TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

Diploma Thesis

Neural Object Classification by Pattern Recognition of One Dimensional Data Arrays Which Represent Object Information Transformed by Non-linear Functions

Written at the

ACIN - Automation and Control Institute

Thesis supervisor:

Univ. Prof. Dipl.-Ing. Dr.techn. Favre-Bulle, Bernard

Thesis Co-Advisor:

Dipl.-Ing. Fauaz Labadi

Submitted by

Kayhan Ince

ID-number: 0127059

Wien, 25.Feb 2006 ……....…………..

Erklärung

Ich erkläre, dass ich die Diplomarbeit selbstständig verfasst habe. Diese Diplomarbeit ist bisher weder im In- noch im Ausland in irgend einer Form vorgelegt.

Wien, im Feb 2006

Acknowledgements

I would like to thank my advisor Univ.Prof. Dipl.-Ing. Dr.techn. Bernard Favre-Bulle for his arrangments at ACIN and for his supervision on the thesis. I am very grateful to Dipl. Ing. Fauaz Labadi who initialized the contact and lead the supervision and for reading and correcting my work on this thesis.
I also need to express my gratitude towards all the people who took time to make me feel guilty enough to sit down and start working. It would also be unfair if I forget to mention my parents, my friends for their great support through all phases of life.

Abstract

The grasping and classification of the objects play important role in visual system. For the grasping and classification process, the shapes must be recognized which is based on matching the descriptors of each shape to standard values representing typical shapes and choosing the closest match. The previous works have been done on the neural control of the grasping and the autonomous operation of hyper-redundant manipulators. The main tasks and aims of the thesis is dealing with tentacle case and linear object scanner case problem. Subsequently, the Fourier descriptors are used for the shape matching and the Fourier transformation of the shapes is used in order to perform the grasping process. A special kind of neural network is used in classification problem. In a final manner, the theoretical information are confirmed by MATLAB/SIMULINK simulations.

Contents

List of Figures

1 Introduction

1.1 Overview

This thesis is related with pattern recognition in the shade of neural networks using one dimensional data arrays and several stages in classification process of objects will be discussed.

A small number of links connected to serial chains by joints are described as the hyper redundant manipulators. In our application, 16 joints are used, since these provide the sixteen degrees of freedom that are necessary to achieve the grasping process. The recognition, in this thesis is based on artificial neural networks which are parallel computing systems derived from biological nervous systems.

In [Bus02], Busch has already taken a step towards an autonomous control of the manipulator and depicted the main purpose of a robot as the interaction with the environment. The grasp planning module developed there can be considered as the first step towards the autonomous interaction of hyper-redundant manipulators with their environments. With this work, the state of the art in neural pattern recognition will be investigated, along with theoretical information. The two main problem tried to be solved are the investigating the “tentacle case” and “linear object scanner case” problem.

In the work of [Ste97], the implemented vision based robot system is introduced to arrange objects in a 2D-scene. He also indicates that it is also important that recognition is invariant with scaling, rotation and position of the objects, and actually this implies the generalization ability. The necessary information is given manually to represent the shapes which are classified for tentacle case and $n \times n$ matrices for linear object scanner case.

Furthermore, I will investigate the categorization of objects and grasping process dealing with neural object recognition and fourier Series. A neural control for whole-arm grasping of objects with the body of a hyper-redundant manipulator has been subject of previous work [Bus02]. As an additional step towards, I provide an approach towards the result of recognition of objects with tentacle case and linear object scanner case which are classified in a desired manner.

1.2 Problem definition

The subject of this thesis is based on neural networks to implement the classification of objects in a desired manner. In fig 1.1, we describe our first problem as the recognition of objects with one-dimensional data arrays and the distance information which is necessary for the manipulator in order to entwine. To solve the tentacle case problem, first of all, we need categorization of objects and secondly with the help of fourier series the distance information for grasping of objects. The input vectors that are desired to be classified are so called one-dimensional data arrays. Using these kind of vectors, our system must be able to recognize the same object even it has different position information. We will investigate it with a special kind of neural network in chapter 3.

The benefits of turning functions and fourier descriptors methods are to scope with the problems of the recognition process. Firstly, the angles that are manually given as input are used for turning functions and with the help of fourier transformation the fourier coefficients are received. The coefficients of the object information through the fourier transformation are the key inputs for our neural networks. In chapter 4, we will find out an answer to this question. The other stage of method is to classify the objects for entwinement which will be used from the manipulator.

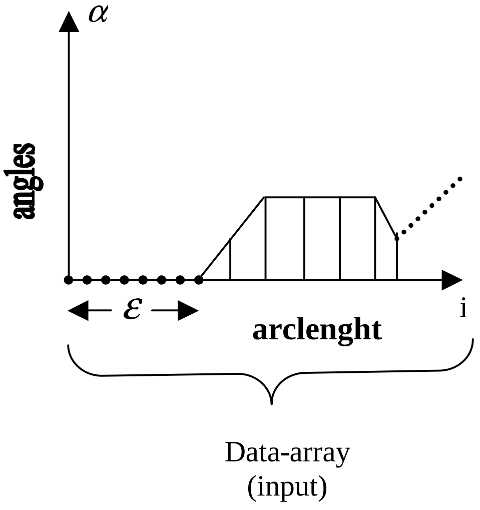


Figure 1.1: Tentacle case „offset-problem“. Turning function of an object. ε is the position information of the object according to the manipulator.

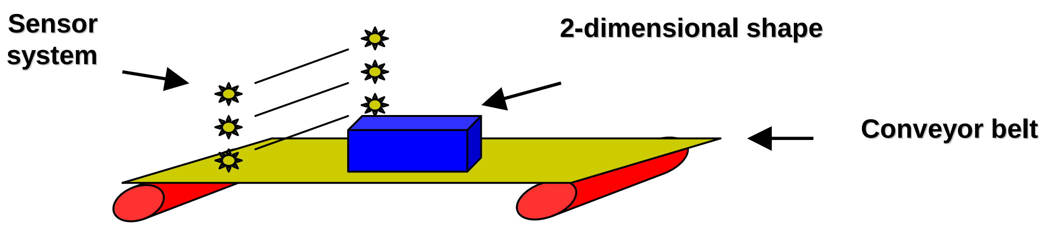


Figure 1.2: Linear object scanner case.

The second problem, showed in fig 1.1, is related with a conveyor belt that detects objects passing through the belt. The objects could be partially defected during the production process and must be noticed from the system with the help of sensors. To fill these requirements, I used an approach which the contours of the desired objects are taken in the training phase and suitable classification results are computed through classification process. The ability of extracting the contour and derived features recognize, localize and identify the objects automatically.

The aim of this work is to develop an approach towards an object recognition that is implemented using artificial neural networks. The simulation results of the problems will be discussed in chapter 5. We will mainly concern with our problems listed earlier, that is the problem of recognizing the objects and the grasping process by the manipulator.

1.3 Outline of thesis

In chapter 2, an overview of hyper-redundant manipulators is given; especially the kinematics and dynamic modelling of this type of robot will be discussed. A suitable artificial neural network algorithm is the topic of the following chapter 3; which deals with the artificial neural network algorithm; called Back-propagation algorithm after a brief overview of the theoretical background, a concept for neural object recognition is presented. After a presentation of a method for the recognition and classification of the objects, the radial-basis functions (RBF) is used and suitability is compared. Chapter 4, is dedicated to the neural object recognition. The shape-based recognition is based on two techniques namely the fourier descriptors and turning functions descriptors are introduced and concepts for a neural pattern recognition built from these are developed. The next chapter, chapter 5, gives an overview of simulation details and a discussion of the results of recognition and grasping process. Finally, implementation is discussed in chapter 6, followed by the future work.

2 Hyper-redundant manipulators

In this chapter, we will give a short overview about hyper redundant manipulators which is a work of [Bush02]. He presented a detailed information for whole-arm grasping of objects. The work of [Mar04] was to develop a neural sweeping pattern generator searching for objects. Furthermore, he provided an approach about recognition of objects used in sensory capability of manipulators. Even our work is more about pattern recognition for classification, it will be useful to have a basis knowledge about the manipulators.

In the following we will give a short view of our hyper redundant manipulator that has a 16 number of rigid links which are connected with revolute joints as a serial chain manipulator using the information from Planar Manipulators Toolbox. In praxis, Hyper redundant manipulators consist of a small number of links connected to serial chains by joints and has fixed at the ground. In our application, 16 joints are used, which provide sixteen degrees of freedom (DOF).

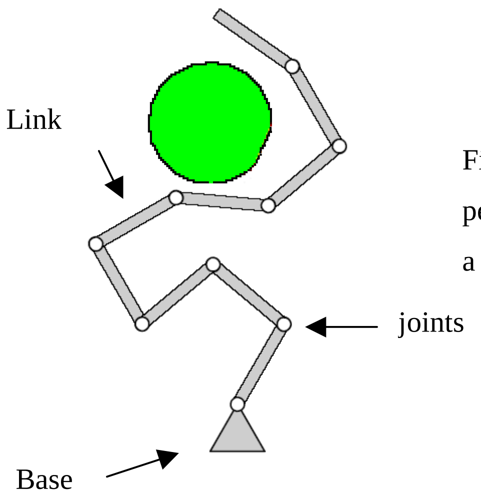


Figure. 2.1: 9-DOF manipulator performs with whole arm grasping of a planar object.

Hyper-redundant manipulator deals with the physical structure of a planar serial chain manipulator. It has *n* degree-of-freedom–each with one degree of freedom-planar manipulators with revolute joints. The manipulator is supposed to move in the vertical plane *x*-*y* as shown in Fig.2.2. The task coordinates ***x*** are the positions in *x*-*y* plane and in the planar case where $m = 2$ the vector ***x*** is ***x*** = $[x, y]^T$ [Leo00].

2.1 Joint positions defined as relative link angles

From this section on, we will deal with the mathematical derivation of manipulators that realize the grasping of objects. The first manipulator type is when joint coordinates are defined as relative angles between two links.

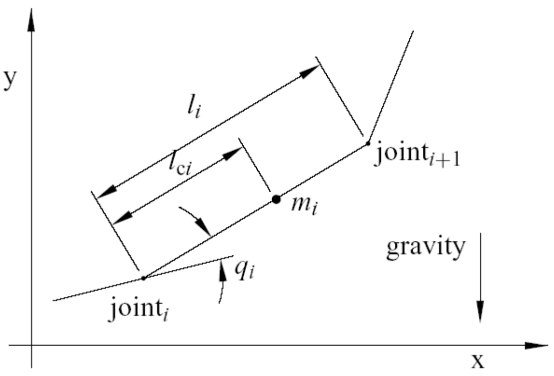


Figure 2.2: Structure of n-DOF planar manipulator [Leo00].

2.2 Kinematics modeling

2.2.1 Direct Kinematics

The direct kinematics of the manipulator provide a mapping between the joint variables and the end-effector position and orientation with respect to a reference frame.

With respect to *n* joint coordinates ***q*** and *m* task coordinates ***x*** the kinematics of the manipulator can be described with the following equations [Leo00]. The manipulator is called *redundant* if $n > m$.

$$x = p(q) \tag{2.1}$$

$$\dot{x} = \mathrm{J}(q)\dot{q} \tag{2.2}$$

$$\ddot{x} = \mathrm{J}(q)\ddot{q} + \dot{\mathrm{J}}\left(q, \dot{q}\right)\dot{q} \tag{2.3}$$

where p is a *m*-dimensional vector function representing direct kinematics, J is the Jacobian matrix and $\dot{\mathrm{J}}$ is its time derivative, $\dot{\mathrm{J}} = dJ/dt$. As we deal with redundant manipulators, *n* > *m* and J is mxn matrix. Let φ be a *n*-dimensional vector with components

$$\varphi_i = \sum_{j=i}^{i} q_j \tag{2.4}$$

for $i = 1,...,n$ and initial value $\varphi_0 = 0$ and l_i be the length of the i^{th} link. In the case of a planar manipulator with revolute joints the end effector positions $x, x = [x_1, y_1]^T$, can be expressed by the following equations.

$$x_i = x_{i+1} + l_i \cos(\varphi_i) \quad \text{and} \quad y_i = y_{i+1} + l_i \sin(\varphi_i) \tag{2.5}$$

for $i = n-1,...,1$ and initial values

$$x_n = l_n \cos(\varphi_n) \tag{2.6}$$

$$y_n = l_n \sin(\varphi_n) \tag{2.7}$$

The pairs $[x_i, y_i]^T$ represent the position of the end of the manipulator measured from the joint i. In the planar case the Jacobian J is a 2 x n matrix

$$J = \begin{bmatrix} \frac{\partial x_1}{\partial q_1} & \cdots & \frac{\partial x_1}{\partial q_n} \\ \frac{\partial y_1}{\partial q_1} & \cdots & \frac{\partial y_1}{\partial q_n} \end{bmatrix} \tag{2.8}$$

to derive $\dot{\mathrm{J}}$, we have to differentiate J with respect to time

$$\dot{\mathrm{J}} = \sum_{i=1}^{n} \left(\frac{\partial \mathrm{J}}{\partial \mathrm{q_i}} \dot{q}_i \right) \tag{2.9}$$

Since the hyper-redundant structure of the manipulator itself is used to handle objects, there is no use for an end-effector. The end-effector frame therefore only represents the endpoint and orientation of the last link.

