Image Analysis - FMAN20
Assignment 4

Graph Cuts for Segmentation
Computer Vision
Final OCR - System Construction and Testing

Hicham Mohamad
hsmo@kth.se

January 28, 2019

1 Introduction

In this assignment we will study *graph cuts for segmentation*, solve a computer vision problem, and make our final version of our own Optical Character Recognition system.

2 Find the errors

In this section, we go through designing a method for automatic `detection` of five distinct differences between two images shown in Figure 1, which depict the same scene apart a slight example of `image registration`.

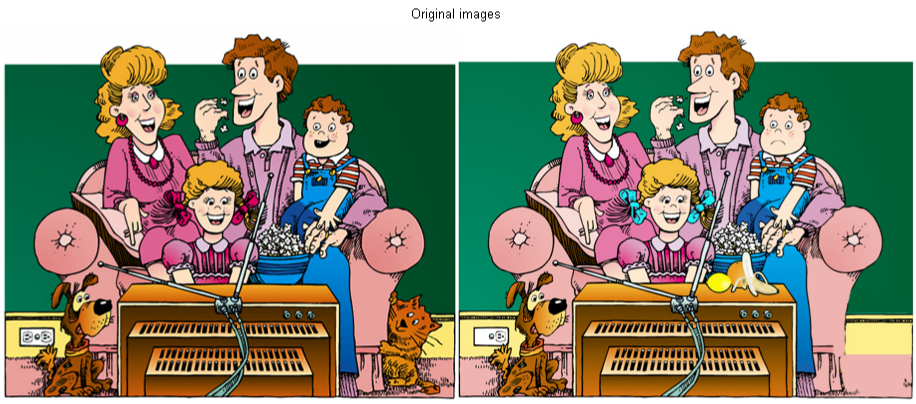


Figure 1: Two images depict the same scen. However there are five distinct differences between them

2.1 Image registration

As we can notice, the two images are out of register, i.e. they are translated with respect to each other. To determine how much one image is translated with respect to the other, one can perform a cross-correlation on the two images. MATLAB has a command for performing a normalized cross-correlation: `b = normxcorr2(im1,im2);` and to visualize the cross-correlation function using the mesh command: `figure; mesh(b);`

The output of this cross-correlation function is a matrix b that is the sum of the size of the two input images. b has a single maximum, which is `offset` from the center of the matrix by a small amount. This offset corresponds to the translation of image 1 with respect to image 2. To determine the location of this peak, we use the `find()` command, which returns the locations of nonzero indices in a given matrix. We can couple this command to conditional statement to find the maximum point of this matrix:

```
[y,x] = find(b == max(b(:)));
```

According to [12], We can perform intensity-based `image registration` with the following steps and the Matlab script is shown in Listing 1:

1. Read the images into the workspace with imread or dicomread.
2. Create the optimizer and metric. See Create an Optimizer and Metric for Intensity-Based Image Registration.
3. Register the images with imregister.
4. View the results with imshowpair or save a copy of an image showing the results with imfuse.

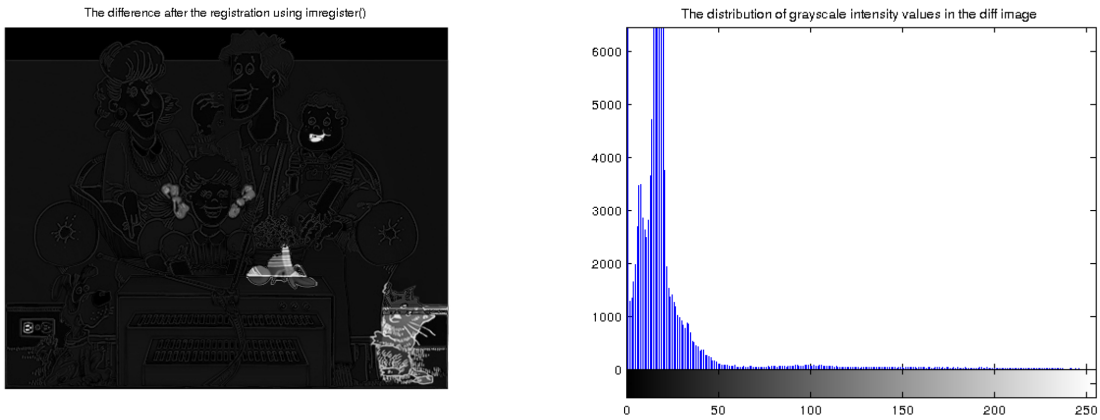
Here in Figure 2, we can see the detected differences marked with purple color after performing image registration using the Matlab function `imregister()`.

Figure 2: The results after performing the image registration. there are five distinct differences between the two images shown in purple color.

2.2 Difference image

It is often possible to visualizing dynamics in image stacks by performing mathematical manipulations. Difference images highlight features of the image that change, much in the same way that a `spatial gradient` enhances contrast around an edge.

In the detection and recognition of objects, the input image is often simplified by generating an output image whose pixels tend to have high values if they are part of an `object of interest` and low value if they are not part of any object of interest. The simplest way to identify these object regions is to perform a `thresholding` operation. Our difference image is illustrated in Figure 3a



(a) (b)

Figure 3: a) Plot of the difference between the two images . b) Histogram of the gray difference of images.

2.3 Thresholding

Thresholding is a `labeling` operation on a gray scale image. Thresholding distinguishes pixels that have higher gray values from pixels that have lower gray values. Pixels whose gray values are high enough are given the binary value 1. Pixels whose gray values are not high enough are given the binary value 0, as shown in Figure 4a.

