Image Analysis - FMAN20
Assignment 2

Convolution, Filtering, Interpolation
Classification
OCR - Feature Extraction

Hicham Mohamad
hsmo@kth.se

November 8, 2018

1 Introduction

In this assignment you will study convolution, filtering, interpolation, classification and continue your work on your own Optical Character Recognition system.

2 Preliminaries: Linear Filtering, Convolution and Correlation

In linear filtering, an output pixel's value is determined as a **weighted sum** of input pixel values. An example of neighbourhood filtering is the *convolution* which is defined by

$$g = f * h \implies g(i,j) = \sum_u \sum_v f(i-u, j-v)\ h(u,v) = \sum_u \sum_v f(u,v)\ h(i-u, j-v) \tag{1}$$

where the entries in the *weight kernel* or *mask* $h(u,v)$ are often called the filter coefficients.

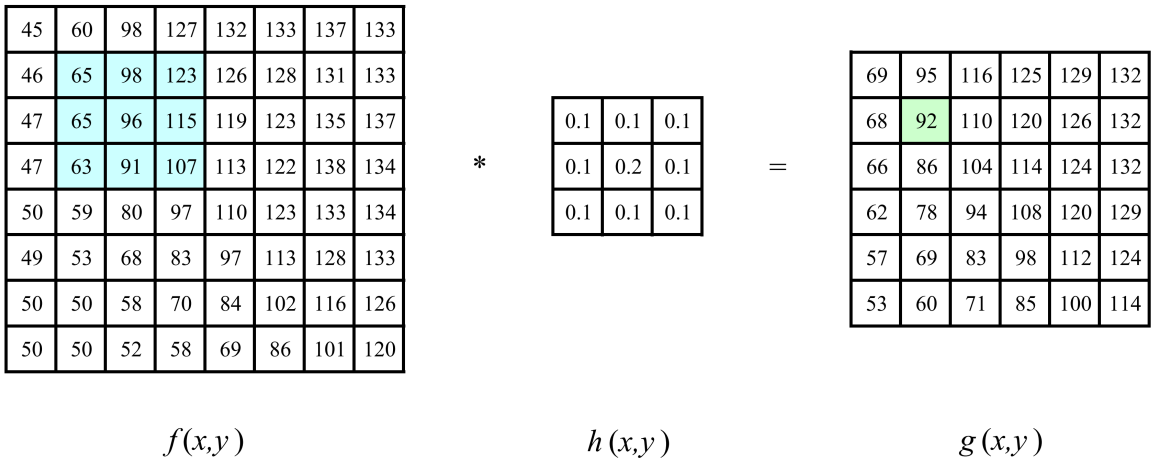


Figure 1: Neighborhood filtering (Convolution): The image on the left is convolved with the filter in the middle to yield the image on the right. The light blue pixels indicate the source neighborhood for the light green destination pixel. Image courtesy: [Richard Szeliski, 2010] [1]

Actually, the process of applying the filter is usually referred to as convolution $g = f * h$, and h is then called the *impulse response function*. The reason for this name is that the *kernel function*, h, convolved

with an *impulse signal*, $\delta(i,j)$ (an image that is 0 everywhere except at the origin) reproduces itself, $h * \delta = h$, whereas *correlation* produces the reflected signal:

$$g = f \otimes h \implies g(i,j) = \sum_u \sum_v f(i+u, j+v)\ h(u,v) \tag{2}$$

This convolution concept can be seen clearly in Figure 1 using the example mentioned in the text book [1]. In fact, Equation 1 can be interpreted as the superposition (addition) of shifted impulse response functions $h(i-u, j-v)$ multiplied by the input pixel values $f(u,v)$. Convolution has additional nice properties, e.g., it is both commutative and associative. As well, the Fourier transform of two convolved images is the product of their individual Fourier transforms [1].

3 Filtering

In this section, the main idea in this problem is to convolve a given image with six different filters. Thus, we need to combine each given convolved image with the correct filter.
Just as smoothing is a fundamental operation in image processing so is the ability to take one or more `spatial derivatives` of the image. The fundamental problem is that, according to the mathematical definition of a derivative, how can we differentiate a digital image $F[x,y]$. This cannot be done directly, because a digitized image is not a continuous function. Derivatives are naturally approximated by the discrete derivative, i.e. `finite differences`

$$\frac{\partial f}{\partial x}[x,y] \approx F[x+1,y] - F[x,y]$$

and consequently we can estimate a partial derivative as a `symmetric difference`:

$$\frac{\partial h}{\partial x} \approx h_{i+1,j} - h_{i-1,j}$$

As an image is a function of two (or more) variables it is necessary to define the direction in which the derivative is taken. as shown in Figure 2. For the two-dimensional case we have the horizontal direction, the vertical direction, or an arbitrary direction which can be considered as a combination of the two. If we use h_x to denote a `horizontal derivative` filter (matrix), h_y to denote a `vertical derivative` filter (matrix), and h_θ to denote the arbitrary angle derivative filter (matrix), then:

$$\mathbf{h}_\theta = \cos\theta \cdot \mathbf{h}_x + \sin\theta \cdot \mathbf{h}_y$$

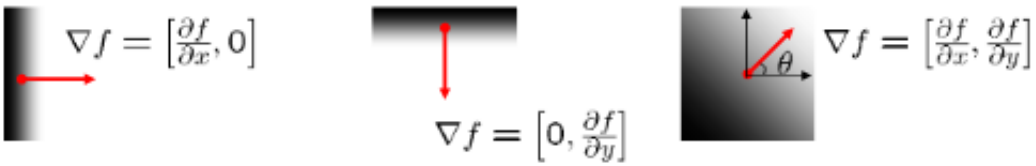


Figure 2: The gradients of an image. The gradient points in the direction of most rapid increase in intensity

Combinations of images and filters:

- Looking at our different filters, we notice that f_1 and f_2 are one-dimensional `basic derivatives filters` which show changes with respect to x and y respectively and they lead to a `phase shift`

$$f_1 = \begin{bmatrix} -1 & 1 \end{bmatrix} = h_x, \quad f_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} = h_y$$

  Thus, f_1 corresponds to image C which shows the partial derivative in the x-direction, because it responds strongly to vertical edges and weakly to horizontal edges. Whereas f_2 corresponds to image B which shows partial derivative in the y-direction and responds strongly to horizontal edges and weakly to vertical edges.