2.3 Dynamic modeling for hyper-redundant manipulators

Before we determine the components of the dynamic model, we have to derive expressions for the position of the center-point of the mass and corresponding Jacobian matrices for all segments. Using equation (2.5) the position of the mass of the i^{th} link can be defined by

$$x_{ci} = \begin{pmatrix} x_1 - x_i + l_{ci}\cos(\varphi_i) \\ y_1 - y_i + l_{ci}\sin(\varphi_i) \end{pmatrix} \tag{2.10}$$

The Jacobian matrices related to the segments have been divided into two parts

$$\mathrm{J} = \begin{bmatrix} \mathrm{J}_{\mathrm{L}} \\ \mathrm{J}_{\mathrm{A}} \end{bmatrix} \tag{2.11}$$

where J_{L} and J_{A} are parts of J associated with linear and angular task velocities. Furthermore,

$$\frac{\partial x_{ci}}{\partial q_j} = \begin{pmatrix} -x_j + x_i - l_{ci}\sin(\varphi_i) \\ -y_j + y_i - l_{ci}\cos(\varphi_i) \end{pmatrix} \tag{2.12}$$

Next, the components of the vector of Coriolis and centrifugal forces h can be expressed by

$$h_i = \sum_{j=1}^{n}\sum_{i=1}^{n} h_{ijk}\,\dot{q}_j\,\dot{q}_k \tag{2.13}$$

where

$$h_{ijk} = \frac{\partial H_{ij}}{\partial q_k} - \frac{1}{2}\frac{\partial H_{jk}}{\partial q_i} \tag{2.14}$$

The vector of gravitational forces can be computed starting from the last link with

$$g_i = g_{i+1} + G\left(m_i l_{ci} + \sum_{k=i+1}^{n} m_k l_i \right)\cos(\varphi_i) \tag{2.15}$$

where G denotes acceleration of gravity.

3 State of the art of neural networks

3.1 Overview

In this chapter, two of the popular neural network architectures particularly Multi Layer Perceptrons (MLP) and Radial Basis Functions (RBF) are introduced with operational principles and specific affairs. Perceptron characterizes a single neuron which makes arbitrary decisions based on data from inputs and can determine input-output relation as learning patterns. In the following, we will give an overview and search suitable neural networks for the suitability. The advantages and disadvantages of the networks are given in a comparative form. The mathematical relations of the networks are quite complex but are also shortly given to understand the principles of the functions of the neural networks.

3.2 Neural Network for object classification

Neural network consists of a large number of simple processing units linked by weighted connections and is powerful because of the combination of many units in a network and therefore describes basically a nonlinear device and is itself nonlinear. The purpose of finding an answer of a special problem by varying the connection is to deal with geometric configurations and values of the connecting weights between units. Each unit in network receives inputs from many other units and generates a single output as we will see in mathematical derivation of bp in detail. The output acts as an input to other processing units and by this way the training process is executed. Once a neural network has trained, it is able to make predictions, for pattern recognition and categorization that is desired to be performed from the manipulator in this thesis.

The purpose of this section is to provide an approach with the help of neural networks, as early mentioned, for the classification problem. Being recognized of the shapes by our hyper redundant manipulator is important, in order to attack grasping process. For this reason neural networks play important role for the classification of the objects.

In the book of Simon Haykin [Sim88], it is apparent that a neural network derives its computing power through its massively parallel distributed structure and therefore produces reasonable outputs for inputs during training. The problem is to integrate the neural networks into a consistent system engineering approach because they cannot provide the solution working by themselves alone. There must be inputs which have enough information about a task or a situation which will be solved by the neural networks. Using neural networks give good results specifically, for a complex problem, which is separated into a number of relatively simple tasks, and neural networks are assigned a subset of the tasks such pattern recognition, which we deal with, associative memory and control.

In the work of our thesis, the neural network will be only used in order to make predictions which classify the object information of squares, rectangles, triangles and circles. As we discuss later in chapter 5, rbf is used and involves the modification of the synaptic weights of a neural network by applying a set of labeled training samples as a sequence of input arrays and gives sufficient results for our tentacle case problem that we will examine in detail. For one-dimensional data arrays, each example consists of a unique 16-data arrays called input vectors and by this way the object boundary information is introduced. Suzanna Becker [Bec91] showed the ability to form internal representations for encoding features of the input and thereby create new classes automatically. The reason why we make use of 16-data arrays is the number of links of our manipulator which we used for grasping process.

The network is presented by an example from the set, and the synaptic weights of the network are modified so as to minimize the difference between the desired response (d) and the actual response (y) of the network produced by the input signal in accordance with an suitable statistical criterion.

We will consider a pattern classification tasks where the requirement is to assign an input signal representing a physical object to one of several pre-specified classes. In our simulation, basically 4 types of objects namely square, rectangle, triangle and circle were used in order to achieve the classification procedure. The requirement is to estimate arbitrary decision boundaries in the input signal space for the pattern-classification task using these set of examples. It is clear that the more adaptive we make a system in a properly designed fashion, the more robust its performance will likely be when the system is required to operate, in a non-stationary environment. That means, it is also desirable that the network must work in noisy environment.

3.2.1 Learning Algorithm

If we deal with object classification, we can define neural networks as work process by giving inputs and as a training process by producing outputs which are used in categorization process. The procedure used to perform the learning process is called *learning algorithm*, the function of which is to modify the synaptic weights of the network to attain a desired design objective.

A neural network has three components, representation, learning and reasoning. We will shortly indicate the importance of components which is depicted in fig.3.1. In the representation process, general knowledge about a problem is represented by several symbol structures. In our work, the numbers were used as input vectors indicating the angles between the joints of manipulator. Explanations in the representation stage are important for recognition, giving decisions and asking for the classes of examples.

Reasoning is the ability to solve problems and give necessary results. In classification process of objects, reasoning can be seen as deciding the class membership of patterns and learning is a process of adapting the weights of neural network by the environment in which the network is located.

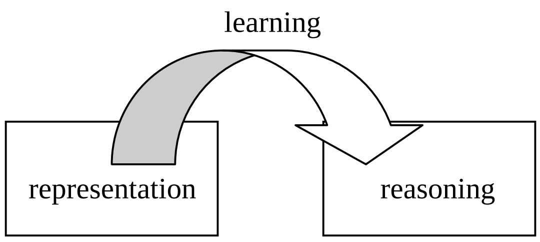


Figure 3.1: Illustrating the three key components of an Al system.

The recognition process is related with these three components of artificial intelligience system and actually based on the learning structure of human beings. The important advantage of this system is to computation and decision time.

Main interest in this thesis is confined largely to an important class of neural networks that perform useful computations through a process of learning and making decision rules for classification of the objects by using pnn and then achieving the grasping process by the manipulator. The learning process in human beings is based on also training sets. The key of learning is to forget the old knowledge and thus new relations between the neurons are created to save and to learn new information. Therefore we can say that learning is a process of forgetting the unusual knowledge. Each second, neurons die and the axons, which carry the information, disappear. By creating new contacts between neurons, the learning process occurs and the knowledge is stored again. The number of neurons in the brain plays significant role in learning process and the more related links between the neurons validate the certain results.

The inputs are given where the information is known and would like to be derived unknown information and it is known that there is a relationship between the inputs and outputs. In chapter 5, four types of inputs for each class is used to train the neural network and at the end of the program the test set is used to give the desired results for the classification. These inputs are the sets of training data for our neural network. These training data contain examples of inputs as a regular sequence of vectors and the network learns to classify the desired object boundary information using the vectors as a statistical manner.

The important point in classification is the useful interpretation to treat the network outputs as probabilities which we used the neural network in this thesis called pnn. With the other words, the network learns the probability density function of the classes. Bishop [Bis95] showed out that it is only valid under certain cases about the distribution of the data. For tentacle case problem, the results are suitable for whole-armed grasped objects but are not sufficient for the other cases namely not fully grasped and more than one-time grasped objects. The classification results will be studied respectively in chapter 5.

3.3 Back-Propagation

3.3.1 Multilayer Perceptrons (MLP)

This section describes an art of neural networks which is applied to solve some difficult problems by training them in a supervised manner. Typically, the mlps consist of a set of sensory units that make up the input layer, one or more hidden layers of computation nodes and an output layer of computation nodes as depicted in fig 3.2.

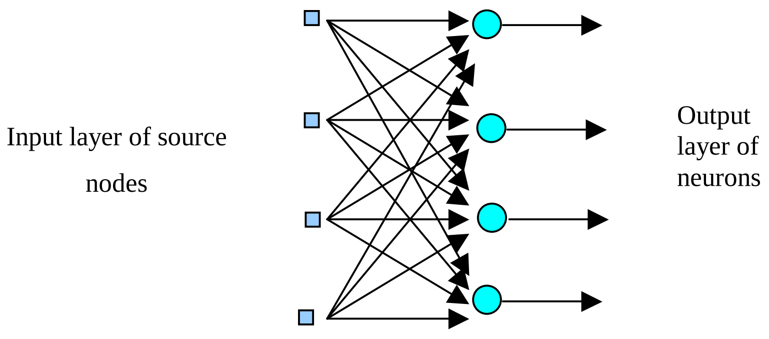


Figure 3.2: Feed-forward network with a single layer of neurons.

The input signal propagates through the network in a forward direction, on a layer-by-layer basis. These neural networks are commonly referred to as multilayer perceptrons, which represent a generalization of the single-layer perceptron [Sim88].

A multilayer perceptron has three distinctive characteristics:

1. The model of each neuron in the network includes a non-linearity at the output end. A commonly used form of non-linearity that satisfies this requirement is a sigmoidal non-linearity defined by the logistic function:

$$y_j = \frac{1}{1+\exp(-v_j)}$$

where v_j is the net internal activity level of neuron *j*, and y_j is the output of the neuron. The presence of non-linearity is important because, otherwise, the input-output relation of the network could be reduced to that of a single-layer perceptron.

2. The network contains one or more layers of hidden neurons that are not part of the input or output of the network. These hidden neurons enable the network to learn complex tasks by extracting progressively more meaningful features from the input vectors where we will see the same effect in radial basis function networks with one hidden layer.

3. The network shows a high degree of connectivity, resolved by the synapses of the network. A change in the connectivity of the network needs a change in the population of synaptic connections or their weights.

3.3.2 Mathematical explanation of the Back-Propagation Algorithm

In the following, we will give an overview of Back-Propagation (BP) learning algorithm and general learning algorithms used in neural networks. Typically these mathematical explanations help us to understand the process of learning paradigm which is inspired from basics of the neurons. The structure of an artificial intelligence machine is to achieve the learning algorithm and produces input-output mapping.

In the book of Simon Haykin has given the mathematical explanations of mlp. The error signal at the output of neuron *j* at iteration *n*, presentation of the n_{th} training pattern, is defined by

$$e_j(n) = d_j(n) - y_j(n), \quad \text{neuron } j \text{ is an output node} \tag{3.1}$$

Fig. 3.3 depicts neuron *j* being fed by a set of function signals produced by a layer of neurons to its left. The net internal activity level $v_j(n)$ produced at the input of the neuron j is given as;

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n) \tag{3.2}$$

where *p* is the total number of inputs applied to neuron *j*.

