PROJECT 'CLASSIFICATION OF RINGS, NUTS AND BOLTS'

INTRODUCTION

In this project, a classification system will be developed for classifying objects that belong to four different categories: *ring*, *nut-6* (6-sided), *nut-4* (4-sided), and *bolt*. A vision system is available that acquires images of these objects. Using some digital image processing techniques (not part of this project) the images are segmented. After that, each imaged object is represented by a connected component in the resulting binary (logical) image. Figure 1 shows already segmented images containing rings, nuts-6, nuts-4 and bolts. These images are available for training and evaluation, thus providing us with a labelled dataset of 121 objects per class.

The classification will be based on so-called normalized Fourier descriptors. These are measurements describing the shapes of the contours of the objects. The software provided with the project can produce many descriptors per objects, but it remains a question whether all of them are discriminating.

The first goal of the design is to find a classifier that strives for minimal error rate. A low computational load is also desirable, but not the main goal. After having decided which classifier meets these requirements its performance should be specified in terms of an estimated error rate (including an indication of the uncertainty of this estimate), and in terms of a confusion matrix.

Literature:

- Chapter 2
- Chapter 5
- Chapter 6
- Section 9.1

Requirements:

- Matlab 6.5 or higher
- PRTools Version CPESE
- Image processing toolbox
- Datafile: CPESE_PROJ_CLASS1.ZIP (includes images and `ut_contourfft.m` code)

a) b) c) d)

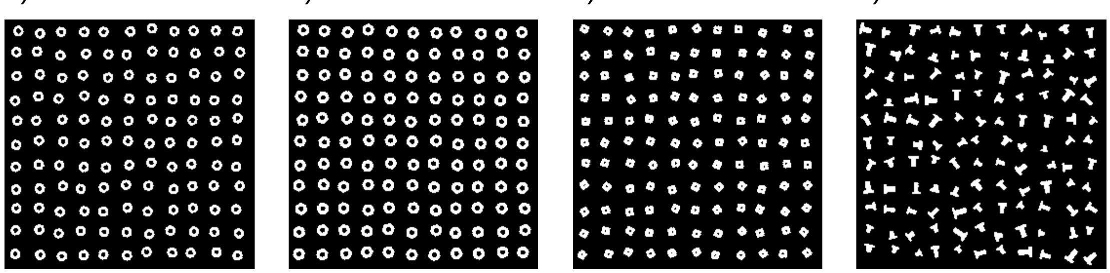

Figure 1. The images of the objects used for training and evaluation

a) rings b) 6-sided nuts c) 4-sided nuts d) bolts

THE DESIGN

Measurement vectors

The software that is provided within this project can calculate up to 64 descriptors denoted by Z_k, where k ranges from -31 up to $+32$. The descriptors are normalized such that they are independent from the orientation and the size. However, Z_0 and Z_1 should not be used, because Z_0 does not depend on the shape (but rather on the position) and Z_1 is always one (because it is used for the normalization). The given Matlab function, `ut_contourfft`, offers the possibility to calculate only a selection of the available descriptors. For instance, Figure 2 shows the descriptors of four objects in an (arbitrary) range from $[-10,-1]$ and $[+2,+10]$.

Creating the dataset

Each image in Figure 1 shows the segments of 121 objects. Thus, extraction of the boundary of each segments, and subsequent determination of the normalized Fourier descriptors yields a training set of $4 \times 121 = 484$ labelled vectors, each vector having 62 elements. An image can be transformed into a set of measurement vectors with the following fragment of code:

```
fdlist = [-31:-1 2:32];                          % exclude Z0 and Z1
imrings = imread('rings.tif');                   % open and read the image file
figure; imshow(imrings); title('rings');
[BND,L,Nring,A] = bwboundaries(imrings,8,'noholes');% extract the boundaries
FDS = ut_contourfft(BND,'fdlist',fdlist,'nmag');    % calculate the FDs
Zrings = zeros(Nring,length(fdlist));            % allocate space
for n=1:Nring
    Zrings(n,:) = FDS{n}';                       % collect the vectors
end
```

Likewise pieces of code are needed to get the measurement vectors from the other classes. The filenames of the four images are: `rings.tif`, `nuts6.tif`, `nuts4.tif` and `bolts.tif`. The function `bwboundaries` is from the image processing toolbox. The function `ut_contourfft` accompanies the images.