- f_5 and f_6 can be considered as two-dimensional derivative filters, i.e. second derivatives which can be calculated from a combination of the three second order derivatives R_{xx}, R_{xy}, R_{yy}:

$$f_5 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad f_6 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

  Where f_5 is a combination of R_{xx} and R_{yy} and corresponds to image E. Whereas, f_6 is a combination of R_{xy} and corresponds to image D.

- f_3 is a negative `Laplacian filter` and it accentuates differences, which it can be a combination of 2 `basic second derivative filters` specified by

$$h_{2x} = (h_{2y})^T = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

  and corresponds to image A.

$$f_3 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- f_4 is a linear `smoothing` filter with uniform average algorithm, where each pixel is replaced with an average of all the values in its neighborhood, i.e. mean value. This type of filter is applied in order to reduce noise and/or to prepare images for further processing such as segmentation. But it produces a bluring effect. This filter corresponds to image F.

$$f_4 = \frac{1}{16} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

4 Interpolation

Here is another issue of interest, after that one of image filtering, we need to address linear interpolation.

a) Definition

It is 1D first order interpolation, i.e. linear, and is a fundamental tool in `digital processing` of images for bridging the continuous world and the discrete world. Thus, given an image with discrete representation one can obtain a continuous version by interpolation. In MATLAB implementation, linear interpolation can be calculated by: `z1=interp2(x,y,z,x1,y1,'linear');`
In this task, it is assumed a one-dimensional signal (image)

$$f(i) = \begin{bmatrix} 1 & 2 & 2 & 4 & 7 & 6 & 5 \end{bmatrix}$$

The function f after linear interpolation is illustrated in Figure 3 below

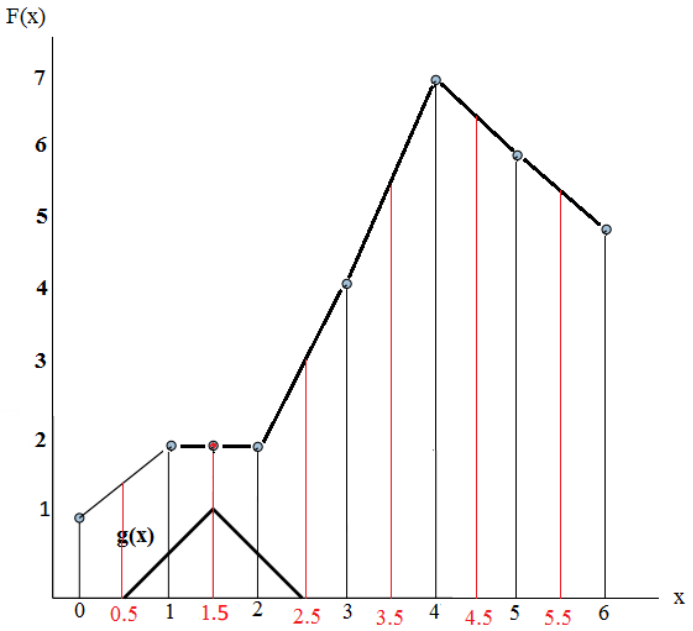


Figure 3: The output of the 1D-dimensional image f after applying linear interpolation in x-direction.

b) Interpolation kernel

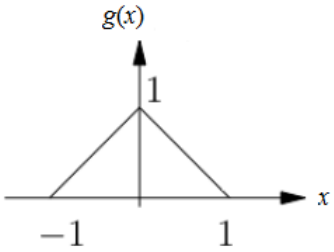


Figure 4: linear interpolation function, i.e. hump $g(x)$. Different choices of g gives different types of interpolation.

Linear interpolation in one demension can be expressed using the following equation

$$F_g(x) = \sum_i g(x-i)f(i) \tag{3}$$

In order to interpolate (or upsample) an image to a higher resolution, we need to select some interpolation kernel (or hump) with which to convolve the image. Different choices of the function $g(x)$ yield different types of interpolation. The function g(x) illustrated in Figure 4 correspond to linear interpolation which represent the `tent or triangle function` and is expressed by

$$g(x) = \begin{cases} 1-|x| & \text{if} \quad |x| \leq 1 \\ 0 & \text{if} \quad |x| > 1 \end{cases} \tag{4}$$

c) Intensity and derivative between the pixels

Assuming that the position for the seven pixels are $0, 1, 2, 3, 4, 5, 6$, and using Equation 3, we need to calculate the intensity and the the derivative of the intensity between the pixels, i.e. at positions

$$x = 0.5, 1.5, 2.5, 3.5, 4.5, 5.5$$

`Linear interpolation` can be expressed by the analytical formula which is derived from equations 3 and 4

$$F_g(x) = (x-i)f(i+1) + (i+1-x)f(i), \quad i < x < i+1 \tag{5}$$

Thus, the intensity between the pixels are obtained by fitting the values of the positions i and those of the intensities $f(i)$ in the analitical formula.

$$F_g(x) = \begin{bmatrix} 1.5 & 2 & 3 & 5.5 & 6.5 & 5.5 \end{bmatrix}$$

Consequently, the derivative of the intensity between the pixels is simply obtained by differentiating $F(x)$, i.e.

$$\frac{\partial F_g(x)}{\partial x} = 1f(i+1) - 1f(i) = \begin{bmatrix} 1 & 0 & 2 & 3 & -1 & -1 \end{bmatrix}$$

This shows the same result illustrated in Figure **??**, where these derivatives represent the slopes or gradients of the lines that connect 2 adjacent intensities.