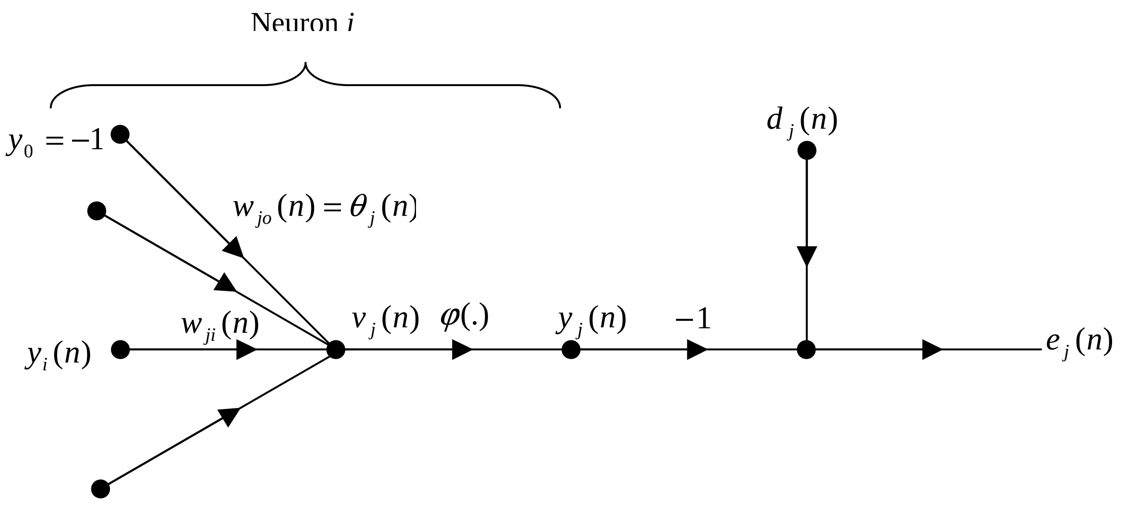


Figure 3.3: Signal-flow graph highlighting the details of output neuron j.

The synaptic weight w_{jo} equals the threshold Q_j, applied to neuron *j*. Hence the function signal $y_j(n)$ appearing at the output of neuron *j* at iteration *n* is

$$y_j(n) = \varphi(v_j(n)) \tag{3.3}$$

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\, e_j(n)\ \varphi_j^{'}(v_j(n))\ y_i(n) \tag{3.4}$$

$$\Delta w_{ji}(n) = \eta\ \delta_j(n) y_i(n) \tag{3.5}$$

According to Eq.(3.5) the learning parameter η affects the change of synaptic weights. The learning parameter must be good chosen in order to have good results. If the learning parameter is small chosen, then the synaptic weights of neural network changes will be small. On the other hand, the network will be unstable, namely oscillatory. The *local gradient* $\delta_j(n)$ is itself defined by

$$\delta_j(n) = -\frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \tag{3.6}$$

$$= e_j(n)\ \varphi_j^{'}(v_j(n))$$

The local gradient points to required changes in synaptic weights. According to Eq. (3.6), the local gradient $\delta_j(n)$ for output neuron *j* is equal to the product of the corresponding error signal $e_j(n)$ and the derivative $\varphi_j^{'}(v_j(n))$ of the associated activation function. If the changes in the synaptic weights are small, the learning process is said to be complete.

3.4 Pattern Recognition Task

The main point in recognizing process is that the pattern classification task has important place in vision-based systems. If a standard rbf network is used to perform a complex pattern classification task, the problem is basically solved by transforming it into a high dimensional space in a nonlinear manner provided by Cover's theorem on the separability of patterns [Cov65]. From the work we did in this thesis, we know that once we have linearly separable patterns by the tentacle case problem, then the classification problem is easy to solve. But for the linear object scanner case object recognition, it is hard to say that classification process is achieved properly. Accordingly, we may develop a great deal of insight into the operation of a rbf network as pattern classifier depicted in this section.

For all our input data arrays there is a pre-defined set of classes of patterns which might be presented, and the aim of the object recognition task is to classify a given pattern as one of these classes. The patterns that we used in this thesis are in the form of numbers and are called as features, which are measurements used as inputs to the classification system as shown below.

square = [0 0 0 pi/2 0 0 0 pi/2 0 0 0 pi/2]

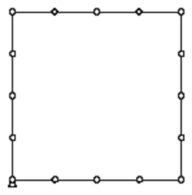


Figure 3.4: One-dimensional data array and its representation as a shape.

The number of one-dimensional data arrays is equal to the number of links of our manipulator for the whole-arm grasping. In the classification process, we have a family of surfaces, each of which naturally divides an input space into four regions namely for squares, circles, triangles and rectangles. With the help of Cover's theorem, say X denotes set of N patterns x_1, x_2, x_3, x_4 each of which is assigned to one of four classes. For each pattern $x \in X$, define a vector made up of a set of real-valued functions $\{\varphi_i(x) \mid i = 1,2,3,4\}$ as shown by

$$\varphi(x) = [\varphi_1(x), \varphi_2(x), \varphi_3(x), \varphi_4(x)]^T \tag{3.7}$$

Suppose that the pattern *x* is a vector in a *p-dimensional* input space. The vector $\varphi\ (x)$ maps points in *p-dimensional* input space into corresponding points in a new space of 4-dimension. We refer to $\varphi_1(x)$ as a hidden function, because it plays a similar role that of a hidden unit in a feed-forward neural network described previously.

To illustrate the significance of the idea of φ-separability of patterns, we consider that there are four kind of patterns, namely, square, triangle, circle and rectangle. Basically, a nonlinear mapping is used to transform a nonlinearly separable classification problem into a linearly separable one. We define Gaussian hidden functions as follows for our shapes $t_1 =$ square, $t_2 =$ circle, $t_3 =$ triangle and $t_4 =$ rectangle.

$$\varphi_1(x) = e^{-\|x-t_1\|^2}, \qquad t_1 = [0\,0\,0\ \mathrm{pi/2}\ 0\,0\,0\ \mathrm{pi/2}\ 0\,0\,0\ \mathrm{pi/2}\ 0\,0\,0\ \mathrm{pi/2}]^T$$

$$\varphi_2(x) = e^{-\|x-t_2\|^2}, \quad t_2 = [\mathrm{pi/8\ pi/8\ pi/8\ pi/8\ pi/8\ pi/8\ pi/8\ pi/8\ pi/8\ pi/8\ pi/8\ pi/8\ pi/8\ pi/8\ pi/8\ pi/8}]^T$$

$$\varphi_3(x) = e^{-\|x-t_3\|^2}, \qquad t_3 = [0\,0\,0\,0\,0\ \mathrm{pi/3}\ 0\,0\,0\,0\ \mathrm{pi/3}\ 0\,0\,0\,0\ \mathrm{pi/3}]^T$$

$$\varphi_4(x) = e^{-\|x-t_4\|^2}, \qquad t_4 = [0\,0\,0\,0\,0\ \mathrm{pi/2}\ 0\ \mathrm{pi/2}\ 0\,0\,0\,0\,0\ \mathrm{pi/2}\ 0\ \mathrm{pi/2}]^T$$

Accordingly, the input patterns are mapped into the $\varphi_1 - \varphi_2 - \varphi_3 - \varphi_4$ plane. φ_1 plane is depicted for squares, φ_2 for circles, φ_3 for triangles and φ_4 for rectangles. For our work, there is no need to increase the dimensionality of hidden units which causes complexity in designing and the training time. In other words, nonlinearity exemplified by the use of Gaussian hidden functions is sufficient to solve classification problem for tentacle case problem. The results of the tentacle case problem simulation have shown that, the object recognition process could not be achieved with rbf and the results will be discussed later.

For the present thesis, derived from the bp learning algorithm, we can say that, the activation of the rbf unit depends on the weights multiplied with Gaussian function.

$$v_j(n) = \sum_{i=1}^{M} w_{ji}(n).\varphi_i(n) + b_j \tag{3.8}$$

where $v_j(n)$ is the activation function of neuron, $w_{ji}(n)$ is the weight parameters, b_j the bias term of *j*. The rbf technique consists of choosing a function F that has the following form

$$F(x) = \sum_{i=1}^{N} w_i \varphi(\|x - x_i\|) \tag{3.9}$$

where $\{\varphi(\|x - x_i\|) i = 1,2,...,N|\}$ is a set of N nonlinear functions, known as radial-basis functions, w_i is the weight of vectors and $\|.\|$ denotes a norm that is usually taken to be Euclidean (distance measure).

$$\varphi(\|x - x_i\|) = \exp\left(- \frac{\|x - x_i\|^2}{2\sigma_i^2} \right) \tag{3.10}$$

where x_i is the center, σ_i indicates the width of clusters. By determining of these parameters, our problem becomes linear. In the paper of [Mic04], the possibilities of rbf has been shown as several outputs. Euclidean distance is the square root of the sum of squared differences for each of the variables describing the four objects whose similarity or dissimilarity is wished to express the quantity [Cast04].

In this thesis, we have four outputs to realize the categorization problem to classify the objects. It is clear that the more examples and classes we have, the more efficient results we carry out but the important point in neural object classification is to determine the shapes with small number of class membership. The work of neural network architecture is to find the optimum way for recognizing the objects which are desired to be classified.

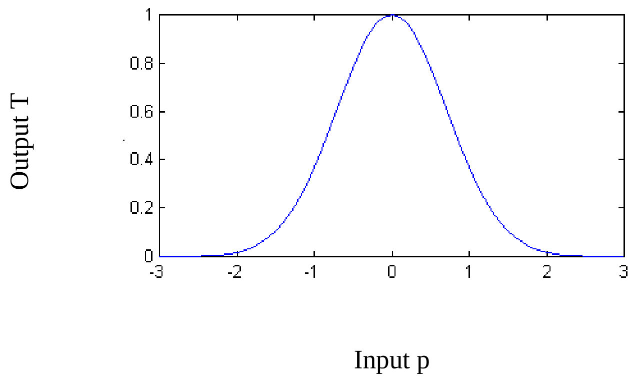


Figure 3.5: Radial basis transfer function.

In the following we will give shortly the basis function of radial basis neural networks using Matlab help. The radial basis function has a maximum of 1 when its input is 0 as depicted in fig 3.6. As the distance between *x* and *t* decreases, the output increases. Thus, a radial basis neuron acts as a detector that produces 1 whenever the input *t* is identical to its weight vector *t*. The bias *b* allows the sensitivity of the radbas neuron to be adjusted. For example, if a neuron had *a* bias of 0.1 it would output 0.5 for any input vector *t* at vector distance of 8.326 (0.8326/b) from its weight vector *x*.

For the classification problem of tentacle case it can be said that it is adequate, but for the linear object scanner case, it could not be possible to receive the right classification results. It will be better if it could be achieved the recognition of objects that are drawn by a user with a combination of other neural network methods as a future work for the manipulators because it will make the use of robots flexible in the areas where the implementations are not possible by human-beings.

In [Smc00], the advantages of rbfns are described as in implementation and with respect to optimization of the training data. In his paper, he mentioned the implementation of the network that it is mathematically simple while it uses only basic linear algebra and iteration is not required for computing on the input data.

3.4.1 Radial Basis Function Networks

In this section, we introduce notions related to feed-forward networks, the second approach as a neural network and the main method that we used in this thesis is so called rbfns, which has two layer feed-forward networks. The aim of this work can be described as achieving the categorization problem using rbfn.

Broomhead and Lowe were the first to make use of radial-basis functions in the design of neural networks [Sim88]. As follow, we will give an overview of rbf and then the pnn, which is a kind of rbf, will be worked out dealing with implementation in our simulation. The construction of a rbf network in its most basic form involves totally different layers, the input layer, the hidden layer with the rbf non-linearity and a linear output layer as depicted in fig 3.4. The input layer is made up of source nodes. The second layer is a hidden layer of high enough dimension, which serves a different purpose from that in a mlp. When the input vectors are expanded into the hidden-unit space, set of functions called radial basis functions are provided. The output layer supplies the response of the network to the activation patterns applied to the input layer.