Section 5.1 describes how to transform the data into PRTools format. (Hint: use `repmat` to create the array with labels. For instance, `repmat('ring',[Nring 1])` creates an array of `Nring` entries containing the string `'ring'`.)

Design strategy

In order to find a good classifier we follow roughly the outline presented in Section 9.1. First we consider a few simple classifiers, such as `ldc` and `qdc`, that can be applied directly to the measurement space. In order to assess the risk of overfitting, we calculate the error rate with and

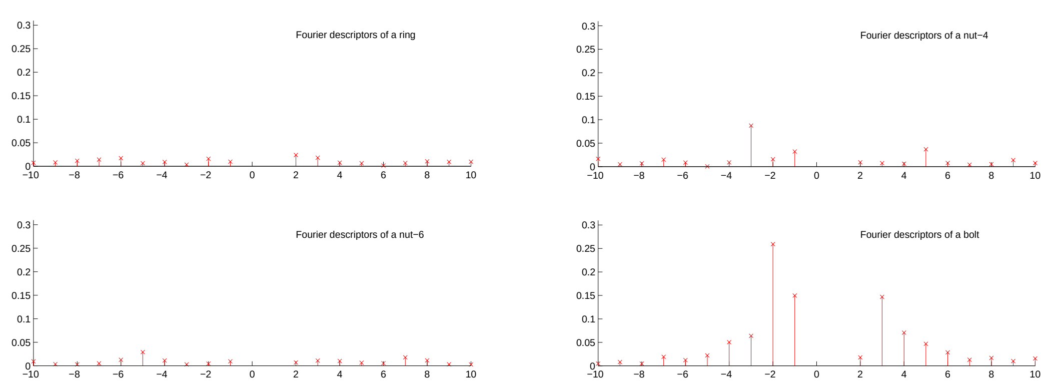

Figure 2 Normalized Fourier descriptors.

without cross-validation (`testc` and `crossval`, respectively).

After this first exercise, more complex classifiers can be tested:

- Parzen classification
- NNR and KNNR classification
- Support vector machines
- Feed-forward neural network

Note that some of these classifiers may need rescaling of the individual elements in order to have comparable units, thus allowing the calculation of a fair distance measure.

These classifiers still use the full measurement space. Thus, the risk of overfitting is still present. Possible countermeasures are:

- Application of regularization techniques to the linear and quadratic classifiers.
- Application of feature selection.
- Application of linear feature extraction, e.g. PCA and extraction based on inter/intra distances.

Feature selection and feature extraction only reduces the measurement space. The classification still has to be done. Therefore, the selection and extraction techniques can be combined with the various classifiers mentioned above.

Note that many of these classification methods rely on the selection of parameters and criterions. For instance, the regularized `ldc` and `qdc` classifiers depend on one or more regularization parameters. The feature selection depends on the strategy to traverse the search tree (branch&bound, forward selection, plus L, take away R, etc) and on the criterion. In order to find the best method an optimization of these design parameters should be accomplished. (See, for instance, Listing 9.7 where the structure of a neural network is optimized).

After having found the most promising classification methods, the corresponding performance must be established. This is not only a specification of the error rate $\hat{E}$ as returned by `crossval`. Also its accuracy, expressed as the standard deviation $\sigma_{\hat{E}}$, should be mentioned. See Section 5.4. $\sigma_{\hat{E}}$ allows us to judge whether the differences between the estimated error rates of the various classifiers are statistically representative.

The confusion matrix gives an impression which objects are likely to be confused. Therefore, specification of the confusion matrix is also of interest.