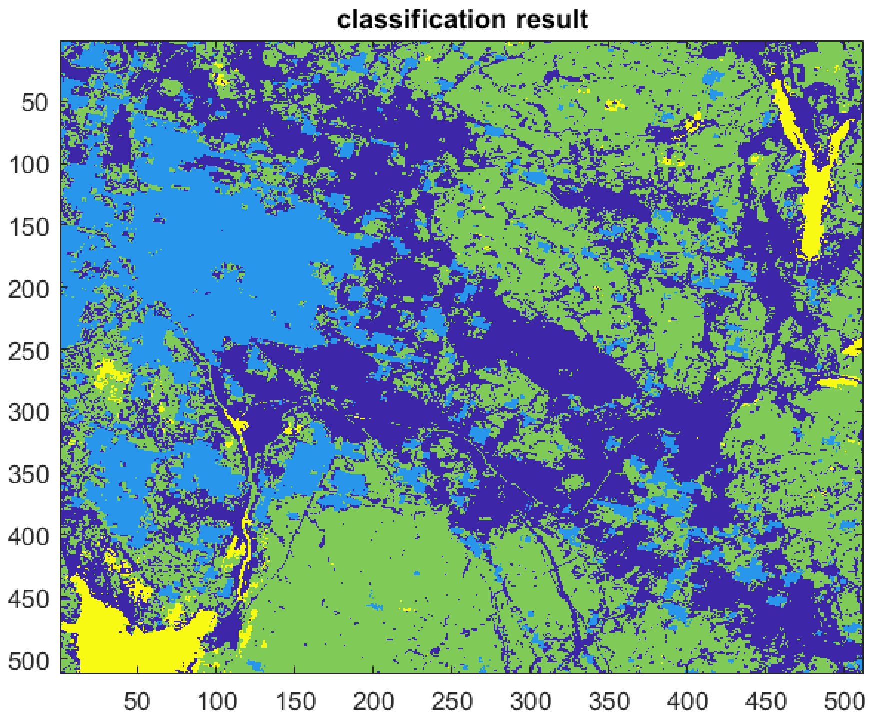
Task 5

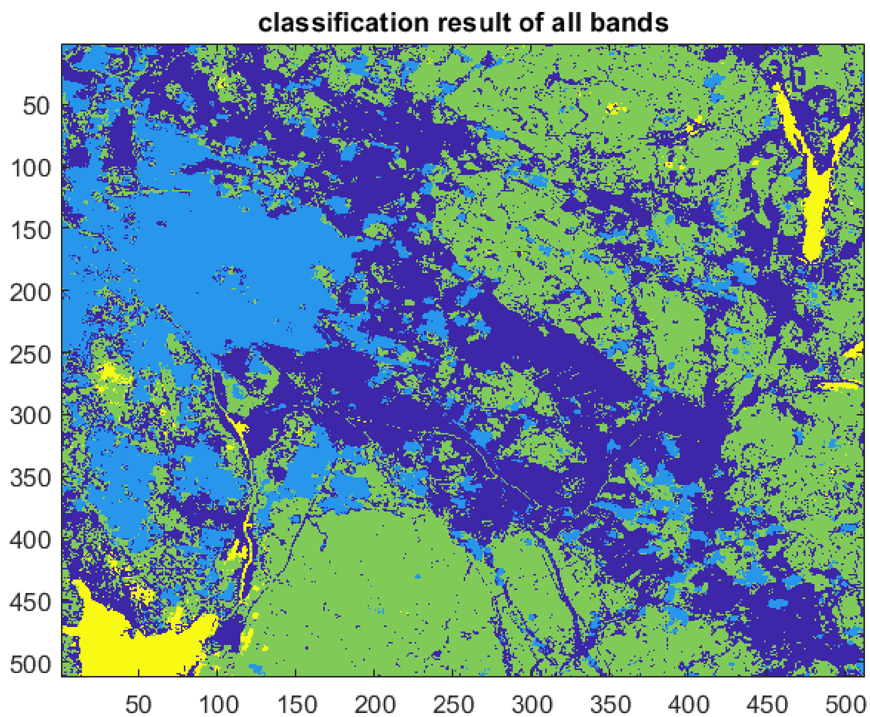
We are using bands 4, 1, and 6, as they are good to discriminate the fields, forests, buildings and lakes.