$F_g(x)$ can be equally expressed when $x = a + k$ by

$$F_g(x) = F(a+k) = af(k+1) + (1-a)f(k), \quad k = 0, ..., 5 \tag{6}$$

We notice that when $a = 0.5$ we simply get the average of the two successive intensities $f(k)$ and $f(k+1)$.

From the derivative property of convolution, the derivative of $F_g(x)$ can be written as

$$\frac{\partial F_g(x)}{\partial x} = \frac{\partial (g * f)}{\partial x} = (\frac{\partial g}{\partial x}) * f = w * f = f * w$$

This is the same as a convolution, where the convolution kernel, i.e. filter, is the partial derivative which can be approximated by

$$w = \begin{bmatrix} 1 & -1 \end{bmatrix}$$

d) The partial derivative of F_g

Here we need to compare two expressions. Since we know from Problem 2c that $dF_g/dx(k) = (f * w)(k)$, we can write out the expressions, s.t.

$$\frac{\partial F_g}{\partial x}(k) = \frac{\partial (g * f)}{\partial x}(k) = \frac{\partial}{\partial x} \sum_i g(k-i)f(i) = (\frac{\partial g}{\partial x} * f)(k) = \sum g'(k-i)f(i) = \sum w(k-i)f(i) = (f*w)(k)$$

this derive from the associative property of convolution, i.e. differentiation is convolution, and convolution is associative and commutative and using the definition of convulotion, i.e.

$$(f * w)(k) = \sum_i f(i)w(k-i)$$

Subtracting one sum from the other gives

$$\sum [w(k-i) - g'(k-i)]f(i) = 0$$

Since this is supposed to be true for all f, we need

$$w(j) = g'(j) \quad \forall j \in Z.$$

Thus, we should conclude by comparing the two equations above that

$$\frac{\partial g}{\partial x}(j) = w(j), \quad \forall j \in Z \tag{7}$$

e) Example

Here we need to find an interpolation function g such that

$$\frac{\partial F_g}{\partial x} = (f * w_2)(k), \quad k = 1, ..., 5, \quad \text{where} \quad w_2 = \begin{bmatrix} 1/2 & 0 & -1/2 \end{bmatrix}$$

As mentioned before, when $x = 0.5 + k$ the values in w corresponds to the derivative of g in the "half points" -0.5, 0.5, 1.5 etc. With $w_2 = [1/2\ 0\ -1/2]$ we get the following:

$$g'(-1 + 0.5) = w_2(-1) = 1/2$$
$$g'(0 + 0.5) = w_2(0) = 0$$
$$g'(1 + 0.5) = w_2(1) = -1/2$$
$$g'(k + 0.5) = w_2(k) = 0 \quad \text{if } |k| > 1$$

In this way, we can create a function $g(x)$ that is piecewise linear and defined in the intervals $-1 < x < 0$, $0 < x < 1$, $1 < x < 2$. From equation 7, we can write

$$\frac{\partial g}{\partial x}(j) = w_2(j) \implies g(x) = \begin{cases} \frac{1}{2}x + C \quad for \quad -1 < x < 0 \\ C \qquad for \quad 0 < x < 1 \\ -\frac{1}{2}x + C \quad for \quad 1 < x < 2 \\ 0 \quad otherwise \end{cases}$$

where C is an arbitrary constant.

5 Classification using Nearest Neighbour and Bayes theorem

Using an imaging system, we get measurements that can be classified in three different classes of objects:

Class 1: 0.4003, 0.3985, 0.3998, 0.3997, 0.4015, 0.3995, 0.3991

Class 2: 0.2554, 0.3139, 0.2627, 0.3802, 0.3247, 0.3360, 0.2974

Class 3: 0.5632, 0.7687, 0.0524, 0.7586, 0.4443, 0.5505, 0.6469

5.1 Nearest Neighbour NN

Using nearest neighbour classification and assuming that the first four measurements in each class are used for `training` and the last three for `testing`, we need to investigate how many measurements will be `correctly classified`. In this way, it is nice to divide the testing data in three vectors as

$$Y1 = \begin{bmatrix} 0.4015 \\ 0.3995 \\ 0.3991 \end{bmatrix}, \quad Y2 = \begin{bmatrix} 0.3247 \\ 0.3360 \\ 0.2974 \end{bmatrix}, \quad Y3 = \begin{bmatrix} 0.4443 \\ 0.5505 \\ 0.6469 \end{bmatrix}$$

Nearest Neighbour algorithm NN classifies using the label of the nearest neighbour. During classification one compares the `distance` between a measured x in testing vectors Y_i with the ones in the training set $(x_1, ..., x_n)$, i.e. solves

$$k = \arg\min_i d(x_i, x) \quad \text{and put} \quad y = k$$

Where y denote the class index. Then, the object is simply assigned to the class of that nearest neighbor. Using Matlab function `knnsearch()`, we find that the nearest neighbors to the first testing data are respectively:

$$0.4003 \quad 0.3997 \quad 0.3985$$

the nearest neighbors to the second testing data are the same one:

$$0.3139$$

the nearest neighbors to the third testing data are respectively:

$$0.4003 \quad 0.5632 \quad 0.5632$$

conclusion 3.1: Here we notice that the first 2 testing data, i.e. of class 1 and 2, are correctly classified but the first measurement in the third testing data is not correctly classified and it must be classified with class 1 and not with class 3.