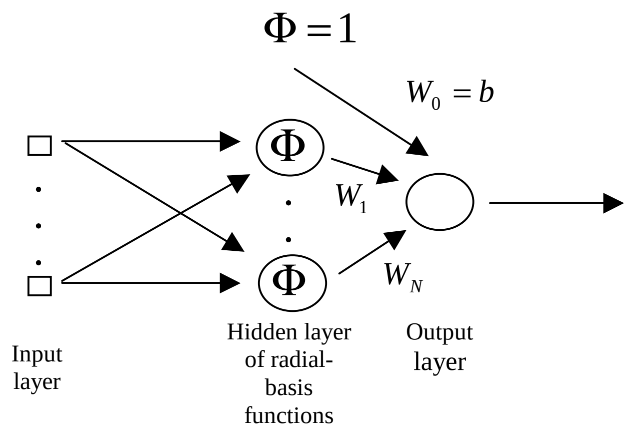


Figure 3.6: Radial-basis function network structure. The transformation from the input space to the hidden-unit space is non-linear, while on the contrary the transformation from the hidden-unit space to the output space is linear.

where ϕ are the radial basis transfer functions, ω are the synaptic weights.

3.4.2 Probabilistic Neural Network (PNN)

I briefly mentioned that, in the context of classification problem, a useful interpretation of network outputs was as estimates of probability of class membership, in which case the network was actually learning to estimate a probability density function. The useful method, which was implemented in this thesis, gives good results for pattern recognition of tentacle case problem.

Pnn, which is a Bayesian classifier is used in our thesis and it provides a general solution to pattern classification problem by following an approach developed in statistics, called Bayesian classifiers. Because of the nature of Bayesian classification, the pnn does not require iterative learning therefore faster than bp networks. The pnn does require calculation of a smoothing factor, which represents the width of the calculated Gaussian curve for each probability density function. The pnn has exactly one internal layer of neurons, with one neuron for each training pattern (circles, triangles, rectangles and squares). The network output corresponds to the estimate of the probability density function for each possible outcome.

Bayesian decision theory [Bay] shows that there is an important relationship between neural networks and pattern classification for the object recognition. This theory takes into account the relative likelihood of events and uses a priori information to improve prediction. Estimating probability density functions from data has a long statistical history. More generally, Bayesian statistics can estimate the probability density of model parameters given the available data. To minimize error, the model is then selected whose parameters maximize this probability density function.

Decide „circle“ if *P(Circle)>P(triangle)*, otherwise, decide “triangle“

Speaking in the context of the above example, we tried to explain the idea of the deciding classification from this model with the object information that was used. The probability distributions are important to achieve the decision rule. The priori information is the probabilities of either a circle or a triangle that is given to network. If a decision must be made with little information, the rule as shown above is used. *P(Circle)* is the probability of the object being a circle. The way that is used by the neural network will classify the objects according to the maximum probability. This example is simple to understand the idea of a pattern recognition problem in Bayesian Theory. The task is to learn the probabilities from the training set. Bayesian decision theory is a classification problem as an example of a decision problem given observed features of an object which finds its class [L02].

In the classification problem, we will give an overview of the structure and discuss the advantages and disadvantages of pnn.

net = newpnn (P,T)

This command creates a network and takes two or three arguments. *P* is the input vectors and *T* is the target class vectors. The detail information about input and target class vectors will be given in the simulation chapter. From this point on, the network uses the vectors and outputs values for the classification and thus outputs are tested.

It is called a “neural network" because of its natural mapping onto a two-layer feedforward network [wpnn]. In the pnn, there are input, hidden and output layers as briefly mentioned in rbfn. The hidden units are copied directly from the source nodes and each model a Gaussian function placed in the middle at the training case. There is one output unit per class. In our thesis, there are four outputs for each kind of shape and each shape is connected to all the radial units belonging to its class. Hence, the output units simply add up the responses of the units belonging to their own class. The outputs are each proportional to the kernel-based estimates of the probability density functions of the various classes, and by normalizing these to sum to 1.0 estimates of class probability are produced.

The greatest advantages of pnns are the fact that the output is probabilistic and allows it to make an interpretation of output and the training speed. Training a pnn actually consists mostly of copying training cases into the network, and so is as close to instantaneous as can be expected. Training of the pnn is much simpler than with back-propagation because no training is involved prior to classification. Their design is straightforward and does not depend on training. A pnn is guaranteed to converge to a Bayesian classifier providing it is given enough training data [MATLAB HELP]. For our tentacle case object recognition problem, pnn performed quite sufficient results.

The necessity for storing the entire training set in memory which leads to higher computational make use of the pnns hard because of the necessity of more neurons compared to back propagation, but with several fast memories has eliminated this problem.

We will now go into the description of pnn from the mathematical side using the work of Maragoudakis [Mar]. If a pnn for classification in *K* classes is considered, the probability density function $f_i(x_p)$ of each class *i* is defined by equation (3.11)

$$f_i(x_p) = \frac{1}{(2\pi)^{d/2}\sigma^d}\frac{1}{M_i}\sum_{j=1}^{M_i}\exp(-\frac{1}{2\sigma^2}(x_p - x_{ij})^T(x_p - x_{ij})) \tag{3.11}$$

where x_{ij} is the *j-th* training vector from class i, x_p is the *p-th* input vector, *d* is the dimension of the feature vectors, M_i is the number of training patterns in class i and σ acts is a smoothing factor to soften the surface defined by the multiple Gaussian factor. Each training vector x_{ij} is assumed to be a centre of a kernel function, and as a result the number of pattern units in the first hidden layer of the neural network is given as a sum of the pattern units for all the classes. The variance acts as a smoothing factor, which softens the surface defined by the multiple Gaussian functions. As seen in equation (3.11), has the same value for all the pattern units.

The pnn classifier decides to which class the test vector belongs, depending on the degree of similarity of the input feature vector to the model of each class.

As depicted above, only two of our shapes are given. To achieve more sufficient results of classification, the learning rate parameter must be good chosen and the training time might be big enough and it will give better results.

If we deal with tentacle case problem, it can be easily predicted that the sufficient results could be taken only by using pnn for classification.

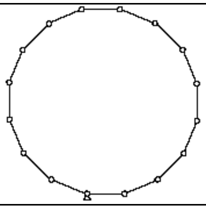
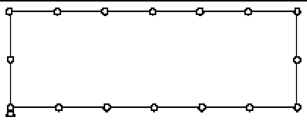
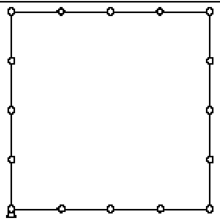
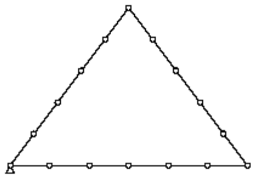
As we mentioned in section pattern classification task we know that once we have linearly separable patterns by the tentacle case problem, then the classification problem is easy to be solved. It also gives adequate consequences to our questions of positioning of the objects. This problem will be discussed in the next chapter.		
	circle = [pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8]	Each input data shows linearly separable patterns and therefore the classification of the probability density functions maps properly input-output mapping.
	rectangle = [0 0 0 0 0 pi/2 0 pi/2 0 0 0 0 0 pi/2 0 pi/2]	
	square = [0 0 0 pi/2 0 0 0 pi/2 0 0 0 pi/2 0 0 0 pi/2]	
	triangle = [0 0 0 0 0 pi/3 0 0 0 0 pi/3 0 0 0 0 pi/3]	

Figure 3.7: The classification results of one-dimensional data arrays.

The greatest disadvantage is the network size of neural network. Pnn network actually contains the entire set of training cases, and is therefore slow to execute and are used for classification problems. Training of the probabilistic neural network is much simple when it is compared with mlps. The detail consequences of the classification can be seen in chapter 5 in the context of simulation results.

3.4.3 BP&RBF

As it will be described in this section, rbfns and mlps are compared which are examples of non-linear feed-forward networks previously mentioned. However, these two networks differ from each other in several important respects, as outlined here. By studying the differences between mlp and rbf, we can generate insights into the classification process. Simon Haykin described the difference of the networks from five points of view.

1. An rbf network has a single hidden layer, whereas an mlp may have one or two hidden layers.
2. Typically the computation nodes of an mlp is located in a hidden or output layer, share a common neuron model. On the other hand, the computation nodes in the hidden layer of an rbf network are quite different and serve a different purpose from those in the output layer of the network.
3. The hidden layer of rbf network is nonlinear, whereas the output layer is linear as depicted in fig 3.4. The hidden and the output layers of mlp used as a classifier are usually all nonlinear.
4. The argument of the activation function of each hidden unit in a rbf network computes the Euclidean norm between the input vector and the center of that unit. The activation function of each hidden unit in mlp computes the inner product of the input vector and the synaptic weight vector of that unit.
5. Mlps construct global approximations to nonlinear input-output mapping. Consequently, they are capable of generalization in regions of the input space where little or no training data are available. Rbf networks using exponentially decaying localized nonlinearities namely Gaussian function construct local approximations to nonlinear input-output mapping, with the result that these networks are capable of fast learning and reduced sensitivity to the order of presentation of training data.

The activation of units in the hidden layer of an rbf network depends directly on the input pattern. The output of a hidden unit depends on the distance of the input vector to the center of the units basis function in Gaussian function. This results in nearby zero activations for distant units and high activation for units close to the input; the representation of a target function with rbf networks is local.

Due to a poor ability of generalization of new input patterns from past data, the quantity of training data needed to specify the mapping grows exponentially in rbf networks and this slows down the speed of network. Therefore the number of training data must be optimal chosen to avoid for a long time of waiting for the classification. In the comment section of Michel Verleysen’s work, it is also indicated of easy learning of rbf compared to mlp.

3.5 Conclusion

In this chapter, we tried to compare the two algorithms of mlp and rbf. The state of the art in neural pattern recognition of one-dimensional data was worked out and in the context of the classification, the results of the objects were shown. We have chosen the variables from numerical variables that were used for the training. A lot of cases are required for the correct results; the more variables, the more cases. For the good classification results we need more training data but in this thesis only four kinds of objects and for each object one kind of training sets are used in order to make prediction of classification using small number of training set.

The reason of choosing rbf networks in this work is the number of advantages over mlp. First, as previously stated, they can model any nonlinear function using a single hidden layer, which removes some design-decisions about numbers of layers. Second, the simple linear transformation in the output layer can be optimized fully using traditional linear techniques, which is fast, rbf networks can therefore be trained extremely quickly. Third, implementation of the network is mathematically simple because it uses only basic linear algebra. Computations on the input data do not require iteration and are therefore relatively cheap. These reasons are the benefits of choosing pnn in classification problem. Typically, for our two problems, only by the tentacle case problem we could take the sufficient results. For the linear object scanner case, the classification of defected objects could not be achieved. Even we change the size of the matrix, it also possible to recognize different types of objects.

In this chapter, we developed a theoretical framework based on neural networks that is concerned with learning from examples. The most general form of neural network is called Probabilistic neural networks, since it is related to the well-known Radial Basis Functions, mainly used for classification. The results of classification with pnn for our two problems will be given in Simulation chapter. Castleman [Cast04] implies that after the training procedure, the ann will be capable of estimating previously unknown output values, given a set of input values. The learning of a bp is done by an example, processing a training file that contains a series of input vectors and the target output vector for each.

In the next chapter, the shape-based recognition techniques will be given and two of these important methods will be the main information sources for the nerual network in order to make the classification and grasping process. The given objects for not grasped, one-time grasped and more than one-time grasped are classified with the help of pnn

4 Neural object recognition

4.1 Overview

In the following, we will discuss the object recognition problem with the help of shape-based recognition methods using fourier transformation and turning functions. The shape of each object is described using several number of descriptor values, typically in our work 16 real numbers. Furthermore, the ability of entwinement of the objects of the manipulator is examined in grasping process.

The purpose of shape-based recognition is to give the necessary information to the neural network for the classification and entwinement process and is related with matching the descriptors of each shape to standard values representing our shapes and choosing the closest match. A variety of different algorithms have been developed to perform two-dimensional object recognition with one-dimensional data arrays, utilizing many different types of features and matching methods. For the present thesis, it has not been practical to consider the other matching methods in detail but it is hoped that the selection which follows in this section carries the important principles used and that any other algorithms are simply variations on a theme. The objects used in this work are two-dimensional which are assumed to be flat so that the whole closed contours of the desired objects can be extracted. We will see the details of the simulation of the objects which shall be grasped by the manipulator in the next chapter.

4.2 Pattern recognition

In this section, we will describe the recognition process of the objects and give an idea how the feature of the objects are extracted and used in the matching procedure for the classification process.