The question of thresholding is to determine the `threshold value`. Since the threshold value seperates the dark background from the bright object or vice versa, the separation could ideally be done if the distribution of dark pixels were known and the distribution of bright pixels were known. The threshold value could then be determined as that `separation value` for which the fraction of dark pixels labeled binary 1 equals the fraction of bright pixels labeled binary 0. Such a threshold value would equalize the probability of the two kinds of errors: the error of assigning a pixel belonging to the `background` as a binary 1 and the error of assigning a pixel belonging to the `object` as a binary 0.

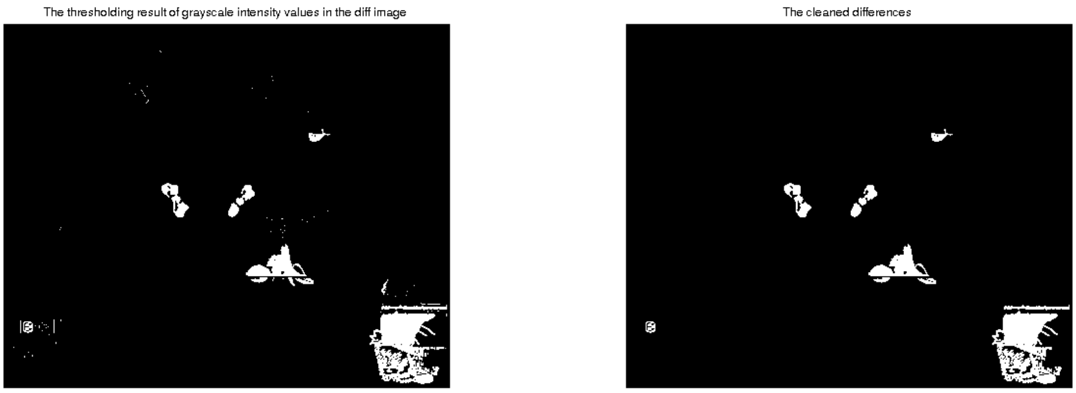
Using the image `histogram`, it is possible to know how many pixels are associated with any gray value in the mixture distribution. The histogram h of a digital image I is defined in [3] by

$$h(m) = \#\{(r,c)| \quad I(r,c) = m\}$$

If the distributions of dark pixels and bright pixels are widely separated, then the image histogram will be `bimodal`, one mode corresponding to the dark pixels and one mode corresponding to the bright pixels. With little distribution `overlap`, the threshold value is easily chosen as a value in the valley between the two dominant histogram modes. This is illustrated in Figure 3b, where the thresholding level 55 is chosen. However, when there is substantial overlap, the choice of the treshold as the valley point is less optimal in the sense of minimizing `classification error`.

2.4 Binary image and Area Opening

For better detecting our objects of interest we can use Matlab function `BW2 = bwareaopen(BW,P)` which removes all connected components (objects) that have fewer than P pixels from the binary image BW, producing another binary image, BW2. This operation is known as an area opening and is shown in Figure 4b.



(a) (b)

Figure 4: a) The results after performing thresholding. b) and after removing small objects from the binary image. The five distinct objects are visible.

Then we can dilate the image with a vertical line structuring element with the function `imdilate()` and then we calculate region properties of objects in the image using the binary image. An object in a binary image is a set of white pixels (ones) that are connected to each other. We can enumerate all the objects in the figure using the bwlabel command: `[L, num] = bwlabel(f)` where L gives the labeled image, and num gives the number of objects. In this way, we can label the binary image.

Once the image has been labeled, we use the `regionprops` command to obtain quantitative information about the objects. There's a lot of useful statistical information about objects that can be extracted using regionprops. In our case, BoundingBox feature is used to mark the detected differences as shown in Figure 5.

[L, N] = bwlabel(bw3);
bb = regionprops(L, 'BoundingBox', 'Area');

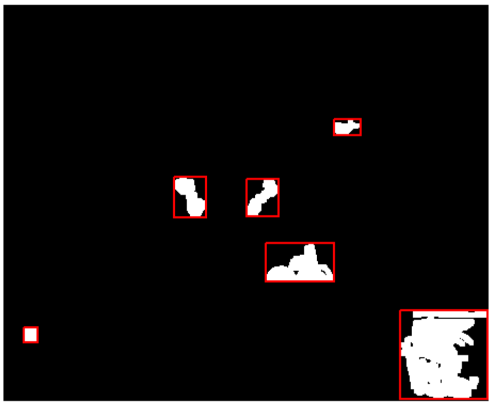


Figure 5: The results after marking the detected differences in the dilated binary image with red squares.