A generalisation of the above method is to find the k nearest neigbours to the vector x. Then this will be classified in the class y with the most representatives among these k. Otherwise, it is classified as 'unknown'. k-NN is a type of `instance-based learning`, or `lazy` learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

5.2 Gaussian distributions

In this task, assuming that the measurments have `normal distribution` with two parameters, mean m and standard deviation σ, we need to find the `maximum a posteriori` classification of the measurements. The parameters of the normal distribution for the three classes are given as following:

- For class 1: $m_1 = 0.4$, $\sigma_1 = 0.01$
- For class 2: $m_2 = 0.3$, $\sigma_1 = 0.05$
- For class 3: $m_3 = 0.5$, $\sigma_1 = 0.02$

The maximum a posteriori classifier is obtained as selecting the class j that maximizes the `posterior probability`, i.e.

$$j = \arg\max_k P(y = k|x) = \arg\max_k P(x|y = k)P(y = k) \tag{8}$$

As usually, we use Bayes rule to express our posterior probability $P(y = k|x)$

$$P(y = k|x) = \frac{P(x|y = k)P(y = k)}{P(x)} \propto P(y, x) = P(x|y = k)P(y = k)$$

where the `total probability` is in the denominator $P(x) = \sum_j P(x|y = j)P(y = j)$. As this term normalizes the posterior and does not affect the the maximization arg max we can drop it. In addition, we can also ignore the `a priori probability` $P(y = k)$ because it is assumed in the problem that all classes are as likely to occur. Thus, the MAP classification in 8 of a measured x can be reduced to the following:

$$j = \arg\max_k P(y = k|x) = \arg\max_k P(x|y = k)$$

Using the Matlab function `normpdf()`, we can get the normal probability density function (pdf) of the normal distribution of every measured x in 3 different combinations according to the parameters of every class, as shown in Listing 1. In this way, we can investigate if the measuements are correctly classified and to classify them in case they are not .

Conclusion

Looking at the results illustrated below, we can conclude that the measurements in classe 1 and classe 3 are correctly classified because they have the highest values with their own parameters. But measurement 4 of class 2 are not correctly classified because it has higest normpdf with the parameters of class 1, as shown in line 16 in Listing 1. Thus, The 4th element of the 2nd class should be in class 1.

Listing 1: The results of calculating the normpdf() of all measirements in the 3 different combinations.

```
1  % parameters of class 1:
2  m1 = 0.4; sigma1 = 0.01;
3  % parameters of class 2:
4  m2 = 0.3; sigma2 = 0.05;
5  % parameters of class 3:
6  m3 = 0.5; sigma3 = 0.2;
7
8  % Comparing class 1 in 3 different parameters
9  pdf_11 = normpdf(c1,m1,sigma1);
10 % pdf_11 = 39.8763  39.4479  39.8862  39.8763  39.4479  39.8444  39.7330
11 pdf_12 = normpdf(c1,m2,sigma2);
12 % pdf_12 = 1.0669  1.1461  1.0885  1.0928  1.0165  1.1016  1.1192
13 pdf_13 = normpdf(c1,m3,sigma3)
14 % pdf_13 = 1.7616  1.7537  1.7594  1.7590  1.7669  1.7581  1.7564
15
16 % Comparing class 2 in 3 different parameters
17 pdf_21 = normpdf(c2,m1,sigma1);
18 % pdf_21 = 0.0000  0.0000  0.0000  5.6183  0.0000  0.0000  0.0000
19 pdf_22 = normpdf(c2,m2,sigma2);
20 % pdf_22 = 5.3600  7.6764  6.0408  2.2042  7.0623  6.1570  7.9681
21 pdf_23 = normpdf(c2,m3,sigma3)
```

```
% pdf_23 = 0.9442   1.2938   0.9867   1.6671   1.3585   1.4252   1.1941

%  Result: The 4th element of the 2nd class should be in class 1

% Comparing class 3 in 3 different parameters
pdf_31 = normpdf(c3,m1,sigma1);
% pdf_31 = 0.0000   0.0000   0.0000   0.0000   0.0022   0.0000   0.0000
pdf_32 = normpdf(c3,m2,sigma2);
% pdf_32 = 0.0000   0.0000   0.0000   0.0000   0.1240   0.0000   0.0000
pdf_33 = normpdf(c3,m3,sigma3)
% pdf_33 = 1.8976   0.8090   0.1630   0.8647   1.9188   1.9321   1.5231
```

6 Classification - Scene of 0 and 1

NOTE: Make sure not to round any number until you reach the end of the calculations.

By studying a large number of noise free realizations of a scene generated by some random mechanism, it has been found that we can assume 3 classes, i.e. $\omega_y = \{1, 2, 3\}$ such that

$$\begin{matrix}0 & 1\\ 1 & 0\end{matrix} \text{ has prob } \frac{1}{4}; \qquad \begin{matrix}1 & 0\\ 0 & 1\end{matrix} \text{ has prob } \frac{1}{2}; \qquad \begin{matrix}1 & 1\\ 1 & 0\end{matrix} \text{ has prob } \frac{1}{4}.$$