In neural object recognition, data representation and decision making plays important role. These terms will be examined detailed in the further section. The system collects the information of the shapes, which are desired to be classified, and through the techniques that we previously mentioned, gives the sufficient results for the classification. The numeric information of the objects and a classification process that deals with recognition relies on the extracted features that are used as manually in this work. In the thesis of Brazda [Mar04], the shape boundary points are determined from the sensor data and the grasping process is achieved through the distance sensors that detect the minimum distance to the boundary. In our work of thesis, the shape information is given manually and the features are directly extracted form the set of patterns. This set of patterns is called the training set and the resulting learning strategy is characterised as supervised.

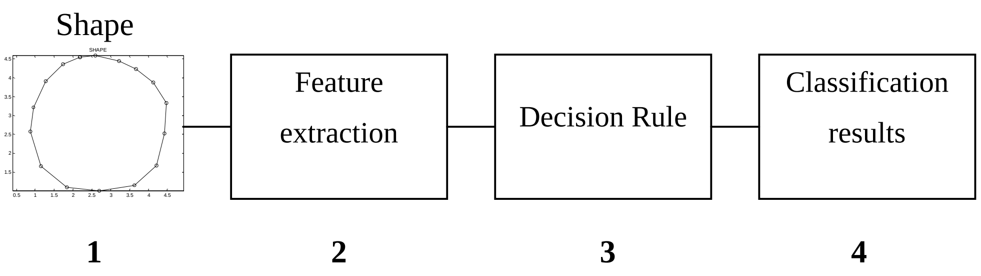
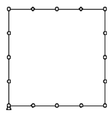


Figure 4.1: The feature extraction and classification problem.

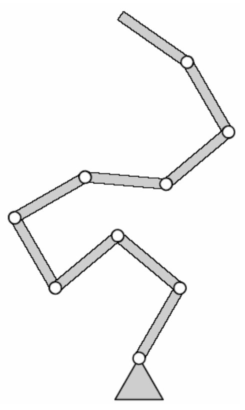
A shape matching for two-dimensional planar objects is a central problem in visual information systems, computer vision, pattern recognition, and robotics [Velt99]. It will be useful if we divide the neural object recognition problem into several intermediate stages, the process of starting with a shape and ending with a decision of class-membership of that shape. Each sub-problem is very important in this recognition process. It must be remembered that in the end the optimality of the entire system is only as optimal as the weakest link in the chain. The pattern recognition problem is separated into a series of sub-problems such as feature extraction, decision rule (see Fig.4.1). The purpose of a pattern recognition in this thesis is to analyze the description of the shape. The first stage consists of feature extraction or measuring the "shape" of the objects. The second stage is concerned with classifying the object into four categories of the shapes. Decision rule and grasping results are the topics of chapter 5 that will be examined how these problems could be solved and the difficulties will be discussed.

In the following, we will introduce the term feature by assuming that *d features* are observed on a pattern or object related with the number of links of the manipulator, then we can represent the pattern by a 1-dimensional vector $X = (x_1, x_2, ..., x_n)$ and usually refer to *X* as a *feature vector* and the space in which *X* lies as the *feature space*. Patterns are thus transformed by the feature extraction process into points in 1-dimensional feature space. In fig 4.2 (a), we see the 16-data arrays as an input to the network. A pattern class can then be represented by a region or sub-space of the feature space. Classification then becomes a matter of determining in what region of the feature space an unknown pattern falls into. For the classification process of our work, the manipulator is characterized by sixteen rotation joint so that the end detector can be arbitrarily positioned and orientated in the working space. The manipulator can be seen as a shape definer (see. Fig 4.2 (b)) for our shapes and therefore the feature vector consists of angles $q_1, q_2, ..., q_{16}$ in a two-dimensional planar. Fig 4.2 (a) implies the shape square with 16-data arrays that show the angles between two points.

square = [0 0 0 pi/2 0 0 0 pi/2 0 0 0 pi/2 0 0 0 pi/2]



(a)



(b)

Figure 4.2: (a) The feature explanation and illustration of a shape. (b) The manipulator as a shape definer.

The recognition of objects plays an important role in shape analysis for robotics. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space [Burks].

The data used for the simulation in chapter 5 was taken from the one-dimensional data arrays as mentioned in fig 4.2.(a). Most applications using fourier descriptors, moment invariants and scalar descriptors for shape recognition deal with the classification of the shapes but for our work we used only fourier descriptors method for definite shapes, namely squares, circles, triangles and rectangles.

Pattern recognition is concerned with making decisions of complex patterns. In [Cast04], it is given an introduction for pattern recognition and underlined the classification as an important part of automatic scene understanding.
It is assumed:

- that an image may contain one or more objects of interest
- that each object belongs to one of several predetermined *types* or *classes*

Actually, our image can be viewed as one object of interest and therefore easy to classify. Basically, the input data set used in this thesis for the classification and pattern recognition performs the conditions of this paper. Our feature vector for tentacle case is one-dimensional vector and for linear object scanner case is typically $5x5$ matrices and each data sets are introduced for a class.

In this chapter, we will deal with the fourier descriptors and turning functions which describe shapes irrespective of position, scale and orientation for the classification and grasping process. For information of the other techniques, several papers are published and discussed.

Before we go deep into the main idea, the term “shape” must be explained in order to understand the classification of objects from the given boundary points. The definition of shape can be done if we start with some properties that we agree on [Died03].

- A shape describes a spatial region and occupies an area in the space.
- A shape consists of points and typically addresses 2D space.

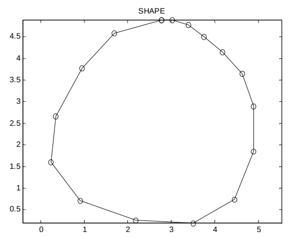


Figure 4.3: Two-dimensional shape.

The shape examples reveal with the conditions of a shape (see Fig. 4.3). We consider shape as something geometrical and will use the term shape for a geometrical pattern, consisting of a set of points, curves and surfaces. The points on the shape boundary can be interpreted as the links of the manipulator in order to determine the feature vector and turning function.

4.2.1 Process analysis and classification

There are two important points of views which will be discussed in the next section. In the following, we will shortly describe the data representation and decision making process and give an overview as follows:

1. Data representation
   - Feature extraction
   - Descriptor
2. Decision Making
   - Matching process

The two important methods in pattern recognition can be described as above. Data representation deals with shape analysis and for the shape analysis development boundary scalar transform techniques, turning functions and fourier transform of boundary, will be examined. Feature extraction and descriptors will be discussed in the next chapter.

Decision making is related with neural networks that classify the patterns comparing with the given inputs. As a result, decision making process can be seen as development and classification.

4.2.2 Feature Extraction

Feature extraction is the name given to a family of procedures for measuring the important shape information contained in a pattern so that the task of classifying the pattern is made easily by a formal procedure [God05]. The term “feature” is described as a simple geometric characteristic of the object and takes place in data representation. In [09], the feature extraction is explained as classification which is carried out through the comparison of objects of interest with reference objects.

For the non-complexity of the feature extraction, we used the features of the shapes by giving them manually and pre-defined classes of the objects were presented. The features or variables, which were measures of quantities considered to be relevant to characterising the objects of interest, were analysed for purposes of the comparison.

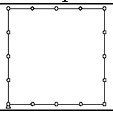
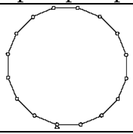
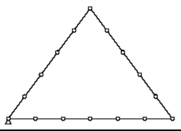
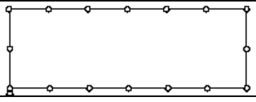
square = [0 0 0 pi/2 0 0 0 pi/2 0 0 0 pi/2 0 0 0 pi/2]

circle = [pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8]

triangle = [0 0 0 0 0 pi/3 0 0 0 0 pi/3 0 0 0 0 pi/3]

rectangle = [0 0 0 0 0 pi/2 0 pi/2 0 0 0 0 0 pi/2 0 pi/2]


Figure 4.4: The feature representation of the shapes that are used for the classification.

The idea is based on the angles between separately distributed points and 16-data points give a two-dimensional planar object. The objects are said to be closed shapes because the last vector meets with the first vector. By this way, the shapes describe a spatial region and occupy an area as 2D space (see fig.4.4). The sum of angles between the points gives 360°, which proves that the first vector and the last vector meets and identify a closed-shape.

4.3 Fourier Analysis

4.3.1 Introduction

As we early mentioned, the most important point in object recognition is that the system has to cope with the arbitrary position, orientation and scaling of the objects. To full these requirements, a model based approach namely fourier descriptors and turning functions were used. The fourier descriptors method is formed by applying fourier transform to the coefficients of wavelet transform of the object boundary. After transforming the turning function to a periodic signal, it will be possible to realize the grasping process with the help of fourier transformation.

4.3.2 Fourier Descriptors

In the scope of this thesis, the fourier transform technique is used for shape description in the form of fourier descriptors and for recognition technique. The shape descriptors generated from the fourier coefficients numerically describe shapes and are normalised to make them independent of translation, scale and rotation. Once a function is obtained, a fourier transform can be used to convert the function from space domain to frequency domain. As depicted in fig 4.5, the coefficients describe a given one-dimensional function. These fourier descriptor values produced by the fourier transformation of a given image represent the shape of the object in the frequency domain. With fourier descriptors, global shape features are captured by the first few low frequency terms, while higher frequency terms capture finer features of the shape. The lower frequency descriptors contain information about the general features of the shape, and the higher frequency descriptors contain information about finer and the small details of the shape. Therefore, the lower frequency components of the fourier descriptors define a rough shape of the original object.

Each fourier coefficient is calculated from every boundary points and therefore sensitive to all the points of shape. Before applying fourier transform on the shape signature, shape is first sampled to fixed number of points and the shape boundary or the shape signature of objects and models must be sampled to have the same number of data points. The larger the number, the more details the shape is represented, consequently, the matching result will be more exact. In contrast, a smaller number of sampled points reduces the accuracy of the matching results, but improves the computational efficiency. Spectral descriptors include fourier descriptors and it is usually derived from spectral transform on shape signatures.

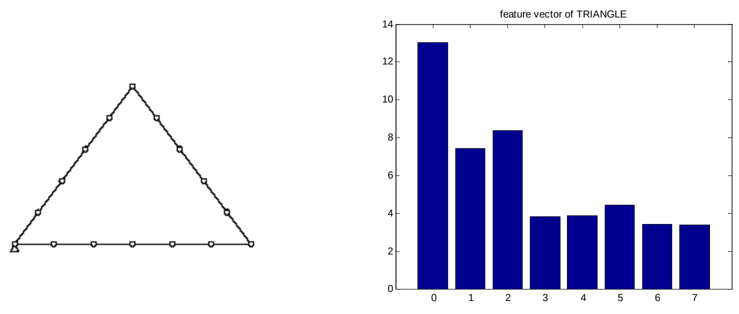


Figure 4.5: “Triangle” and its discrete fourier transformation.

4.3.3 Fourier Series or Fourier Transformation

The necessary method for object entwinement process is can be selected after the difference between fourier series and fourier transformation is made. The important point is if the signal of the grasped objects is periodic or non-periodic. We will now give shortly the application areas of the two methods.

The mathematical relationships between the time domain and frequency domain versions of the same signal are termed transforms. We transform a signal from one representation, $x(t)$ to another representation $X(f)$. A signal's time and frequency domain representations are uniquely related to each other. In both the time and frequency domains our signal exists and with the Fourier transform we make relationship between the two.

4.3.3.1 Derivation of Fourier Series

We begin with a brief review of fourier series. Generally, a fourier serie expansion for a function is a representation of any periodic function as sum of sines and cosines.

The periodic signal of $x(t)$ can be expressed as sum of harmonically related sine waves.[m0039]

$$x(t) = a + \sum_{k=1}^{\infty}\left(a_k \cos\left(\frac{2\pi kt}{T}\right)\right) + \sum_{k=1}^{\infty}\left(b_k \sin\left(\frac{2\pi kt}{T}\right)\right) \tag{4.1}$$

The family of functions called basis functions $\cos\left(\frac{2\pi kt}{T}\right)$ and $\sin\left(\frac{2\pi kt}{T}\right)$ form the foundation of the fourier series. These functions are always present and form the representation's building blocks. They depend on the signal's period *T* and are indexed by k. The frequency of each term is $\frac{k}{T}$. For k = 0, the frequency is zero and the corresponding term a_0 is a constant. The basic frequency $\frac{1}{T}$ is called the fundamental frequency because all other terms have frequencies that are integer multiples of it. These higher frequency terms are called harmonics. The fourier coefficients, a_k and b_k, depend on the signal's waveform. Because frequency is linked to index, the coefficients implicitly depend on frequency.