The image below shows the feature vectors. Therefore we are using the quadratic classifier, as lines could not separate that well.

The next two images show our classification resuls with the selected bands and with all bands. There is not really a benefit/improvment using all bands.


classification result
50
100
150
200
250
300
350
400
450
500
50
100
150
200
250
300
350
400
450
500



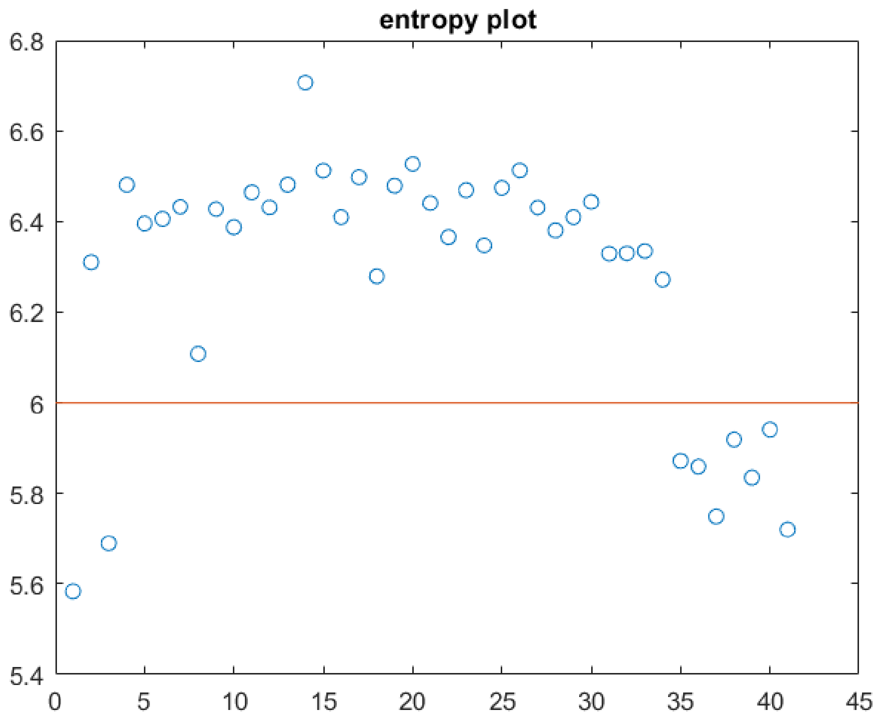
Task 6

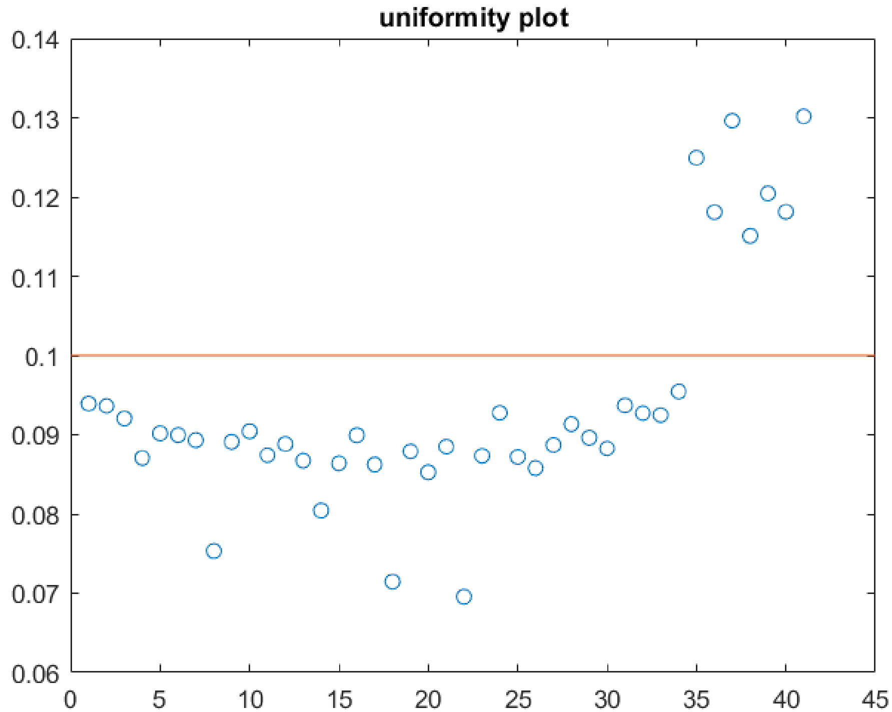
No, it will be slower and it tends to overfit.

Task 7

We have chosen an offset of one pixel in each direction (4 neighborhoods), because the real viruses have small areas with same greyvalues. And thus the cooccurence matrices of real viruses nearly only have (high) values on the diagonal, whereas the non-viruses have quite a few more values next to the diagonal.

The two plots below show the entropy and uniformity of the cutouts.


entropy plot
6.8
6.6
6.4
6.2
6
5.8
5.6
5.4
0
5
10
15
20
25
30
35
40
45



The entropy is a statistical measure of randomness that can be used to characterize the texture of the input image. Some kind of "amount of uncertainty"/measure of variability. So noise has for example a low entropy. The uniformity is maximum for images with one grayvalue So real viruses should have a higher entropy and a lower uniformity. In both plots the red line separates the outliers/non-viruses from the others/viruses.

Appendix

In the subsections below, the code for each exercise is written with proper comments.

Exercise 1 and 2

```
load('cdata');
figure(1);plot(cdata(:,1),cdata(:,2),'.'); title('Ex.:
   1 data distribution');
```

```

%1: Are the two classes possible to separate by using either the x or y coordinate as a
%single feature? Explain why or why not.
%-> neither of them would be a good discriminative feature


%2. Does multiple thresholding give a successful classification of the three classes?
%Explain why or why not.
I = imread('handBW.pnm'); % Read the image
figure(2);imshow(I); title('original image'); % Show the image
figure(3);imhist(I); title('histogram of image'); % Show the histogram

t1= 115;
t2= 171;
figure(4);mtresh(I,t1,t2); title('image segmented with two thresholds');
%-> no, because the histogram in not really trimodal. The object and the
%hand have nearly the same greyvalues
```