It is assumed in the problem that the images has been distorted by noise in the sense that there is ϵ probability for error in the value of a pixel. It is also assumed that errors in different pixels are independent.
Using Bayes theorem, we need to calculate a MAP, maximum a posteriori, estimation of

$$\begin{matrix}0 & 1\\ 1 & 1\end{matrix}$$

As usually the `a posteriori probabilities` can be calculated using the `Bayes` rule

$$P(y = j|x) = \frac{P(x|y = j)P(y = j)}{P(x)} \tag{9}$$

Case a) $\epsilon = 5\%$

P(observing 0 | the correct value is 1) = P(observing 1 | the correct value is 0) = ϵ.
First we calculate for each class the product between the `measurement probability` and the `a priori probability`, i.e. the `joint probability`

$$p(x, y) = p(x|y = j)p(y = j)$$
$$p(x|y = 1)p(y = 1) = 0.05^1 * 0.95^3 * 0.25 = 0.0107$$
$$p(x|y = 2)p(y = 2) = 0.05^3 * 0.95^1 * 0.5 = 5.9375e - 005$$
$$p(x|y = 3)p(y = 3) = 0.05^2 * 0.95^2 * 0.25 = 5.6406e - 004$$

Now we can calculate the `total probability` $p(x)$

$$p(x) = \sum_j p(x|y = j)p(y = j) = 0.0107 + 5.9375e - 005 + 5.6406e - 004 = 0.0113$$

We may as well calculate the a posteriori probabilities using Bayes rule for each j:

$$P(y = 1|x) = \frac{p(x|y = 1)p(y = 1)}{p(x)} = \frac{0.0107}{0.0113} = 0.9469$$
$$P(y = 2|x) = \frac{p(x|y = 2)p(y = 2)}{p(x)} = \frac{5.9375e - 005}{0.0113} = 0.0053$$
$$P(y = 3|x) = \frac{p(x|y = 3)p(y = 3)}{p(x)} = \frac{5.6406e - 004}{0.0113} = 0.0499$$

We should remember that the probabilites sum to 1.
From this result we can conclude that the probability that the image is from class y = 1 is 0.95 which is the MAP according to the fact that

$$j = \arg\max_k P(y = k|x)$$

Case b) $\epsilon = 50\%$

In the same procedure we do calculations when $\epsilon = 50\%$.

$$p(x|y=1)p(y=1) = 0.5^1 * 0.5^3 * 0.25 = 0.0156$$

$$p(x|y=2)p(y=2) = 0.5^3 * 0.5^1 * 0.5 = 0.0313$$

$$p(x|y=3)p(y=3) = 0.5^2 * 0.5^2 * 0.25 = 0.0156$$

Now we can calculate $p(x)$

$$p(x) = \sum_j p(x|y=j)p(y=j) = 0.0156 + 0.0313 + 0.0156 = 0.0625$$

calculate the a posteriori probabilities using Bayes rule for each j:

$$P(y=1|x) = \frac{p(x|y=1)p(y=1)}{p(x)} = \frac{0.0156}{0.0625} = 0.2496$$

$$P(y=2|x) = \frac{p(x|y=2)p(y=2)}{p(x)} = \frac{0.0313}{0.0625} = 0.5008$$

$$P(y=3|x) = \frac{p(x|y=3)p(y=3)}{p(x)} = \frac{0.0156}{0.0625} = 0.2496$$

We should remember that the probabilites sum to 1.

From this result we can conclude, according to maximum a posteriori, that the probability that the image is from class y = 2 is 0.5 which is the MAP.

7 Classification - Black vertical line

In this task we are dealing with a binary 4×4 image of a scene with a vertical line. *In the correct image all pixels would be white except one vertical row with blak pixels.* It is assumed that errors in different pixels are independent with

P(white | line) = P(black | not line) = ϵ.

Consequently, we can say that

P(black | line) = P(white | not line) = $1 - \epsilon$.

we need to calculate the maximum a posteriori estimation MAP of the following image, in Figure 5, when $\epsilon = 0.2$

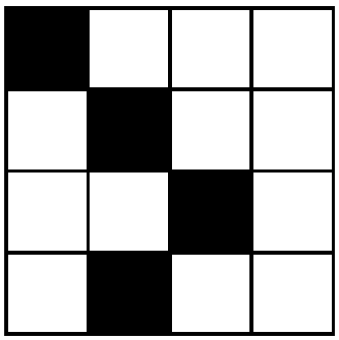


Figure 5: The binary 4×4 image with a vertical line

Here we can distinguish between 4 classes which correspond to the four columns in in Figure 5. It is assumed that the a priori probability for the line to be located in column 1 or 4 is 0.3 each and that one for the line to be in column 2 or 3 is 0.2 each.

First we calculate for each class the product between the `measurement probability` and `a priori probability`, i.e. the `joint probability`

$$p(x, y) = p(x|y = j)p(y = j)$$

We should remember that when computing $p(x|y = i)$ we don't only multiply 4 different numbers, representing whether the pixels in column i are correct or not if we compare black vertical row x to i. However, the same probabilities goes for the rest of the pixels which all should be white. Thus, $p(x|y = i)$ should be a product of 16 different factors, one for each pixel:

$$p(x|y = 1)p(y = 1) = 0.2^6 * 0.8^{10} * 0.3 = 2.0616e - 006$$

$$p(x|y = 2)p(y = 2) = 0.2^4 * 0.8^{12} * 0.2 = 2.1990e - 005$$

$$p(x|y = 3)p(y = 3) = 0.2^6 * 0.8^{10} * 0.2 = 1.3744e - 006$$

$$p(x|y = 4)p(y = 4) = 0.2^8 * 0.8^8 * 0.3 = 1.2885e - 007$$

Now we can calculate the `total probability` $p(x)$

$$p(x) = \sum_j p(x|y = j)p(y = j) = 2.0616e{-}006{+}2.1990e{-}005{+}1.3744e{-}006{+}1.2885e{-}007 = 2.5555e{-}005$$