4.3.3.2 Derivation of Fourier Transform

As we mentioned early fourier series give the frequency domain as a respond to periodic signals. We need a definition for the fourier spectrum of a signal, periodic or not. This spectrum is calculated by the fourier transform. The method that we use for grasping process requires the signals periodic and non-periodic. If the manipulator entwines the object at least one time, it is clear that our signal becomes periodic. For the non-entwinement condition, the signal is not periodic, therefore the fourier transformation have advantage over fourier series. We need a definition for the fourier spectrum of a signal, periodic or not. In the chapter 5 we will determine in detail the state of fourier transformation in our implementation.

In the following, we will go deep in to the theory of the mathematical explanations. The fourier transform theory can be applied in different ways for shape description by closed curves. The fourier transform of a continuous function of *shape(n)* is given by the equation:

$$shape(n) = \sum_{-\infty}^{\infty} F_k \exp(j2\pi nk/T) \tag{4.2}$$

where F_k is the fourier coefficients of the boundary.

The purpose of several entwinement processes is more related with the continuous representation of the shape information. As we mentioned in the previous section, it is very important for the human eye to recognize the coefficients of the fourier descriptors. This is necessary because of the interpretation of this process and we need a periodical signal of the shapes which are grasped by the manipulator.

When dealing with the classification of shapes, the discrete fourier transform (DFT) is used. In the simulation of thesis, we see the results of classification with the help of DFT. So equation (4.2) transforms into:

$$F_k = \frac{1}{T}\sum_{n=0}^{T-1} SHAPE_n . e^{-i.\frac{2\pi k}{T}.n} \tag{4.3}$$

yields the fourier coefficients F_k of order $k = 0,..T-1$, from a periodic sequence of T real values *shape(n)*, assuming it is normalized to T points in the sampling stage, the discrete fourier transform of *shape(n)* is given by with the equation (4.3).

Shape(n) is called shape signature which is any one dimensional function that we represent shape boundary. With this equation, the feature vector of shapes are transformed into fourier descriptors and will be used in the neural network which is responsible for the recognition of the objects.

The implementation of discrete fourier transformation is set up with the help of mlp that produces the necessary information using the periodic signal of objects. This kind of network has two layers and each layer outputs the real and imaginary parts of the coefficients. As a result it becomes a good interpretation of the object grasping method. It can be easily understood if the objects are entwined and also the distance information can be determined. The results will be seen in the next chapter under the “Simulation results”.

Shape(n) indicates our input data for all our links n from 0 to 16. All equations in shape analysis by fourier theory are based on continuous curves. This is the reason why we need one dimensional function that represent shape boundary. However, given the nature of the image, the curve should be described by a collection of points.The discrete approximation has two important effects on the representation of the shape information. Firstly, it limits the number of frequencies in the fourier expansion. Our 16-data-array shape is illustrated with *n/2* number of frequencies. Secondly, it forces numerical approximation to the integral defining coefficients.

4.3.4 Conclusion

Fourier descriptors are a good way to describe and recognize shapes of all kinds. To capture more local features of the shape, higher frequency descriptors must be added. To achieve effective and exact matching, the set of objects, that should be recognized, is required, since the matching-function and the proper number of descriptors to use depend strongly on the given shapes. We see the best results with feature vector that was given manually.

As the result, we have possibility to make an interpretation of grasping objects with the help of fourier transform. It is a practical way of describing the entwinement of objects with 16-arm manipulator.

4.4 Turning function

In this section we will introduce a boundary scalar transform technique called turning function. It is a popular method for polygon shape representation [Ark00], which is invariant to position, scale, and rotation. The turning function represents the tangent of a point on the boundary with respect to a reference axis of an arbitrary orientation. The tangent angle function $\Theta(n)$ can only assume values in a range of length 2π, usually in the interval of $[0,2\pi]$ [Guo01]. During the traversal of the boundary, the tangent at each point is computed. The starting point on the outline corresponds to the origin point on the turning function. Turning function is supposed to examine on all possible reference axes with different degree from 0 to 360, and on all the choice of origin. The formula in our simulation is below.

```
vec = [[0;1],diff(x,1,2)]
len = [sqrt(sum(vec.^2,1))]
for i = 1 : n - 1
cosphi = vec(:,i)' * vec(:,i+1) / (len(1,i) * len(1,i+1))          (4.4)
phi(i) = acos(cosphi)
t = cumsum(phi)
```

In this explanation each point on the object boundary is computed in the space of $0 \leq n \leq 16$ where n depicts the number of sampled points on the boundary. By this way, every link of the manipulator describes boundary information of the shape. At the end, the sum of the angles gives the turning function of boundary.

4.4.1 Grasping of objects

As we previously mentioned, Arkin et al. published an efficient method for comparing polygonal shapes. They establish the notion of the turning function that represents the shape of an object. This algorithm has been applied out to be very fast. The turning function is easily extended from the unit interval by adding or subtracting multiples of 2π.

The parametrization implies the invariance under scaling. The two important degrees of freedom of the turning functions are that the choice of the starting point Q and the orientation of the object. The turning function does not change with the starting point and this verifies a robust approach because of the periodic signal.

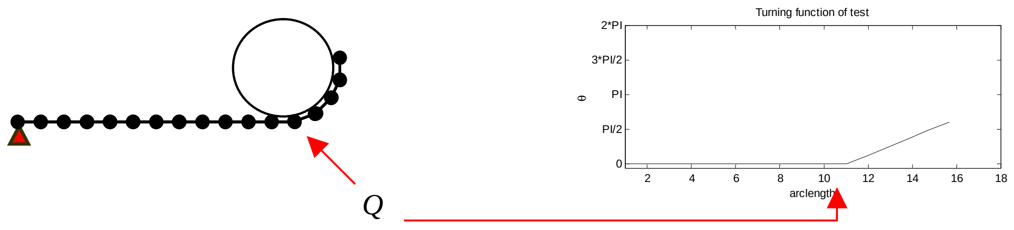


Figure 4.6: Object and its turning function.

For the grasping process we need the turning function as a periodic signal. Since it originates from the cumulation of the periodic sequence of relative turning angles, it is not periodic. However, the relative turning angles of a polygon sum to 2π; thus, the turning function can be transformed into a periodic function with [Mar04].

$$Shape(n)_{periodic} = Shape(n) - \frac{2\pi}{N}.n \tag{4.5}$$

where $Shape(n)$ denotes the value of the turning function at the n^{th} vertex and N is the number of vertices in the polygonal shape. The use of this mathematical derivation in Matlab program is written as

```
n = length(shape)                    (4.6)
shape = shape - 2*pi/n * [1:n]
```

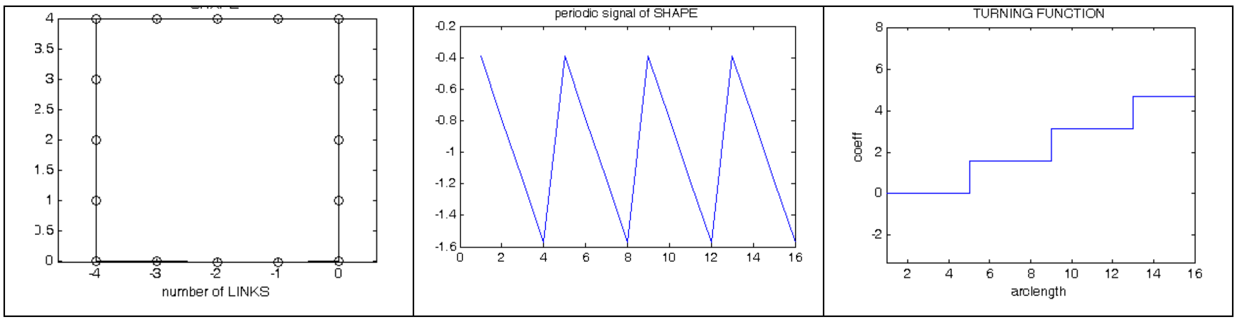


Figure 4.7: The shape, its periodic signal and turning function.

As we see above, the periodic signal and the turning function of a square is illustrated. Actually, these results are received according to the whole-arm grasping process. In the next chapter, we will study the results for all cases of object positions and discuss the differences of coefficients with the help of fourier transformation. The turning function is restricted with the number of links of the manipulator.

4.4.2 Conclusion

In this chapter, we have introduced turning functions as a way of describing the shapes that are going to be recognized during the grasping process. The choice of grasping point and orientation of the objects do not effect the result of turning function with tentacle case object recognition. The difference between the manipulator and the starting point of object gives us the distance information. Also we have knowledge of distance information by investigating the periodic signal of turning function. Rotation of the objects only cause a vertical shift in turning function.

4.5 Nyquist Theorem

In the following, we will discuss the importance of the distance between the links in the recognition and grasping process. Firstly, we will introduce the theorem and then indicate the effects of the frequency of the shape classification.

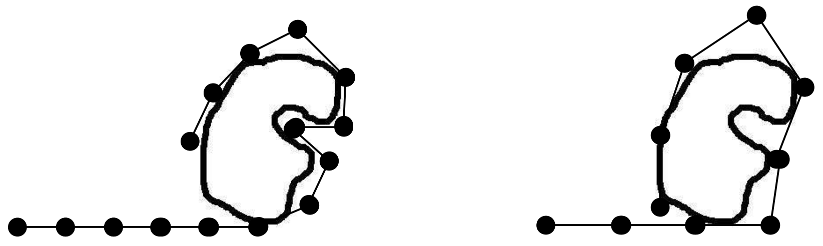


Figure 4.8: The different length of links in entwinement process and its effect on recognition process.

As we see in fig 4.8, the length of the links of the manipulator plays an important role in shape entwinement and recognition process. The length of the link in fig 4.8 (a) is smaller than in the link in fig 4.8 (b). The interpretation of the lengths of the links can be assumed as follows:

- We indicated with fourier descriptors that, the lower frequency descriptors contain information about the general features of the shape, and the higher frequency descriptors contain information about finer and the small details of the shape. Here as we see, the longer links the manipulator has, the smaller shape information we have. The purpose of the manipulator must be good decided if several different shapes will be involved in recognition process.
- If the length of the links are small, then we will need more links than we have and in this case, the grasping process of the shapes can not be achieved properly. Due to the lack of entwinement of objects, the shape information will be incomplete and classification results will be wrong.

The nyquist theorem says that, the sampling rate must be at least $2f_{\max}$. If the sampling rate is less than $2f_{\max}$, some of the highest frequency components cause undesirable condition that is a form of distortion called aliasing. When this happens, the original signal cannot be uniquely reconstructed from the sampled signal. If B is the bandwidth and F_s is the sampling rate, then the theorem can be stated mathematically [Kor04].

$$2B < F_s \tag{4.7}$$

Below, the implementation of function is illustrated. This formula applies fourier transformation and strip contents over nyquist frequency and static content.

$$\text{ftc} = \text{abs}(\text{fft}(\text{shape})) \qquad (4.8)$$
$$\text{ftw} = \text{ftc}(1{,}2{:}\text{ceil}(n/2)-1)$$

where *ftw* is the feature vector. The command ceil rounds the elements of (n/2) to the nearest integers towards infinity. The following command plots the feature vector of shape and the dimensionality of a feature vector consisting of the fourier transformed turning function is smaller that a feature vector that includes the turning function itself which helps to reduce the effects of the curse of dimensionality. Therefore the feature vector has to contain the coefficients $k = N/2-1$ where k denotes the band-with and *N* denotes the sampling rate F_s

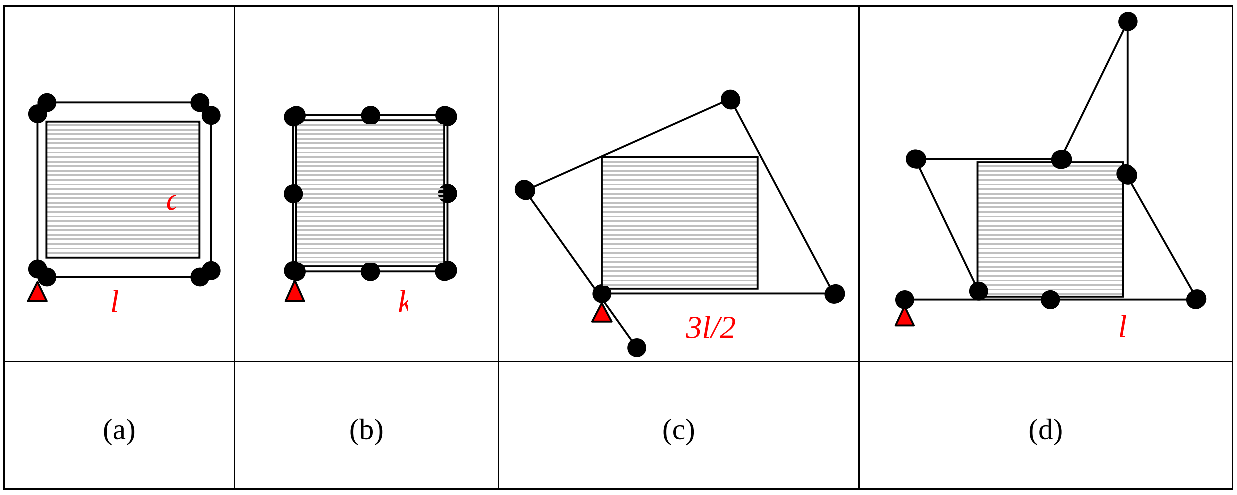


Fig 4.9: The entwinement of square with different length of links.