Exercise 3 and 4

```
I2 = imread('hand.pnm'); % Read the image
figure(1);imshow(I2); % Show the image
R = I2(:,:,1); % Separate the three layers, RGB
G = I2(:,:,2);
B = I2(:,:,3);
figure(2);plot3(R(:),G(:),B(:),'.'), title('3D scatterplot of the RGB data') % 3D scatterplot of the RGB data
figure(3),
subplot(2,2,1), imhist(I2(:,:,1)), title('R histograms');
```

```
subplot(2,2,2), imhist(I2(:,:,2)),  title('G histograms');
subplot(2,2,3), imhist(I2(:,:,3)),  title('B histograms');

label_im = imread('hand_training.png'); % Read image with labels
figure(4);imagesc(label_im), title('View the training areas'); % View the training areas

I3(:,:,1) = R; % Create an image with two bands/features
I3(:,:,2) = B;
[data,class] = create_training_data(I3,label_im); % Arrange the training data into vectors
%figure(5);scatterplot2D(data,class), title('View the training feature vectors'); % View the training feature vectors

Itest = im2testdata(I3); % Reshape the image before classification
C = classify(double(Itest),double(data),double(class)); % Train classifier and classify the data
ImC = class2im(C,size(I3,1),size(I3,2)); % Reshape the classification to an image
figure(6);imagesc(ImC), title('classification result of 2 bands (R and B)'); % View the classification result



% What sort of classification is used? You can use a number of different discriminant
% functions. The default is 'linear'. What assumptions are made by the classifier in
% this case?
%-> classifies the data based on the specified discriminant function ->
```

```
%unsupervised, no user-knowledge used, cluster according to color, then
%apply knowledge?
%the assumption is, that the data can be separated by a linear function (line)

%classification on grayscale image
gray = rgb2gray(I2);
[data_gray,class_gray] = create_training_data(gray,label_im); % Arrange the training data into vectors
Itest_gray = im2testdata(gray); % Reshape the image before classification
C_gray = classify(double(Itest_gray),double(data_gray),double(class_gray)); % Train classifier and classify the data
ImC_gray = class2im(C_gray,size(gray,1),size(gray,2)); % Reshape the classification to an image
figure(7);imagesc(ImC_gray), title('classification result of gray image'); % View the classification result


%Single bands from the RGB image
single_band = I2(:,:,3);
[data_single_band,class_single_band] = create_training_data(single_band,label_im); % Arrange the training data into vectors
Itest_single_band = im2testdata(single_band); % Reshape the image before classification
C_single_band = classify(double(Itest_single_band),double(data_single_band),double(class_single_band)); % Train classifier and classify the data
ImC_single_band = class2im(C_single_band,size(single_band,1),size(single_band,2)); % Reshape the classification to an image
figure(8);imagesc(ImC_single_band), title('classification result of a single band'); % View the classification result
```

```
%All three bands in the RGB image
all_bands = I2;
[data_all_bands,class_all_bands] = create_training_data(all_bands,label_im); % Arrange the training data into vectors
Itest_all_bands = im2testdata(all_bands); % Reshape the image before classification
C_all_bands = classify(double(Itest_all_bands),double(data_all_bands),double(class_all_bands)); % Train classifier and classify the data
ImC_all_bands = class2im(C_all_bands,size(all_bands,1),size(all_bands,2)); % Reshape the classification to an image
figure(9);imagesc(ImC_all_bands), title('classification result of all 3 bands'); % View the classification result

% 4: Have the results improved using classification compared to thresholding? Is the classi-
% fication more successful in the case with the grayscale image or single bands? Explain.
% Does it improve the classification to incorporate pairs of bands or the full RGB infor-
% mation? Discuss. Show your results from grayscale classification, one pair of features
% and full RGB classification.
%-> neither works
%-> single band is more successful than grayscale, as the hand and the
%object have same grayvalues, but different rgb values (object blue, hand
%more red), so the single band is easier to discriminate
%-> no difference between full RGB and RB, as RB are the main
%discriminative features
```

Exercise 5

```
load('landsat_data');
%[data,class] = create_training_data(I,label_im); % Arrange the training data into vectors
figure(1);imshow(landsat_data(:,:,[4,1,6])./255); % 4,1,3
R = landsat_data(:,:,1)./255; % Separate the three layers, RGB
G = landsat_data(:,:,2)./255;
B = landsat_data(:,:,3)./255;
% figure(2), imshow(R);
% figure(3), imshow(G);
% figure(4), imshow(B);

T = zeros(512,512); % Create an empty image
T(20:30,60:90) = 1; % Class 1
T(140:150,100:160) = 2; % Class 2
T(495:500,420:440) = 3;
T(470:490,20:45) = 4; %Lakes
%...

I_bands = landsat_data(:,:,[4,1,6])./255; % Create an image with specific bands
% I_bands(:,:,2) = G;
% I_bands(:,:,3) = B;
band_image = I_bands;

figure(5);imagesc(T), title('View the training areas'); % View the training areas
[data_,class_] = create_training_data(I_bands,T);
figure(6);scatterplot2D(data_,class_), title('View the training feature vectors'); % View the training feature vectors
%figure(6);plot3(R(:),G(:),B(:),'.'), title('3D scatterplot of the RGB data') % 3D scatterplot of the RGB data

