EFME 2013 LU Exercise 1

Exercise due: 10th November 2013, 23:55

Abstract

The goal of this exercise is to give you some introduction into MATLAB and to learn how to extract and use features from images and use them in a decision algorithm. You are going to implement your first pattern recognition algorithm, the K nearest neighboring algorithm (K-NN).

1 How to submit your report

You should make the report of the exercise available as a PDF document. The hand-in for this and later laboratory exercises is done by using TUWEL before the hard deadline (see above) containing the following (please follow this standard):

- In the subject: 'EFME-', your group and exercise number, e.g. 'EFME-Gr0-Ex0' for the zero group and zero exercise.
- As attachments (.7z or .zip): the PDF document with the ALL the MATLAB code necessary to RUN your solution (including your chosen images, and (important) a file that runs ALL your MATLAB code).

You may write the PDF document in English or in German. Include in the document your results and most importantly, a discussion of the results. DO NOT forget to attach the MATLAB code in the same zip-file. Be careful: If the attached MATLAB does NOT run, we will reject your exercise completely. We ask you to put in the MATLAB code also a matlab file (e.g. main.m or exercise.m etc) that runs ALL your solutions (just by one call to it). It is NOT necessary to include a copy of all the code in the PDF document, although key parts necessary to explain a point can be included. It is necessary to comment the code in details. More details are available on the TUWEL web page.

2 MATLAB

If you have not used MATLAB yet, or only used it very little, then your first task is to familiarize yourself with MATLAB. A good introduction is given by the MATLAB-Primer available on the TUWEL Download Area web page. A summary of the most important MATLAB commands is given in the MATLAB Introduction also available on this page. Read this introduction and try out some of the commands. You should become familiar enough with MATLAB to use it in solving the EFME exercises. Nothing about this part has to be included in the report. This part will not be evaluated.

3 Feature Extraction

The EFME Slides and the MATLAB online help on the `regionprops` command are useful references.

3.1 Introduction (8 points)

The widely tested MPEG7 CE-Shape-1 database [2] consists of 1400 silhouette images from 70 classes. Each class has 20 different shapes. Figure 1 shows examples of some different classes. You will find a link to download the database on the TUWEL web page.

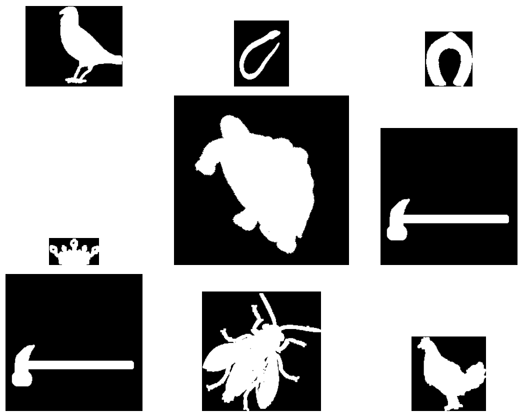

Figure 1: Shapes of MPEG-7 CE-Shape-1 database.

The aim of this exercise is to find a set of blob features (e.g. formfactor, roundness, compactness, convex hull etc.) which allow to discriminate at least between five object classes. This means features with values that are:

- very similar for objects of the same type,
- different for objects of different types.

It is important to notice that the objects are distributed in random places in the image, and that they have random orientations and scale (as shown in the Figure 1). Based on this information, it should already be possible to exclude some features as being useless for this application. Which are they? Stated technically, we are looking for features which are invariant with respect to changes in position, orientation and scale.

3.1.1 How to proceed

Some hints on using MATLAB to calculate image features are given in this section. MATLAB commands are given in the `typewriter` font. Use the MATLAB `help` function to get more information on how to use the commands.

1. Choose at least five object classes (e.g. apple, bottle, car, key, octopus, etc.) and load the images into MATLAB (`imread`). Since some images are of different size use for instance a cell array to store the images. Cell arrays can be used to store a sequence of matrices of different sizes. The following code example supports loading all bat pictures:

```
D = dir ;
I = cell( 20 , 1 ) ;
j = 1 ;
for i = 1 : size( D, 1 )
    if D( i ).isdir == 0 && strncmp( D( i ).name( 1 : 3 ) , 'bat' , 3)
      disp( D( i ).name ) ;
      I_temp = imread ( [D( i ).name ] ) ;
      I{j} = im2bw( I_temp , graythresh ( I_temp ) ) ;
      j = j +1;
    end
end
```

2. Use `im2bw` to convert the images into a binary format. The next step is to label the images (`bwlabel`) in order to compute features for each blob. Hint: you can display the images with `imshow(imgage)`.