Fig 4.9 represents four types of grasped squares. In the cases of grasping process, we used different length of links in order to realize the entwinement process properly. If we consider the length of manipulator as *l* and the length of square as *a*, it will be possible to write a relationship between these two parameters. To realize the grasping results correctly, the length of manipulator *l* must be equal or smaller than the length of square *a*. In this case we need a second assumption in order to have sufficient results. In fig 4.9 (b), the length of a link (*k*) is half of the length of one side of square (*a*). We can generate a formula as follows:

$$\text{The length of a link} = \frac{a}{2}, \frac{a}{4}, \dots, \frac{a}{n} \qquad (4.9)$$

The smaller length of link the better entwinement results. In the other hand, in case of $l > a$, the grasping results will be not sufficient (see fig 4.9 (c)).

We have an interesting situation in fig 4.9 (d). The lengths of links in fig (a) and (d) are equal but the grasping results are different. We must make an addition to the rule that we described in (4.9). The joints of the links and the edge of the shape must come together. It is clear that if the joint is placed in the middle of the side of the square, the manipulator will not entwine the shape and this will not prevent the true grasping results. On the edges of the square (see fig. 4.9 (d)) the shape of the manipulator becomes like triangles and of course is not desired.

In the following we will continue to discuss the results for the other class of shapes individually.

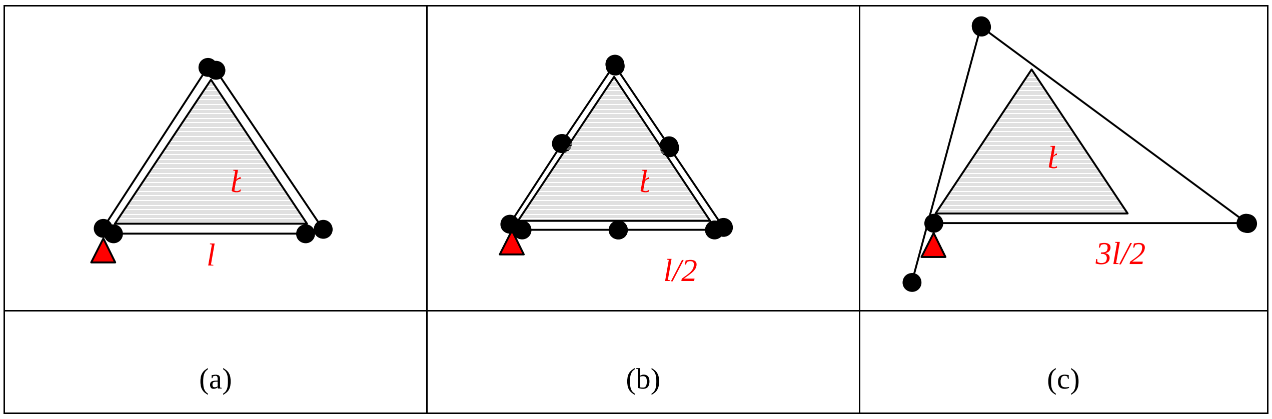


Fig 4.10: The entwinement of triangle with different length of links.

As we previously for the square mentioned, all the conditions are also valid for the triangles and for the rectangles. But we will denote a point for the rectangles. In fig. 4.11 (b) and (c), we see the same result of having small triangles on the corners of the shapes. To prevent this problem, the length of a link must be smaller than the smallest side of the rectangle as shown in fig. 4.11 (a).

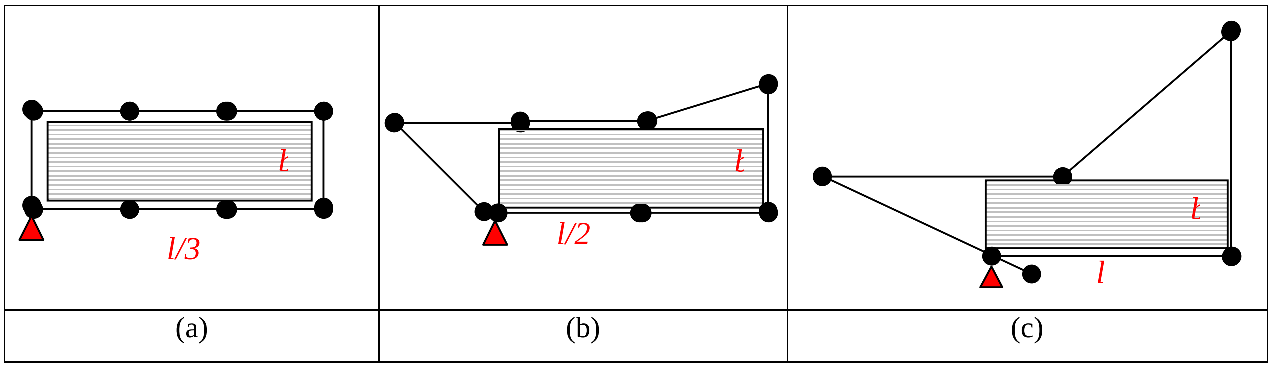


Fig 4.11: The entwinement of rectangle with different length of links.

In the following, we will deal with the shape circle as a last shape in our training set. The necessary conditions for the entwinement is related with the radius of circles. The length of the sides of the shapes were important for squares, triangles and rectangles. In fig 4.12, we see three circles that are grasped by the manipulator that has different length of links.

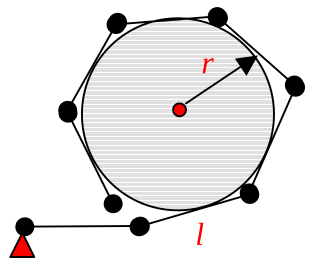
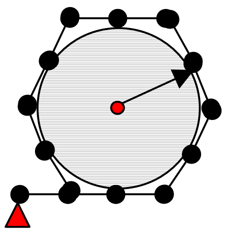
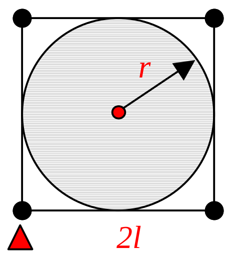
		
(a)	(b)	(c)

Fig 4.12: The entwinement of circles with different length of links.

We assume two different cases for the circle grasping.

1. The length of the manipulator is constant and the radius of circle changes
2. The length of the manipulator changes and the radius of circle changes

For the both cases, it is clear that the critical limit of adequate results of being grasped by the manipulator is that the length of a link (*l*) must be smaller than the radius (*r*) of circle (see fig. 4.12 (a) and (b)).

$$l < r \tag{4.10}$$

In fig. 4.12 (c), the length of the link is bigger than the radius of circle and in this case our manipulator gives out square instead of circle. Also, from the equation (4.7), it is known that the nyquist frequency must be half or smaller than the half of the sampling rate (*f*).

$$\frac{1}{l} = f \tag{4.11}$$

If the length of the link increases, the frequency of sampling rate will decrease. This can be released by applying the equation (4.11) in (4.17) as follows

$$2B < \frac{1}{l} \tag{4.12}$$

By increasing the length (*l*), the bandwidth will decrease and the radius of circle will again increase in order to yield sufficient grasping results.

5 Simulation

In the following, we will discuss implementation details and the results of the simulations we carried out. In the previous chapter, the theoretical background of neural object recognition was achieved to be explained. For this purpose, a simulation was proposed basing on the numerical software package MATLAB/SIMULINK.

Neural Network Toolbox, Version 7.0
Intel Celeron M Processor 1.5 GHz. 992 Ram

5.1 Classification and Grasping

In this section, we will describe implementation details and simulation results of the neural object recognition and classification. Furthermore, we hypothesize the object to be fully entwined, several times entwined and even not entwined meaning the manipulator covers the whole object boundary. The objective of this thesis is to develop precise and efficient methods to identify the location and orientation of a particularly object model. Secondly, adaptive application of a selected feature and limitation of a feature character under the concept of data representation were involved.

In this thesis, two different problems were discussed. The first problem is concerned with the "tentacle case problem" with one-dimensional data arrays. The second problem tried to be solved is to investigate the „Linear object scanner case". In this chapter, we will discuss each problem separately.

For the purpose of object recognition, the manipulator was assumed to have 16 links. Actually, the more links in usage, the better the obtained results. This is due to the high number of feature vectors that characterizes more information about the shape. The turning function in the training sets as well as the test set used to verify the object recognition system was specified with the program. RBNN is useful for classification and mlp is suitable for entwinement for our objects. In the following, three neural networks will be introduced separately and the results will be given in the section of "Simulation results".

All of the simulations were performed in MATLAB using the Neural Networks Toolbox.

5.1.1 Classification process based on RBNN

In this section a PNN- (Probabilistic neural networks) a kind of radial basis network suitable for classification problem- is implemented. The input to the network is a 16-dimensional vector that represents the periodical turning function.

In the following, the structure of network using Matlab help is introduced. Newpnn creates a two-layer-network. The first layer consists of radial basis transfer function neurons and calculates its weighted inputs with the euclidean distance weight function. The weights are applied to an input with the help of weight functions to get weighted inputs. Net input functions calculate a layer's net input by combining its weighted inputs and biases. The second layer has neurons that uses transfer functions which calculate a layer's output from its net input, and calculates its weighted input dot product weight function and its net inputs. Only the first layer has biases. The first layer includes sigmoidal units and the second layer holds a single linear unit.

RBNN (Radial Basis Neural Networks) is trained to classify objects to be either rectangular, square, circular or a triangle

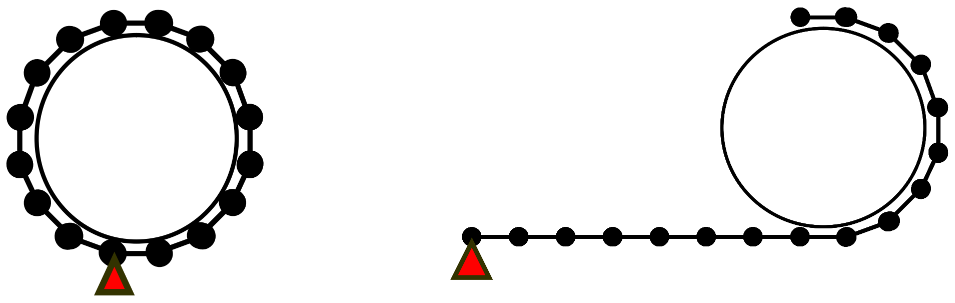


Figure 5.1: The grasping problem with a 16-DOF manipulator. Fully entwined and partially entwined objects are illustrated.

5.1.1.1 PNN for classification

In this thesis, focus is laid on RBF-networks. Its features are extracted with the help of the neural network introduced in the previous section. They can be classified by using a special type of the RBF networks described in section 4.2.3.1.

Probabilistic neural networks are a kind of radial basis network suitable for classification problems. In chapter 3, several advantages of pnn over back propagation (BP) networks are indicated, i.e. the improvement of training effects. This is due to the enhanced network architecture. In case of enough input data, the pnn will act as a Bayesian classifier. Pnn allows true incremental learning by the opportunity to add new training data at any time without requiring retraining of the entire network. Due to the statistical basis of the pnn, it can give an indication of the amount of evidence it has for basing its decision [pnn].