```

```
%classification
[data_,class_] = create_training_data(band_image,T); % Arrange the training data into vectors
Itest_ = im2testdata(band_image); % Reshape the image before classification
C_ = classify(double(Itest_),double(data_),double(class_), 'quadratic'); % Train classifier and classify the data
ImC_ = class2im(C_,size(band_image,1),size(band_image,2)); % Reshape the classification to an image
figure(7);imagesc(ImC_), title('classification result'); % View the classification result


%classification with all bands
I_all_bands = landsat_data(:,:,:)./255; % Create an image with specific bands

[data_all_bands,class_all_bands] = create_training_data(I_all_bands,T); % Arrange the training data into vectors
Itest_all_bands = im2testdata(I_all_bands); % Reshape the image before classification
C_all_bands = classify(double(Itest_all_bands),double(data_all_bands),double(class_all_bands), 'quadratic'); % Train classifier and classify the data
ImC_all_bands = class2im(C_all_bands,size(I_all_bands,1),size(I_all_bands,2)); % Reshape the classification to an image
figure(8);imagesc(ImC_all_bands), title('classification result of all bands'); % View the classification result

%Ex. 6: Is there any reason not to include all bands if they are not needed to separate the
%classes?
% -> no, it will be slower and it tends to overfit
```

Exercise 6, 7

```
%% Read image and template from disk

% Read image
I = imread('viruses.tif');

% Read template
template = imread('virusTemplate.tif');

% Show image and template
figure('name','Original image');imshow(I);
figure('name','Template');imshow(template);

%% Template matching

% Select step length (How many pixels the template is moved in each
% iteration)
stepSize = 10;

% Do the template matching
ccimg = templatematching(I,template,stepSize);

% Show the resulting correlation image
%figure('name','correlation coefficients');imshow(ccimg,[]);colormap(copper);colorbar;


%% Find localmaxima in the correlation image

maxima = imextendedmax(ccimg,0,4);

% Shrink objects to points
maxima = bwmorph(maxima,'shrink',inf);

% Calculate the correlation coefficients for the maxima
maxvals = maxima .* ccimg;
```

```matlab

% Pick the corresponding correlation values and threshold them sung Otsu's
% method.
h = hist(maxvals(maxvals > 0),128);
h2 = imfilter(h,[1 1 1 1 1] ./ 5);
thresh = graythresh(h2);

% Create a binary image keeping only those with a correlation value that
% lies above the threshold.
maxvals(maxvals < thresh) = 0;
maxvals(maxvals ~= 0) = 1;

% Label the binary image and obtain their positions
maxlbl = logical(maxvals);
maxcentroids = regionprops(maxlbl,'centroid');

% Create a circular mask which will be used to cutout individual objects
% from the original image.
[h,w] = size(template);
d = max([h w]);
mask = imcircle(d);

% Cut out all objects using the mask and store it in a cell array
% named "cutouts".
nrOfObjects = length(maxcentroids);
cutouts{nrOfObjects} = [];
figure('name','Segmentation result');imshow(I);hold on;
for i = 1 : nrOfObjects
    % The position corresponding to the centroid has to be calculated since
    % the centroid position depends on the step size (stepSize) used.
    realpos = [((maxcentroids(i).Centroid(1)-1)*stepSize+1) ((maxcentroids(i).Centroid(2)-1)*
```

```
        stepSize+1)];

    cutouts{i} = I(realpos(2):realpos(2)+d-1,realpos(1):realpos(1)+d-1);
    cutouts{i} = cutouts{i} .* uint8(mask);