Now we may as well calculate the a posteriori probabilities using Bayes rule for each j:

$$P(y = 1|x) = \frac{p(x|y = 1)p(y = 1)}{p(x)} = \frac{2.0616e - 006}{2.5555e - 005} = 0.0807$$

$$P(y = 2|x) = \frac{p(x|y = 2)p(y = 2)}{p(x)} = \frac{2.1990e - 005}{2.5555e - 005} = 0.8605$$

$$P(y = 3|x) = \frac{p(x|y = 3)p(y = 3)}{p(x)} = \frac{1.3744e - 006}{2.5555e - 005} = 0.0538$$

$$P(y = 4|x) = \frac{p(x|y = 4)p(y = 4)}{p(x)} = \frac{1.2885e - 007}{2.5555e - 005} = 0.0050$$

We verify that the probabilites sum to 1 = 0.0807 + 0.8605 + 0.0538 + 0.0050

Classifying according to maximum a posteriori, we can conclude that the probability that the image is from class y = 2 is 0.8605 which is the most probable image, i.e. MAP.

8 Classification - B08x

In this problem, we need to classify an observed 5×3 image x using maximum a posteriori probabilities (MAP) when images for three different classes are given, as illustrated in Figure 6.

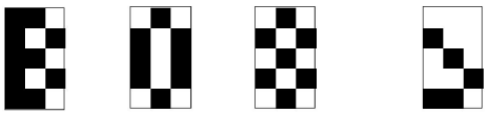


Figure 6: Mean images for three different classes, and an observed image x.

The `a priori` probabilities, i.e. the probabilities that the class is $y = j$ are

- To be a 'B': p(y = 1) = 0.25
- To be a '0': p(y = 2) = 0.4
- To be a '8': p(y = 3) = 0.35

It is assumed that there exist errors in pixels in image x because of a measurment noise:

- p(being white | originally black) = 0.25.
  $\implies$ p(being black | originally black) = 1 - 0.25 = 0.75
- p(being black | originally white) = 0.35
  $\implies$ p(being white | originally white) = 1 - 0.35 = 0.65

First we calculate for each class the product between the `measurement probability` and `a priori probability`, i.e. the `joint probability`

$$p(x, y) = p(x|y = j)p(y = j)$$

$$p(x|y = 1)p(y = 1) = 0.25^5 * 0.35^0 * 0.65^5 * 0.75^5 * 0.25 = 6.7222e - 006$$
$$p(x|y = 2)p(y = 2) = 0.25^5 * 0.35^2 * 0.65^5 * 0.75^3 * 0.4 = 2.3423e - 006$$
$$p(x|y = 3)p(y = 3) = 0.25^3 * 0.35^1 * 0.65^7 * 0.75^4 * 0.35 = 2.9689e - 005$$

Now we can calculate the `total probability` $p(x)$

$$p(x) = \sum_j p(x|y = j)p(y = j) = 6.7222e - 006 + 2.3423e - 006 + 2.9689e - 005 = 3.8754e - 005$$

Now we may as well calculate the a posteriori probabilities using Bayes rule for each j:

$$P(y = 1|x) = \frac{p(x|y = 1)p(y = 1)}{p(x)} = \frac{6.7222e - 006}{3.8754e - 005} = 0.1735$$

$$P(y = 2|x) = \frac{p(x|y = 2)p(y = 2)}{p(x)} = \frac{2.3423e - 006}{3.8754e - 005} = 0.0604$$

$$P(y = 3|x) = \frac{p(x|y = 3)p(y = 3)}{p(x)} = \frac{2.9689e - 005}{3.8754e - 005} = 0.7661$$

We see that the probabilites sum to 1 = 0.1735 + 0.0604 + 0.7661

Classifying according to maximum a posteriori, we can conclude that the probability that image x is from class y = 3 is 0.7661 which is the most probable image, i.e. MAP.

9 The OCR system - part 2 - Feature extraction

As part of developping and testing the OCR system, i.e. `Optical Character Recognition`, in this task we need to write a matlab function `segment2features()` that given a binary image matrix B (thus only with zeros and ones) as input, returns a `feature vector`. Here, were assume that the features are represented as a column vector

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

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.

To solve this problem, we need to study the region of pixels that are equal to 1. then we define 9 different features (numbers) that can be used to classify which letter the region corresponds to. The feature, i.e. the value, should be independent of the position the region has, such that, if the regions are shifted in the x or y direction, then we will obtain the same features.

We should remember that when a number of feature candidates may have been generated, using all candidates will easily lead to `over-training` (unreliable classification of new data). In this task, the following features are selected and the Matlab code is shown in Listing 2:

1. **Moment**
2. **Vertical histogram**

3. **Sum of pixels**
4. **Center of mass**
5. **Orientation**: returns the angle in degrees
6. **Crossings**: Scan through the rows and columns to obtain the horizontal and vertical crossing times of a character.
7. **Numbre of holes**: Using the Matlab function `EUL = bweuler(BW)` which 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.
8. **Average of row sum**
9. **Average of column sum**