Listing 1: The code for the identifying the five differences in the two images femfel .

```

clc; close all; clear;
load femfel

figure(1);
imshow([femfel1, femfel2]), title('Original images')

% Display the difference of femfel1 and femfel2.
```

```
figure(2), imshowpair(femfel1, femfel2, 'diff')
title('the difference of femfel1 and femfel2')

%% convert to grayscale images
gray1 = rgb2gray(femfel1);
gray2 = rgb2gray(femfel2);
%figure(2)
%imshow([gray1, gray2]), title('Grayscale images')

graydiff = gray2 - gray1;
figure(22), imshow(graydiff);
title('The differences between the two images')

%% cross-correlation function
im1 = rgb2gray(femfel1);
im2 = rgb2gray(femfel2);

b = normxcorr2(im1,im2);
figure(23); mesh(b);
title('The cross-correlation function on the two images')

% find the maximum point of this matrix:
[y,x] = find(b == max(b(:)));

% determine the offset of these positions from the origin of the matrix:
[yc,xc] = size(im2);
yoff = y - yc;
xoff = x - xc;

%% Perform the registration
% transforms the 2-D or 3-D image, moving, so that it is registered with
% the reference image, fixed.
fixed = im1;
moving = im2;

% Use imregconfig to create the default metric and optimizer for a capture
% scenario in one step.
[optimizer,metric] = imregconfig('monomodal');
%movingRegistered = imregister(moving, fixed, 'translation', optimizer, metric);

% the 'similarity' transformation types always involve nonreflective
% transformations.
% The transformed image, movingRegistered, returned as a matrix.
% Any fill pixels introduced that do not correspond to locations in
% the original image are 0.
movingRegistered = imregister(moving, fixed, 'similarity', optimizer, metric);


% View the registered images
figure(24), imshowpair(fixed,movingRegistered, 'Scaling','joint')
%figure(24), imagesc(movingRegistered)
title('The output results of the registration using imregister()')

figure(25), imshowpair(fixed,movingRegistered, 'diff')
title('The difference after the registration using imregister()')

diffreg = movingRegistered - fixed;

%% Image histograms and manual thresholding
```

```
figure(26)
imhist(diffreg)
title('The distribution of grayscale intensity values in the diff image')

% thresholding manually
bwdiffreg = diffreg > 55;
figure(255), colormap(gray), imshow(bwdiffreg)
title('The thresholding result of grayscale intensity values in the diff
    image')

%% automatic thresholding
level = graythresh(diffreg);
imb = im2bw(diffreg,level);
figure(27), imshow(imb)

%% remove disturbances
bw2 = bwareaopen(bwdiffreg,40,8);
figure(28), imshow(bw2), title('The cleaned differences')

% creates a rectangular structuring element, where mn specifies the size.
se = strel('square',4);
% Dilate the image with a vertical line structuring element and compare
% the results.

bw3 = imdilate(bw2,se);
figure(29), imshow(bw3), title('The dilated differences')


%% calculate region properties of objects in the image

[L, N] = bwlabel(bw3);
bb = regionprops(L, 'BoundingBox', 'Area');
figure(299), imshow(bw3)
% imtool provides access to several other tools for navigating and
    exploring
% images, such as the Pixel Region tool, Image Information tool, and the
% Adjust Contrast tool.
%imtool(bw3)
hold on

for i=1:size(bb)
    rectangle('position',bb(i).BoundingBox, 'EdgeColor', 'r', 'LineWidth'
        ,2)
end


%% using imtool to localize the positions
posx = [402, 285, 244, 319, 27, 177];
posy = [330, 246, 182, 117, 310, 179];
for i=1:length(posx)
 rectangle('position',[posx(i) posy(i) 10 10], 'EdgeColor', 'b', 'LineWidth
    ',2)
end
```

3 Segmentation with Graph Cuts

In this task we need to segment out two heart chambers in an image by `Graph Cuts`. The given data `heart_data.mat` contains a number of intensities which have been observed inside the two chambers, *chamber class*, and a number of intensities observed in the background, *background class*. The given heart image is shown in Figure 6a.

3.1 Estimation of the mean and the standard deviation

Assuming that the pixel intensities in the two classes are generated by two different `Gaussian distributions`, we need to estimate μ_1, μ_2, σ_1 and σ_2 for these two distributions. Thus, for pixel i, the likelihoods of observing intensity f_i, P(f_i| chamber class) and P(f_i| background class), are Gaussian, i.e.

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

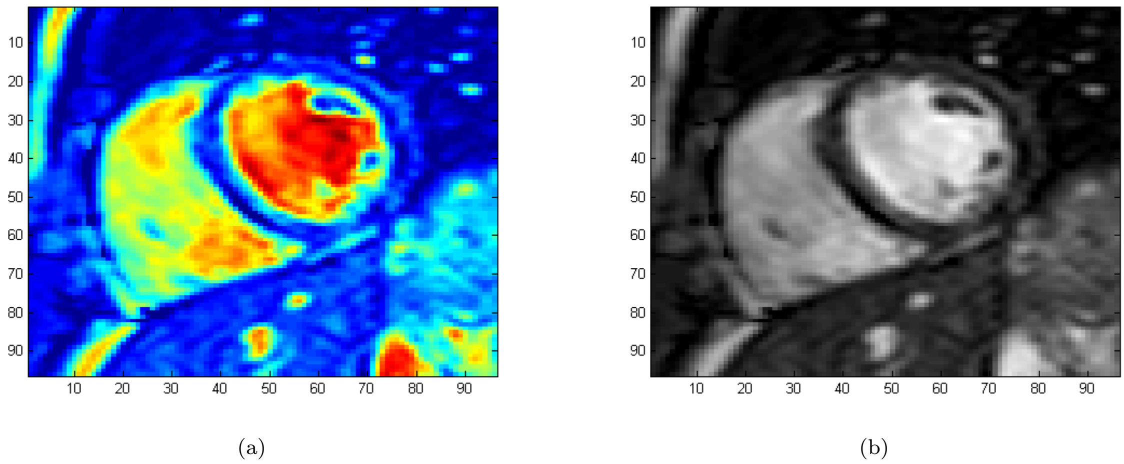


Figure 6: a) The data given in the task. A slice of a heart imaged by a MRI camera. This view is known as the "short-axis view" taken in a direction where the left and the right heart chambers are visible at the same time. b) Heart image in grayscale.