    rectangle('Position',[realpos(1),realpos(2),d,d],...
        'Curvature',[1,1],...
        'edgecolor', 'w',...
        'linewidth', 1 );
end
hold off;

%% Adding one noisy disk and one gray disk

%refimg = noise(100 .* ones(size(mask)),'mu',1);
%cutouts{1,end+1} = uint8(refimg.* mask);
%cutouts{1,end+1} = uint8(mask).*128;

%% Collect all cutouts in an image

imagesPerRow = 10;
cols = imagesPerRow;
rows = ceil(size(cutouts,2) / cols);

objectMap = uint8(zeros(size(mask).*[rows cols]));
counter = 1;

for i = 1 : rows
    for j = 1 : cols
        if counter <= size(cutouts,2)
            objectMap((i-1)*d+1:(i-1)*d+d,(j-1)*d+1:(j-1)*d+d) = cutouts{counter}(:,:);
            counter = counter + 1;
        end
    end
```

```
end

figure('name','Cutouts of segmented objects');imshow(objectMap);

%% cleaning the workspace from surplus variables
keep('I','template','mask','cutouts','nrOfObjects');%'objectMap');

%% Texture analysis
%
% I              - original image
% template       - the template used in the template matching procedure
% cutouts        - a cell array with all the segmented objects
% nrOfObjects    - the number of objects in 'cutouts'
% mask           - the binary mask used to cutout the objects
% objectMap      - a compiled image of all the objects in 'cutouts'
%


% Place for your own texture analysis.

%graycomatrix: by calculating how often a pixel with gray-level (grayscale intensity)
% value i occurs horizontally (depending on the offset) adjacent to a pixel with the value j
%Offset: Distance between the pixel of interest and its neighbor
%Idea of offset: viruses have small dots -> small areas with same values ->
%so in virus images the pixels tend to have similar neighbors -> can be
%seen in the matrixes where the viruses have a better distribution only on
```

```
%the diagonal
offset = 1;
offsets = [0 offset; -offset offset;-offset 0;-offset -offset; 0 -offset; offset -offset; offset 0; offset offset]; %0[U+FFFD],45[U+FFFD],90[U+FFFD],135[U+FFFD],180[U+FFFD],225[U+FFFD],270[U+FFFD],315[U+FFFD]
[glcms,SI] = graycomatrix(I,'Offset',offsets);
figure('name','gray-level co-occurrence matrix '); imshow(rescale(SI));

%Entropy: Entropy is a statistical measure of randomness that can be used to characterize the texture of the input image.
% some kind of "amount of uncertainty"/measure of variability. So noise has
% for example a low entropy.
%Uniformity: maximum for images with one grayvalue -> non-viruses are more uniform

%=> real viruses should have a higher entropy and a lower uniformity

entropy_cutouts = zeros(1,nrOfObjects);
glcms_cutouts{nrOfObjects} = [];
uniformity_cutouts = zeros(1,nrOfObjects);
for i=1:nrOfObjects
    %calculating the glcms of the cutouts
    [glcms,SI] = graycomatrix(cutouts{i},'Offset',offsets);
    glcms_cutouts{i} = glcms;

    %calculating entropy of the cutouts
    entropy_cutouts(1,i) = entropy(cutouts{i});

    %calculating uniformity of the cutouts
    h_cutout = histogram(cutouts{i},'Normalization', 'probability');
    uniformity = 0;
```

```
    for j=1:h_cutout.NumBins
        uniformity = uniformity + (h_cutout.Values(1,j)
            *h_cutout.Values(1,j));
    end
    uniformity_cutouts(1,i) = uniformity;

end

figure, plot(entropy_cutouts, 'o'), title('entropy plot
    '); hold on
x= 0:45;
plot(x, 6*ones(size(x)));
figure, plot(uniformity_cutouts, 'o'), title('
    uniformity plot');hold on
plot(x, 0.1*ones(size(x)));
```