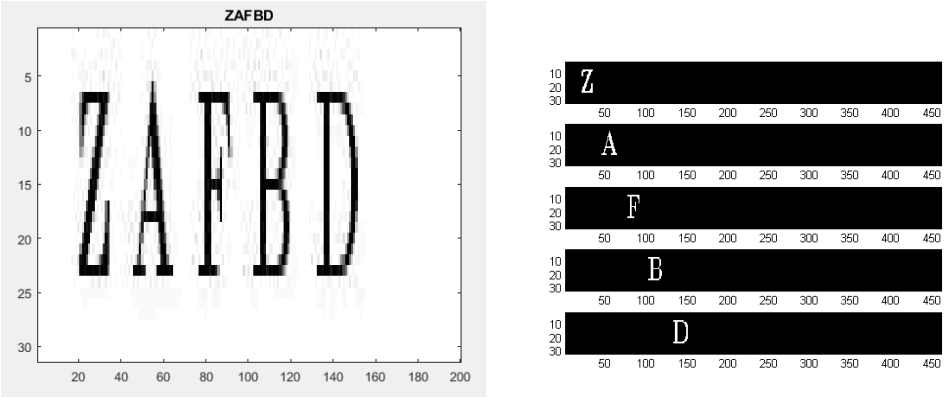
Listing 2: Matlab function that returns a vector of features of an image.

```
1
2  function x = mySegment2features(B)
3  % features = segment2features(I)
4
5  % initialize the features vector x
6  %features = randn(6,1);
7  x = zeros(9,1);
8
9  % Given a binary image B (thus only with zeros and ones), 1 => object pixel
10 % 0 => background pixel.
11 % study the region of pixels that are equal to 1.
12 % Use your imagination to define at least 6 different features (numbers),
13 % that can be used to classify which letter the region corresponds to.
14
15 % Localisation: Identify the region of pixels that are equal to one
16 [yPosition xPosition] = find( B == 1 );
17 xmax = max(xPosition);
18 xmin = min(xPosition);
19 ymax = max(yPosition);
20 ymin = min(yPosition);
21 regionOfB = B(ymin:ymax,xmin:xmax);
22
23 % size of the image region
24 [m,n] = size(regionOfB);
25
26 % Features extraction
27 i = 1;
28
29 % Feature 1: moment
30 % SIGMA = moment(X,ORDER) returns the ORDER-th central sample moment of
31 % the values in X.  For vector input, SIGMA is MEAN((X-MEAN(X)).^ORDER).
32 % For a matrix input, moment(X,ORDER) returns a row vector containing the
33 % central moment of each column of X.
34 x(i) = mean(moment(regionOfB,3));
35 i = i + 1;
36
37 % Feature 2: vertical histogram
38 n2 = floor(n/2);
39 % A histogram is the frequency of occurrence vs. gray level
40 x(i) = sum(sum(regionOfB(:,n2:n)));        % XXXXXX
41 i = i + 1;
42
43 % Feature 3: the sum of pixel values
44 x(i) = sum(sum(regionOfB));
```

```
45  i = i + 1;
46
47  % Feature 4: Center of mass
48  % STATS = regionprops(BW,PROPERTIES) measures a set of properties for
49  % each connected component (object) in the binary image BW, which must be
50  % a logical array; it can have any dimension.
51  % Calculate centroids for connected components in the image
52  stats = regionprops(regionOfB,'Centroid');
53  sx = stats.Centroid(1);
54  sy = stats.Centroid(2);
55  x(i) = sx + sy;
56  i = i + 1;
57
58  % Feature 5: Orientation
59  % Angle between the x-axis and the major axis of the ellipse that has the
60  % same second-moments as the region, returned as a scalar.
61  % The value is in degrees.
62  stats = regionprops(regionOfB,'Orientation');
63  x(i) = abs(stats.Orientation);
64  i = i + 1;
65
66  % Crossing feature 6: Scan through the rows and columns to obtain the
67  % horizontal and vertical crossing times of a character.
68
69  % Vertical crossings: for any column j = 0,1,2,...,n
70  % number of vertical crossings in column j
71  numVerCross = 0;
72  pixel = 0;
73  j = 0;
74  % for any row j = 0,1,2,...,n
75  while j < m
76      while pixel == 0 && j < m
77          j = j + 1;
78          pixel = regionOfB(j,floor(n/2));
79          % floor(X) rounds the elements of X to the
80          % nearest integers towards minus infinity.
81      end
82
83      while pixel == 1 && j < m
84          j = j + 1;
85          pixel = regionOfB(j,floor(n/2));
86      end
87
88      numVerCross = numVerCross + 1;
89  end
90
91  if pixel == 0
92      numVerCross = numVerCross - 1;
93  end
94
95  % Horizontal crossings
96  % number of horizontal crossings in row i
97  numHorCross = 0;
98  pixel = 0;
99  j = 0;
100
101 % for any column j = 0,1,2,...,n
102 while j < n
103     while pixel == 0 && j < n
104         j = j + 1;
105         pixel = regionOfB(floor(m/2),j);
```

```
106         end
107
108         while pixel == 1 && j < n
109             j = j + 1;
110             pixel = regionOfB(floor(m/2),j);
111         end
112
113         numHorCross = numHorCross + 1;
114     end
115
116     if pixel == 0
117         numHorCross = numHorCross - 1;
118     end
119
120     x(i) = (numHorCross + numVerCross);
121     i = i + 1;
122
123     % Feature 7: Number of holes using bwEuler() function
124     % EUL = bweuler(BW,N) returns the Euler number for the binary
125     % image BW. EUL is a scalar whose value is the number of
126     % objects in the image minus the total number of holes in those
127     % objects.  N can have a value of either 4 or 8, where 4
128     % specifies 4-connected objects and 8 specifies 8-connected
129     % objects; if the argument is omitted, it defaults to 8.
130     x(i) = bweuler(regionOfB);
131     i = i + 1;
132
133     % Feature 8: Average of row sum
134     % S = mean(X) is the mean value of the elements in X if X is a vector.
135     % For matrices, S is a row vector containing the mean value of each column.
136     x(i) = mean(sum(regionOfB'));
137     i = i + 1;
138
139     % Feature 9: Average of column sum
140     x(i) = mean(sum(regionOfB));
141     i = i + 1;
142
143 end
```

When running my segmentation function and then `segment2features` on the input image shown in Figure 7a, similar feature vectors for the same character and different feature vectors for different characters.