3.2 Construction of a graph for the heart image

In this section of the task, we need to onstruct a graph of the heart image with a `data-term` consisting of the negative `log likelihoods` for the two classes, and then Solve it via `max-flow/min-cut`.

In order to obtain a reasonable segmentation, we can experiment with the `prior/regularization weight` denoted by ν in the lectures. we should remember that it is hard to get a perfect segmentation.

A useful model for image segmentation can be such that the foreground is modelled as having a constant grey level μ_1 and the background as being constant equal to mu_0. The segmentation task is to find a curve γ such that the region Γ inside the curve is foreground and the exterior is background. See Figure **??**. Finding the best possible curve results in an optimization problem and is the topic of this computer session.

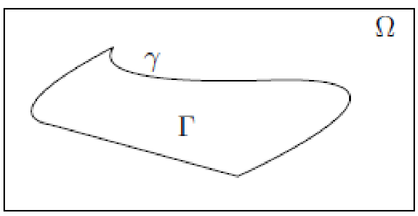


Figure 7: Image segmentation model.

3.3 Segmentation problem - The Mumford-Shah functional

Segmentation problem according to Mumford-Shah consists in computing a decomposition of the domain of the image $f(x, y)$

$$R = \cup_{i=1}^{n} R_i \cup \Gamma$$

where f varies smoothly and/or slowly within R_i but varies discontinuously and/or rapidly across most of the boundary Γ between regions R_i. Thus, segmentation problem may be restated as finding `optimal approximation` of a general function f by piece-wise `smooth functions g`, whose restrictions g_i to the regions R_i are differentiable.

Mumford and Shah defined a `general functional` E

$$E(g, \Gamma) = \mu^2 \int_R (g - f)^2 dxdy + \int_{R-\Gamma} \|\nabla g\|^2 dxdy + \nu|\Gamma| \tag{1}$$

As we can see, the energy functional E depends on g and Γ and the problem is to find g and E such that $E(g, \Gamma)$ is minimized. Thus, the smaller E, the better (g, Γ) segments f. In this way, (g, Γ) is simply a `cartoon` of the original image f where the objects are drawn smoothly without texture and g can be considered to be a new image with edges drawn sharply.

A special case of E where $g = a_i$ is constant on each open set R_i, i.e. a piecewise constant approximation

$$E_0(g, \Gamma) = \sum_{i=1}^{n} \int_{R_i} (a_i - f)^2 dxdy + \nu|\Gamma| \tag{2}$$

It is minimized in a_i by setting a_i to the mean of f in R_i

$$a_i = \int_{R_i} \frac{f}{area(R_i)} dxdy$$

3.4 Statistical two-phase Mumford-Shah

For the statistical interpretation, the `observed image` is denoted by f, the `sought or unknown image` is denoted by g and the `posterior distribution` is denoted by $P(g|f)$. Using the Maximum a Posteriori (MAP) principle, we should maximize the posterior distribution through the application of Bayes rule

$$P(g|f) = \frac{P(f|g)P(g)}{P(f)}$$

Then negative logs give

$$-log(P(g|f)) = -log(P(f|g)) - log(P(g)) + constant$$

Using Equation 2, two phase Mumford-Shah functional is defined as the energy is based on two segments R_1 and R_2, assuming that a_1 and a_2 are known and the regularization is based on boundary length

$$E(f, g) = E_{data}(f, g) + E_{prior}(g)$$

$$E_0(a_1, a_2, \Gamma) = \int_{R_1} (a_1 - f)^2 dxdy + \int_{R_2} (a_2 - f)^2 dxdy + \nu|\Gamma| \tag{3}$$

More general formulation with a `data term` consisting of the negative log likelihood for the two classes

$$E_0(\Gamma) = \int_{R_1} -log(P(f(x, y)|class1))dxdy + \int_{R_2} -log(P(f(x, y)|class2))dxdy + \nu|\Gamma| \tag{4}$$

As mentioned in `lab3en.pdf`, a simple version of the `Mumford-Shah functional` takes the length of γ into account

$$E(\gamma) = \lambda \text{ length}(\gamma) + \iint_{\Gamma} (I(x) - \mu_1)^2 dx + \iint_{\Omega\backslash\Gamma} (I(x) - \mu_0)^2 dx \tag{5}$$

Then, we need to find the curve γ which minimizes $E(\gamma)$ using a discrete approximation of E and an indicator variable θ for the foreground, i.e.

$$\theta_i = \begin{cases} 1, & \text{if i is foreground} \\ 0, & \text{if i is background} \end{cases}$$

Let i and j be two image pixel locations. We say that i is a neighbor of j if they are adjacent, either horizontally or vertically. We write this as $j \in N_i$.

Now we can approximate Equation 5 by

$$E(\theta) = \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{j \in N_i} \mathbb{I}(\theta_i \neq \theta_j) + \sum_{i=1}^{n} \theta_i (I(i) - \mu_1)^2 + \sum_{i=1}^{n} (1 - \theta_i)(I(i) - \mu_0)^2. \tag{6}$$

where I is a discrete image with n pixels. The first double summation counts the total number of separated neighbors. Minimizing Equation 6 is a `minimum cut` problem that we can solve it using the software written by Boykov and Kolmogorov.

3.5 Results

The estimated mean and the standard deviation of the two distributions related to chamber class and background class are

$$\mu_0 = 0.1065 \quad \mu_1 = 0.3542$$

$$\sigma_0 = 0.0936 \quad \sigma_1 = 0.0992$$

The result of running the algorithm in Listing 2 is illustrated in Figure 8.