3. Use the `regionprops` command to calculate features for each labeled blob. Let us say that you put the results into a variable called `STATS` using the command `STATS = regionprops(L,'all')` . To see the values of all the features calculated for the blob labeled 2, for example, type `STATS(2)`. To see the results of only one measurement, the area for example, type `STATS(2).Area`. To see the area measurements for all the blobs, type `STATS(:).Area`. Note that not all the features listed in the Lecture are directly available with the `regionprops` command.

4. To calculate the shape descriptors, such as the *formfactor*, *roundness*, etc., some of these blob features have to be combined using arithmetic functions.

3.2 Classification algorithm (4 points)

Draw/write up a simple algorithm containing the steps to go through to recognize each type of your chosen objects. You may use thresholds on feature values, so a step such as `if formfactor > 0.5 then Object1 else ...` is possible. In the next excercise (k-NN), you will be using an algorithm which avoids the manual setting of thresholds on feature values. Be sure to answer the following in the report:

- Are the features used by your algorithm position, orientation and scale invariant? - Invarcianes of features.
- How well does your algorithm perform (percentage of objects correctly classified)? - Evaluation of the algorithm.

4 k-NN Classifier (8 points)

Write a k-NN classifier in MATLAB **by your own! Very important**. It should use a set of pre-classified feature vectors of any dimension as the training set, and then provide the classes for a second set of such vectors.

Use your classifier on the shape classes from the shape database that you have used in the Exercise 3.1. Use the feature vectors that you extracted for each shape as input to the k-NN classifier. Compare the percentage of shapes correctly classified by the k-NN classifier and by your hand-tuned classified from Exercise 3.2. You should not train and test your classifier on the same data. Why? Why would doing this be extremely stupid for the k-NN classifier? The usual approach is to divide the data into a training and test set. However, doing this usually only works well if you have a large amount of training data. Why? For this shape classification problem, each class only has examples of 20 different shapes. It is therefore difficult to find a perfectly fair division into a training set and test set. A solution is to make use of Leave-one-out cross-validation to evaluate your classifier. This is explained in Section 4.1. You should use this solution. Compare the percentage when different values of k are used. Make a table and draw a graph showing the percentage error (i.e. of misclassified samples) as a function of k (starting from $k = 1$, i.e. having the NN algorithm). If you wish, try out different subsets of features too, and compare the results. Your k-NN Matlab implementation should be as short as possible (think about vectorizing).

4.1 How to: Leave-one-out cross-validation

For getting good performance with classification algorithms, one should use as much training data as possible. However, to get an unbiased estimate of the generalisation of a classification algorithm, it should not be tested on data that was used for training. This leads to the problem of how much data should be used for training and how much kept aside for testing. A solution to this problem is Leave-one-out cross-validation (LOOCV). Leave-one-out cross-validation works as follows: let's say that one has a data set containing n feature vectors x_i, i.e. $x_1, x_2, x_3, x_4, ..., x_n$. The classifier is trained n times, each time leaving out one of the training vectors, with which it is then tested. In algorithmic form:

1. Train with $x_2, x_3, x_4, ..., x_n$, test with x_1.

2. Train with $x_1, x_3, x_4, ..., x_n$, test with x_2.
3. Train with $x_1, x_2, x_4, ..., x_n$, test with x_3.
4. and so on ...

The classification error is then simply the number of feature vectors which were classified incorrectly divided by the number of feature vectors. For classifiers which have a long training time, using LOOCV takes a long time. However, for the k-NN classifier, it can be implemented very efficiently. As training simply entails saving all the feature vectors from the training set, it is easy to remove each feature vector in turn from the training set and then classify it. It is not even necessary to remove it, one can simply ignore the training set feature vector with a distance of zero to the feature vector which is being classified.

Extra points: K-fold cross-validation is also possible. If removing each feature vector in turn would take excessively long (due to a very large dataset or a classifier that requires a long training time), one can divide the training set up into K groups of feature vectors. The training is then done K times, with each time a different group being removed from the training set to be used for testing.

4.2 (5 points) Paper Summary

Please write (at least) one page summary of the paper 'When Is Nearest Neighbor Meaningful' by Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft [1]. Please read carefully the paper and start a discussion among the group members. It is important to put it in your own words, and only the main message of the paper. No need for formulas or tables. You can find the paper in the TUWEL. The font should be (max.) 11 point. No need for subtitle.

4.3 Outcome of this exercise

By the end of this exercise, you should be able to explain the following:

- Feature extraction from binary and gray value images in MATLAB.
- What is meant by features which are invariant to changes in:
  - position,
  - orientation,
  - scale.

  and which features have these properties.
- Building a pattern recognition algorithm by hand
- NN and K-NN algorithm

References

[1] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *In Int. Conf. on Database Theory*, pages 217–235, 1999.

[2] L.J. Latecki, R. Lakamper, and T. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 424 – 429, 2000.