(a) (b)

Figure 7: a) The original input image. b) The output of the segmented images after running the segmentation function.

Then for testing the function we use also the benchmark script. An output of the resulting test is

illustrated in Listing 3.

Listing 3: The obtained feature vectors after using the benchmark script.

```
Studying the character A
There are 4 examples in the database.
The feature vectors for these are:

ans =

    'Moment'                  0.05209     0.051404     0.051404     0.051404
    'Vertical Histogram'           67           67           67           67
    'Sum of pixels'                93           94           94           94
    'Center of mass'           22.183       21.947       21.947       21.947
    'Orientation'              80.066        80.77        80.77        80.77
    'Crossings'                     4            4            4            4
    'Numbre of holes'               0            0            0            0
    'Average of row sum'       5.1667       5.2222       5.2222       5.2222
    'Average of column sum'    4.8947       4.9474       4.9474       4.9474


Studying the character B
There are 1 examples in the database.
The feature vectors for these are:

ans =

    'Moment'                  0.038002
    'Vertical Histogram'            66
    'Sum of pixels'                126
    'Center of mass'           18.675
    'Orientation'              71.409
    'Crossings'                      5
    'Numbre of holes'               -1
    'Average of row sum'       7.4118
    'Average of column sum'    7.4118


Studying the character D
There are 2 examples in the database.
The feature vectors for these are:

ans =

  'Moment'                  0.047886     0.047564
  'Vertical Histogram'            59           60
  'Sum of pixels'                120          121
  'Center of mass'           19.142       19.331
  'Orientation'              5.8709       4.0527
  'Crossings'                     4            4
  'Numbre of holes'               0            0
  'Average of row sum'       7.0588       7.1176
  'Average of column sum'    6.3158       6.3684


Studying the character E
There are 4 examples in the database.
The feature vectors for these are:

ans =

   'Moment'                 0.055643     0.055643     0.052056     0.051979
```

```
    'Vertical Histogram'          46          46          48          47
    'Sum of pixels'              106         106         108         107
    'Center of mass'           17.34      17.349      17.463      17.355
    'Orientation'              84.698      84.361      85.047      86.338
    'Crossings'                     5           5           5           5
    'Numbre of holes'               1           1           1           1
    'Average of row sum'       6.2353      6.2353      6.3529      6.2941
    'Average of column sum'     6.625       6.625        6.75      6.6875

Studying the character F
There are 2 examples in the database.
The feature vectors for these are:

ans =

    'Moment'                0.061307    0.061307
    'Vertical Histogram'          33          33
    'Sum of pixels'               90          90
    'Center of mass'           15.044      15.044
    'Orientation'              69.744      69.744
    'Crossings'                     5           5
    'Numbre of holes'               1           1
    'Average of row sum'       5.2941      5.2941
    'Average of column sum'         6           6


Studying the character G
There are 3 examples in the database.
The feature vectors for these are:

ans =

    'Moment'                0.041997    0.043558     0.04064
    'Vertical Histogram'          45          43          45
    'Sum of pixels'               96          94          97
    'Center of mass'           18.063      17.915      17.814
    'Orientation'              30.458        30.3      23.508
    'Crossings'                     3           3           3
    'Numbre of holes'               1           1           1
    'Average of row sum'       5.6471      5.5294      5.7059
    'Average of column sum'    5.3333      5.2222      5.3889


Studying the character H
There are 2 examples in the database.
The feature vectors for these are:

ans =

    'Moment'                0.05691     0.05691
    'Vertical Histogram'          68          68
    'Sum of pixels'              134         134
    'Center of mass'            19.5        19.5
    'Orientation'                  0           0
    'Crossings'                    3           3
    'Numbre of holes'              1           1
    'Average of row sum'       7.8824      7.8824
    'Average of column sum'       6.7         6.7
```

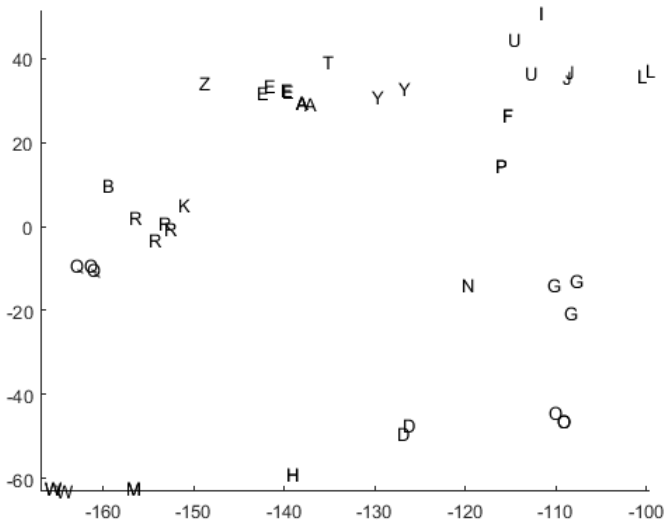


Figure 8: The output after running the benchmark script that contains 50 different letters spread everywhere, i.e. different positions

10 Appendix

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

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

[6] Mathworks Documentation, knnsearch, Find k-nearest neighbors using data [Online]. Available: `https://www.mathworks.com/help/stats/knnsearch.html#mw_9be1c136-c5c3-4071-a1ce-f92f7a837f3b`

[7] Mathworks Documentation, normpdf, Normal probability density function [Online]. Available: `https://www.mathworks.com/help/stats/normpdf.html#f1131060`

[8] Image Processing Fundamentals, Derivative-based Operations [Online]. http://www.mif.vu.lt/atpazinimas/dip/FIP/fip-Derivati.html

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

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