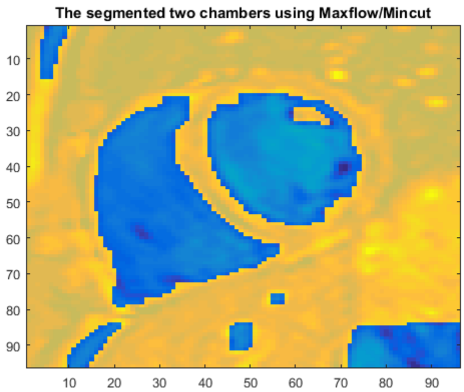


Figure 8: The segmented two chambers using Maxflow/Mincut algorithm.

Listing 2: The code for heart segmentation problem using Max-flow Min-cut theorem .

```

%                    Image Analysis − Handin 4 − task 2
%                    2 − Segmentation the heart image with Graph Cuts
%                        using Max−flow Min−cut theorm
%                        Hicham Mohamad − hsmo@kth.se

clear, clc, close all;

load heart_data.mat
%figure(1); clf;
%imagesc(im)
[M N] = size(im);

%%
figure(2); clf;
% colormap() Set and get current colormap
% gray returns a linear grayscale colormap.
colormap gray
grayheart = imagesc(im);

%% Number of image pixels
n = M*N;
```

```
%{
Let i and j be two image pixel locations. We say that i is a neighbor of j
if they are adjacent, either horizontally or vertically.
We write this as j in N_i.
%}
% create the neighbors and the sparse matrix
Neighbors = edges4connected(M,N);
i = Neighbors(:,1);
j = Neighbors(:,2);
lambda = 2;
% create the sparse matrix
A = sparse(i,j,lambda,n,n);

% 1) Estimate the mean and the standard deviation for the 2 distributions.
mu0 = mean(background_values);
mu1 = mean(chamber_values);

sigma0 = std(background_values);
sigma1 = std(chamber_values);

%{
 Now we handle the other two sums in (3). They will be represented with s
and t connections in the graph (see the lecture notes).
In the software package we are going to use, these are represented as a
separate n x 2 matrix:
%}
T = [ ((im(:) - mu0).^2)/sigma0^2 ((im(:) - mu1).^2)/sigma1^2 ]/2;
%T = [ ((im(:) - mu0).^2)/sigma0^2 ((im(:) - mu1).^2)/sigma1^2 ];
T = sparse(T);

% Finally, we solve the minimum cut problem
[E, Theta] = maxflow(A,T);
Theta = reshape(Theta,M,N);
Theta = double(Theta);
Theta = Theta ~= 0;

% And we can now view the output
%imshow(Theta);

im_mincut = im + Theta;

figure(12)
imagesc(im_mincut);
%imshow(mincut)
title('The segmented two chambers using Maxflow/Mincut')

%imshowpair(grayheart,mincut,'montage')
```

4 Computer Vision

Assuming that the camera matrices for two projections are

$$P_1 = \begin{pmatrix} 3 & 2 & 1 & 0 \\ 2 & 2 & 3 & 0 \\ 2 & 2 & 2 & 1 \end{pmatrix} \quad and \quad P_2 = \begin{pmatrix} 2 & 4 & 3 & 3 \\ 1 & 2 & 0 & 2 \\ 1 & 1 & 3 & 0 \end{pmatrix}$$

The so called `fundamental matrix` is then

$$F = \begin{pmatrix} 3 & -3 & -6 \\ -6 & 7 & 9 \\ -4 & 6 & 2 \end{pmatrix}$$

The following three points are detected in image 1:

$$a1 = (1, 2), \quad a2 = (16, 10), \quad a3 = (-7, -8).$$

In image 2 the following three points are detected:

$$b1 = (1, 1), \quad b2 = (3, 2), \quad b3 = (-1, -3).$$

The fundamental matrix F maps points in image 1 to lines in image 2. If $\bar{x}$ corresponds to x then the `epipolar constraint` can be written

$$\bar{x}^T F x = 0.$$

For cameras P_1 and P_2, the corresponding image points x_i and $\bar{x}_i$ fulfills

$$\bar{x}_i^T F x_i = 0.$$

In Lecture 5 we considered the `two-view structure` from motion problem, that is, given a number of measured points in two images we want to compute both camera matrices and `3D points` such that they project to the measurements. We showed that the 3D points can be eliminated from the problem by considering the fundamental matrix F. If x is an image point belonging to the fist image and $\bar{x}$ belongs to the second then there is a 3D point that projects to these if and only if the epipolar constraint

$$\bar{x}^T F x = 0$$

is fulfilled.

In general we may assume (see Lecture 5) that the cameras are of the form $P_1 = [I \quad 0]$ and $P_2 = [A \quad e_2]$ where e_2 is the `epipole` in the second image. Since we know that $F^T e_2 = 0$ we can find e_2 by computing the null space of F^T. A solution for the second camera is then given by

$$P_2 = [[e_2]_\times F \quad e2].$$

These cameras have the fundamental matrix F when their projections fulfill the `epipolar constraint`. If

$$X = \begin{bmatrix} X \\ \rho \end{bmatrix}, \quad where \quad X \in R^3 \quad and \quad \rho \in R$$

then the projections in the two cameras are given by

$$x \approx \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} X \\ \rho \end{bmatrix} = X$$

$$\bar{x} \approx [[e_2]_\times F \quad e2] \begin{bmatrix} X \\ \rho \end{bmatrix}$$

4.1 Results

b_2 corresponds to a_2 and b_3 corresponds to a_3.

For finding these results, the calculations are performed using matlab, as shown in Listing 3. This derives from the fact, as explained above, that for cameras P_1 and P_2, the corresponding image points x_i and $\bar{x}_i$ fulfills

$$\bar{x}_i^T F x_i = 0$$

Listing 3: The code for the calculations to find the correspondences for the points .

```

%% Computer vision
clear, clc, close all;

```

```
% Assume that the camera matrices for two projections are
P1 = [3  2  1  0;
      2  2  3  0;
      2  2  2  1  ];
P2 = [  2  4  3  3;
        1  2  0  2;
        1  1  3  0  ];
% The so called fundamental matrix is then
F = [  3  -3  -6;
      -6   7   9;
      -4   6   2  ];
% The following three points are detected in image 1:
a1 = [1  2  1  ]'; a2 = [16  10  1  ]'; a3 = [-7  -8  1  ]';
% In image 2 the following three points are detected:
b1 = [1  1  1  ]'; b2 = [3  2  1  ]'; b3 = [-1  -3  1  ]';
%Which points can be in correspondence?
%For the report: Provide your calculations, your answer and your motivation
    .

% compute the projection of a1, a2, a3 in P1
% a1p = P1 * a1;
% a2p = P1 * a2;
% a3p = P1 * a3;
% compute the projection of b1, b2, b3 in P2
% b1p = P2 * b1;
% b2p = P2 * b2;
% b3p = P2 * b3;

% Any corresponding abar to a should fullfill abar'*F*a = 0
Funda = cell(1,3);
Funda{1} = F * a1;
Funda{2} = F * a2;
Funda{3} = F * a3;
[m,n] = size(Funda);
for i = 1:n
    b1' * Funda{i}
    b2' * Funda{i}
    b3' * Funda{i}
end

% Result: b2 corresponds to a2 and b3 corrsponds to a3
```

5 OCR system construction and system testing

Here we will continue and finish the development of our OCR system. From the three previous assignments we have constructed

`S = im2segment(Im)` - a segmentation algorithm that takes an image to a number of segments.

`x = segment2features(S)` - an algorithm for calculating a feature vector x from a segment S.

`y = features2class(x,classification_data)` - an algorithm for classifying a feature vector x as class y using machine learning.

Again, the variable `classification_data` is whatever we need to store from the training of the classifier. This depends on the machine learning algorithm we are using, which is K-nearest neighbors. In the folder `task4/ocr_project/matlab`, there is a Matlab file `ocrsegments.mat`. It contains a number of segments in a cell array S and corresponding letter code y. Again, these are coded from 1 to 26, where 1 corresponds to 'A', 2 corresponds to 'B' and so forth. Using our own set of favorite features, we can go from segments in S to corresponding feature vectors in a matrix $\mathbf{X}$, where each column corresponds to features of a segment in S.

In this task, we should try running `inl4_test_and_benchmark` on all of the five datasets: short1, short2, home1, home2 and home3, before modifying the code. In this way, we can test the version 1 of our system from Assignment 3 on these five datasets and check that it works by obtaining the `overall hitrate`. Then, we need to improve the system and produce a version 2 that works better on the five datasets.

Here in the table, we can show the results of the hitrates and the average error rates for my implemented classifier KNN running on the five datasets.

	short1	short2	home1	home2	home3	Mean hitrate	Error rate
OCR-system version 1	0.8600	0.7400	0.0790	0.0850	0.0820	0.3692	0.6308
OCR-system version 2	0.4400	0.5200	0.8880	0.8730	0.7720	0.6986	0.3014

Table 1: Average error rates for the four classifiers both on training and testing data.

Looking at the results in table, we notice that it didn't go well for the version 1 of the OCR system. It is bad for the last 3 data sets and the overall hitarate is only 0.3692. This bad performance must be related to bad features extraction.

For the version 2 instead, we can say that it went well. The hitrate for the last 3 datasets is nice and the overall hitrate is almost 0.7. This good result derives from the fact that I had used other extra features using `regionprops` and I should perform some form of normalization by dividing the features by the area or the width such that the feature vector becomes more independent of the size. In addition, I needed to fine-tune the model also by decreasing the threshold value that seperates the background from the objects in the gray scale image. The changing and tuning of the KNN classifier is illustrated in Listing 4 while the same function in version 1 is shown in Listing 5.

5.1 Conclusion and Challeges

We can conclude that having `good features` is essential for successful Machine Learning. It can be 90% of the effort, as mentioned in [5]. Then, these error rates on the datasets can be also derived from the fact that this simple algorithm is not compatible with this type of problem we are working on. The simple machine learning algorithms work well on a wide variety of important problems. They have not succeeded, however, in solving the central problems in AI, such as recognizing speech or recognizing objects which is similar to our problem. The development of `deep learning` was motivated in part by the failure of traditional algorithms to generalize well on such AI tasks. In other words, Many machine learning problems become exceedingly difficult when the number of dimensions in the data is high. This phenomenon is known as the `curse of dimensionality`. There are also other factors that have limited the ability of traditional machine learning to generalize to new data. These challenges have motivated the development of `Deep Learning` algorithms that overcome these obstacles.

Listing 4: The code for the function segment2features in version 2 .

```

                    % Handin 4 - Task 4
                    % fucnction segment2features2(S)
%{
  Here we implement an algorithm for calculating a feature vector x from a
segment S. From the previous assignments, this function has been
constructed and now we need to try improving the code to make it as good
as possible. So we need to change the system and produce a version 2 that
works better on the five datasets (short1, short2, home1, home2, home3).

NOTE: Szeliski 14.4
The problem of searching for features (patterns) in data is a fundamental
one. The major goal of image feature extraction is that given an image,
or a region within an image, generate the features that will subsequently
be fed to a classifier in order to classify the image in one of the
    possible classes.

%}

function features = segment2features2(B)
```

```

% initialize the features vector x
%features = randn(6,1);
features = ones(7,1);

% Given a binary image B (thus only with zeros and ones), 1 => object pixel
% 0 => background pixel. So an object in a binary image is a set of white
% pixels (ones) that are connected to each other.
% study the region of pixels that are equal to 1.
% Use your imagination to define at least 6 different features (numbers),
% that can be used to classify which letter the region corresponds to.

% region of interest: Identify the region of pixels that are equal to one
[y x] = find( B == 1 );
xmax = max(x);
xmin = min(x);
ymax = max(y);
ymin = min(y);
regionOfB = B(ymin:ymax,xmin:xmax);

% size of the image region
[m,n] = size(regionOfB);

%               ####### Features extraction  #########

                % Feature 1: the sum of pixel values
                % Feature 2: the left vertical half of the image
                % Feature 3: the upper horizontal half of the image
                % Feature 4: the under horizontal half of the image

                % Some region properties
                        % prop 1: Area of smallest convex polygon
                        % prop 2: Pixelsum of the box that contains the letter
                        % prop 3: Center of mass
                        % prop 4: Circumfere properites
                        % prop 5: equal diameter
                        % prop 6: Filled area
                        % prop 7: Mean of convex hull pixels which will be added to the hitrate
                        % prop 8: Minor and major axis
                % Feature 5: calculate the norm of the features


i = 1;

% Feature 1: the sum of pixel values
features(i) = sum(sum(regionOfB))/(m*n);
i = i + 1;

% Feature 2: the left vertical half of the image
n2 = ceil(n/2);
features(i) = sum(sum(regionOfB(:,1:n2))) / (m*n2);
i = i + 1;
% Feature 3: the upper horizontal half of the image
m2 = ceil(m/2);
features(i) = sum(sum(regionOfB(1:m2,:))) / (m2*n);
i = i + 1;
% Feature 4: the under horizontal half of the image
features(i) = sum(sum(regionOfB(m2:m,:))) / (m2*n);
i = i + 1;

```

```
% calculate region properties of objects in the image
%{
Using the binary image, we can then calculate region properties of objects
in the image, such as area, diameter, etc...
Using the regionprops() command to obtain quantitative information about
the objects:
    D = regionprops(L, properties)
There is a lot of useful statistical information about objects that can be
extracted using regionprops
%}
reg = regionprops(regionOfB,'all');

% prop 1: Area of smallest convex polygon
features(i) = reg.ConvexArea / (m*n);
i = i + 1;

% prop 2: Pixelsum of the box that contains the letter
bb = reg.BoundingBox;
features(i) = sum(bb) / m;
i = i + 1;

% prop 3: Center of mass
cent = reg.Centroid;
features(i) = cent(2) / m;
i = i + 1;
features(i) = cent(1) / n;
i = i + 1;

% prop 4: Circumfere properites
if reg.PerimeterOld ~= 0
    features(i) = reg.Perimeter / reg.PerimeterOld;
    i = i + 1;
else
    features(i) = reg.Perimeter / (m+n);
    i = i + 1;
end

% prop 5: equal diameter
features(i) = reg.EquivDiameter / (n);
i = i + 1;

% prop 6: Filled area
features(i) = reg.FilledArea / (m*n);
i = i + 1;

% prop 7: Mean of convex hull pixels which will be added to the hitrate
hull = reg.ConvexHull;
hull = mean(hull(:,1))/(m);
features(i) = hull;
i = i + 1;

% prop 8: Minor and major axis
features(i) = reg.MinorAxisLength / reg.MajorAxisLength;
i = i + 1;


% Feature 5: calculate the norm of the features
features(i) = norm(features);
i = i + 1;

```

```
end
```

Listing 5: The code for the function segment2features in version 1 .

```

        % OCR - Feature Extraction
        % FMAN20 - Image Analysis - Handin 2
        % Hicham Mohamad - hsmo@kth.se

function x = mySegment2features(B)
% features = segment2features(I)

% initialize the features vector x
%features = randn(6,1);
%x = zeros(9,1);
x = zeros(7,1);

% Given a binary image B (thus only with zeros and ones), 1 => object pixel
% 0 => background pixel.
% study the region of pixels that are equal to 1.
% Use your imagination to define at least 6 different features (numbers),
% that can be used to classify which letter the region corresponds to.

% Localisation: Identify the region of pixels that are equal to one
[yPosition xPosition] = find( B == 1 );
xmax = max(xPosition);
xmin = min(xPosition);
ymax = max(yPosition);
ymin = min(yPosition);
regionOfB = B(ymin:ymax,xmin:xmax);

% size of the image region
[m,n] = size(regionOfB);

%        ####### Features extraction #########
            % Feature 1: moment
            % Feature 2: vertical histogram
            % Feature 3: the sum of pixel values
            % Feature 4: Center of mass
            % Feature 5: Orientation
            % Feature 6: Crossing
            % Feature 7: Number of holes
            % Feature 8: Average of row sum
            % Feature 9: Average of column sum
i = 1;

% Feature 1: moment
% SIGMA = moment(X,ORDER) returns the ORDER-th central sample moment of
% the values in X. For vector input, SIGMA is MEAN((X-MEAN(X)).^ORDER).
% For a matrix input, moment(X,ORDER) returns a row vector containing the
% central moment of each column of X.
x(i) = mean(moment(regionOfB,3));
i = i + 1;

%{
% Feature 2: vertical histogram
%n2 = floor(n/2);
n2 = ciel(n/2);
%if n2 == 0          % XXXXX
%     n2 = n2 + 1;
%end
% A histogram is the frequency of occurrence vs. gray level
```

```
x(i) = sum(sum(regionOfB(:,n2:n)));          % XXXXXX
i = i + 1;
%}

% Feature 3: the sum of pixel values
x(i) = sum(sum(regionOfB));
i = i + 1;

% Feature 4: Center of mass
% STATS = regionprops(BW,PROPERTIES) measures a set of properties for
% each connected component (object) in the binary image BW, which must be
% a logical array; it can have any dimension.
% Calculate centroids for connected components in the image
stats = regionprops(regionOfB, 'Centroid');
sx = stats.Centroid(1);
sy = stats.Centroid(2);
x(i) = sx + sy;
i = i + 1;

% Feature 5: Orientation
% Angle between the x-axis and the major axis of the ellipse that has the
% same second-moments as the region, returned as a scalar.
% The value is in degrees.
stats = regionprops(regionOfB, 'Orientation');
x(i) = abs(stats.Orientation);
i = i + 1;

%{
% Crossing feature 6: Scan through the rows and columns to obtain the
% horizontal and vertical crossing times of a character.

% Vertical crossings: for any column j = 0,1,2,...,n
% number of vertical crossings in column j
numVerCross = 0;
pixel = 0;
j = 0;
% for any row j = 0,1,2,...,n
while j < m
    while pixel == 0 && j < m
        j = j + 1;
        pixel = regionOfB(j, floor(n/2));
        % floor(X) rounds the elements of X to the
        % nearest integers towards minus infinity.
    end

    while pixel == 1 && j < m
        j = j + 1;
        pixel = regionOfB(j, floor(n/2));
    end

    numVerCross = numVerCross + 1;
end

if pixel == 0
    numVerCross = numVerCross - 1;
end

% Horizontal crossings
% number of horizontal crossings in row i
numHorCross = 0;
pixel = 0;
```

```
j = 0;

% for any column j = 0,1,2,...,n
while j < n
    while pixel == 0 && j < n
        j = j + 1;
        pixel = regionOfB(floor(m/2),j);
    end

    while pixel == 1 && j < n
        j = j + 1;
        pixel = regionOfB(floor(m/2),j);
    end

    numHorCross = numHorCross + 1;
end

if pixel == 0
    numHorCross = numHorCross - 1;
end

x(i) = (numHorCross + numVerCross);
i = i + 1;
%}

% Feature 7: Number of holes using bwEuler() function
% EUL = bweuler(BW,N) returns the Euler number for the binary
% image BW. EUL is a scalar whose value is the number of
% objects in the image minus the total number of holes in those
% objects. N can have a value of either 4 or 8, where 4
% specifies 4-connected objects and 8 specifies 8-connected
% objects; if the argument is omitted, it defaults to 8.
x(i) = bweuler(regionOfB);
i = i + 1;

% Feature 8: Average of row sum
% S = mean(X) is the mean value of the elements in X if X is a vector.
% For matrices, S is a row vector containing the mean value of each column.
x(i) = mean(sum(regionOfB'));
i = i + 1;

% Feature 9: Average of column sum
x(i) = mean(sum(regionOfB));
i = i + 1;

end
```

6 Optional: OCR using Deep Learning

This exercise is optional, but for those of you who are interested in deep learning it might be both interesting and useful. On the homepage `http://www.robots.ox.ac.uk/ vgg/practicals/cnn/` you can find an introduction to `convolutional neural networks` (CNN) for classification.

Download the data and the code according to the instructions.

Then do exercise 4 on the homepage and answer all questions.

If you have time, try to integrate the final classifier in your own ocr code.

References

[1] Szeliski, Computer Vision - Algorithms and Applications, Springer.

[2] Forsyth and Ponce, Computer Vision - A Modern Approach, Pearson Education, ISBN 0-13-191193-7

[3] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*, volume 1 Addison-Wesley, 1991.

[4] Kevin P. Murphy. *Macine Learning: A Probabilistic Perspective*, The MIT Press, Cambridge, Massachusetts.

[5] Ian Goodfellow, Yoshua Bengio, Aaron Courville. *Deep Learning.* MIT Press, 2016, ISBN: 9780262035613.

[6] Matlab journal demos in Image Analysis. Available: `http://www.ctr.maths.lu.se/matematiklth/personal/kalle/imageanalysis/`.

[7] Linear Algebra. Available: `http://www.immersivemath.com`

[8] Computer Vision - CS6670, Lectures notes [Online]. https://www.cs.cornell.edu/courses/cs6670/2018fa/calendar-final.html

[9] Maskinsyn - UNIK4690, Lectures notes [Online]. https://www.uio.no/studier/emner/matnat/its/UNIK4690/v16/time

[10] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *Gradient-Based Learning Applied to Document Recognition.* PROC. OF THE IEEE, November 1998.

[11] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning.* Second Edition, Springer.

[12] Mathworks, Intensity-based image registration [Online]. https://www.mathworks.com/help/images/ref/imregister.ht