In pnn, probability density functions such as the Gaussian are used as a basis function and are centred around the training cases. The weights between the hidden and output units are set to the prior probabilities of the class that is represented by a specific output unit alternatively. For each hidden unit, a weight of 1 is used connected to the output the current case belongs to, while all other connections are set to zero.

The topology of the network is feed-forward combined with an unsupervised training paradigm. The outputs of this network are derived from the probability of the input belonging to the class that is represented by the output unit. There are 4 type of classification results. Square, circle, rectangle and triangle are used as training sets resulting in the desired outputs. Here a classification problem is defined with a set of inputs *P* and class indices *T*.

```
y = sim(MLP, {circle'})        y = sim(MLP, {triangle'})      y = sim(MLP, {square'})        y = sim(MLP, {rectangle'})
coeff = abs(y{1}+i*y{2})       coeff = abs(y{1}+i*y{2})       coeff = abs(y{1}+i*y{2})       coeff = abs(y{1}+i*y{2})
P = [P, coeff]                 P = [P, coeff]                 P = [P, coeff]                 P = [P, coeff]
```

The inputs are used in order to create a class which test set will be trained according to produced feature vector shown above.

```
T = ind2vec([1 2 3 4])
```

Matlab help documents that *ind2vec* takes one argument and returns sparse matrices of vectors, with one number in each column. As indicated above, there are 4 types of classes. In order to show the meaning of the numbers it can be stated that 1 points out a “circle”, 2 a “triangle”, 3 a “square”, 4 a “rectangle”.

```
net = newpnn (P,T)
```

This function creates a network and takes two arguments of *P*. *P* represents the input vectors being compared with the *T* target class vectors. The classification process is performed on the basis of the input arguments. The class indices are converted to target vectors and a pnn is designed and tested. As below described a new vector with the network is classified. First the desired object is given as “Test” to enable a comparison with the input vectors. By this way, a MLP network is entrained to calculate the euclidean length of a vector. This enables to achieve the classification independent of translation, scale and rotation.

```
class = sim(net, coeff')
vec2ind(class)
test = vec2ind(class)
```

vec2ind transforms vectors to indices in order to show the four different classification results. The newpnn method is simulated with the sim (Simulate a Simulink model) command. Computed values of fft are stored in newpnn and simulated according to the probabilistic neural network.

5.1.1.2 Training

Normally, the training of the network is repeated for many examples in the set until the network reaches a steady state, lacking any further significant changes in the synaptic weights. In this condition of the learning process, complex problems were processed with several hidden units. The network learns from examples by constructing an input-output mapping for the problem that is concerned with. The choice of the number of hidden units and the learning parameter plays a huge role in learning paradigm. An optimal network structure has to be found for solving related tasks.

A set of input vectors are applied to a network that is updated at each step, until some stopping criteria- for example maximum number of epochs, a minimum error gradient, an error goal are met.

As a training set, one kind of data input is used for each object, the network is trained with 50 epochs and the learning rate is determined as 0,6. Increasing number of epochs decelerates the network speed. Obviously, the more training sets in usage, the more absolute classification and recognition results are received. It is not recommended to have large training sets, as these are decreasing the speed of the processing.

For the examination of the tentacle and the linear object scanner case, different classification methods were used. The classification of the tentacle case leads to adequate results. The training parameters that are shown below were used in the simulation of all recognition problems.

```
net.trainParam.epochs = 50
net.trainParam.goal = 0.01
net = train(net,p,t)
```

As depicted above, the network is trained with the optimal training number of 50 both for the tentacle and object scanner case problem. For the mlp, the weight matrix is determined analytically; no need to perform an additional training on the net.

5.1.2 Grasping process based on MLP

In chapter 4, the concept of feature extraction is described. In the following, the representation of shapes of the turning functions, which can be easily shown through the mathematical explanations describing them as angels will be discussed. In the turning function, summation of the relative joint angles results as coefficients. The length of links are accepted equal and the arc length between the vertices of the polygonal shape can easily be normalized to unit length, which provides invariance with respect to scaling of the object. The output has two layers, each of them performing the real and imaginary parts of the fourier coefficients beyond the nyquist frequency with the exception of the zeroth order coefficient. The feature vector comprises the magnitudes of the complex fourier coefficients.

Being of importance of this thesis, the entwinement process is implemented with a kind of mlp network. This network sets up a mlp network that performs a discrete fourier transformation. It computes only $N/2$ coefficients; both the static component and the coefficients above the nyquist frequency are stripped. The network uses one input and the two layers in which the transfer functions are selected as purelin. Purelin takes one input and calculates layer's output from its net input.

The two layers that take part in the network output the real and imaginary parts of the coefficients. The weight matrix is determined analytically so no need to perform additional training on the net. The fourier serie can be written as

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} shape_{periodic}\left[\cos\left(\frac{2\pi kn}{N}\right) - i\sin\left(\frac{2\pi kn}{N}\right)\right]$$

The real and the imaginary part of the coefficients become apparent and the functions do not change with the value of periodic signal $shape_{periodic}$. They only depend on the period of sequence *N*, the order of the coefficient *k* and the index n.

The fourier serie which is performed in mlp network has real and imaginary parts. The input given to the network is 64-dimensional data arrays to see the high frequencies in detail for the grasping process. The output includes both the real and the imaginary part of the fourier coefficient of order *k* of the given *test* data array. Only the absolute values are observed in the figures.

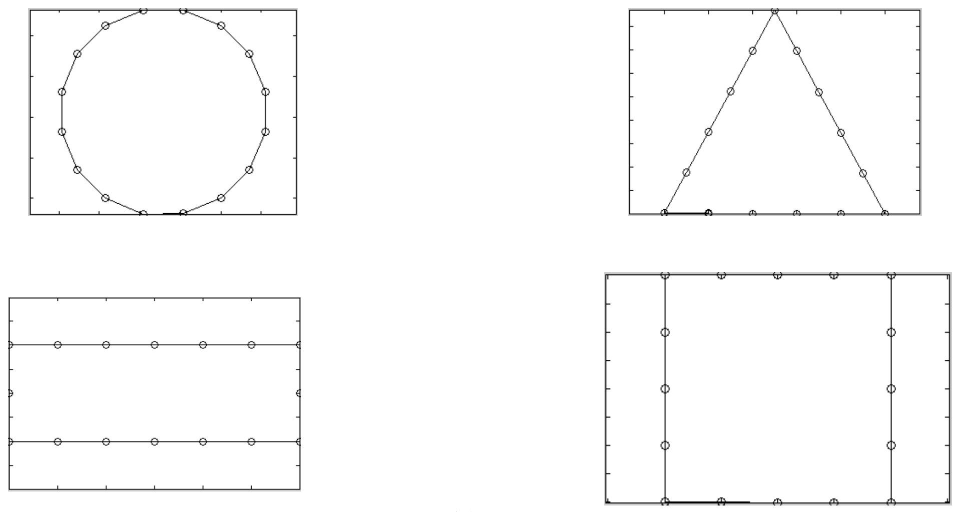
The method used in mlp is based on the periodic signal of depicted objects. The turning function of the angels describing the object boundary information is calculated and transformed to a periodic signal as mentioned above. Depending on the grasping of objects, the signals can not be even periodic. The reason using the fourier transformation is that it is applicable to all signals if they are periodic or not. In this way, the necessary information enables the interpretation for entwinement. The spectrum of coefficients must give regularly descending values for the objects that are being entwined more than one-time grasped objects. The reason is that fourier transformation is applicable both to periodic and non-periodic signals. If the objects are not fully grasped, the spectrum will have coefficient values at the high frequencies like a wavy form. Otherwise, the spectrum will decrease to a constant value periodically.

In the following, the details of the results that were received by using mlp in case of not-grasped, whole-arm grasped and more than one-time grasped objects will be examined.

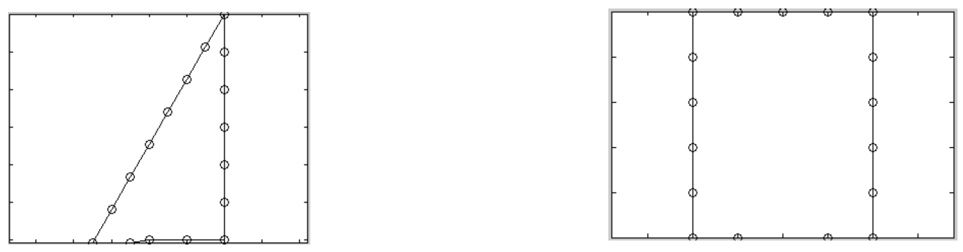
5.2 Simulation results

In this section, we will introduce the simulation process separately and show the results for each problem. The different grasping process for three kinds of object cases is as follows;

1. Not fully grasped objects
2. One-time grasped objects
3. More than one-time grasped objects



(a)



(b)

Figure 5.2: Shapes in the training set (a) and the test set (b).

The procedure of neural object classification has two main process respectively classification of objects and grasping process. This is necessary for deciding if the other objects are entwined or not entwined that have different position, rotation and scaling information. Possible input data for the neural network based on classifier consists of joint angle information. The neural network has to recognize the object's shape and assign it to certain classes. For instance, circular, rectangular, triangular and square shapes should be considered.

An overview of the training set and test set being used to train is shown in fig. 5.2. For the present work of thesis, the fourier series of the turning function determines the several times of object entwinement and the coefficients change rapidly. As test set, several one-dimensional data arrays were used with different distance information to the manipulator.

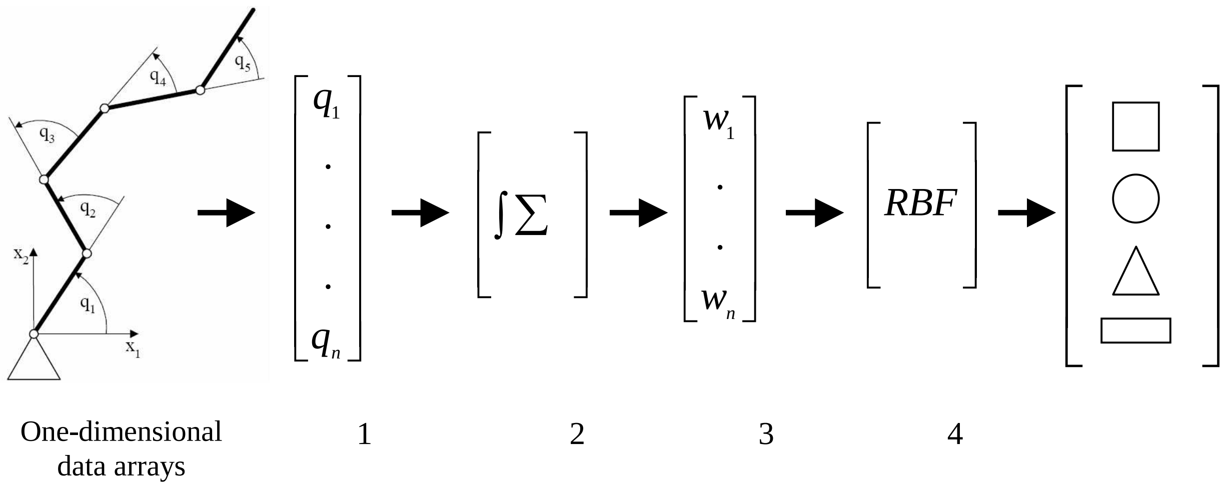


Figure 5.3: Investigating of the tentacle case problem and the classification process.

Fig. 5.3 shows the method that was used in this thesis. First, winkles are given manually to the network. They are transformed into two dimensional data arrays for the plotting of shapes. Then, the turning function of the boundary points is computed. However, the computed function is not invariant to rotation and the turning function is not sufficient to represent a feature vector for making classification from the classes of objects. The periodic function is necessary for the fourier series. The obtained turning function of the object which the fourier transformation is applied, strip contents over nyquist frequency.

Except the zeroth fourier coefficient of the turning function is invariant with respect to the vertical shift coaused by rotation of shape. The magnitudes of the fourier coefficients are also invariant with starting vertex. The feature vector contains up to order $k = N/2 - 1$, therefore the dimensionality of periodicised turning function is smaller than the feature vector that containes the turning function itself.

After implementation of fourier transformation to the periodic signal of the object, the necessary coefficients are taken for deciding grasping process. As a last step, the classification with pnn is proceed and tested for the right results of the objects as square, circle, triangle, rectangle.

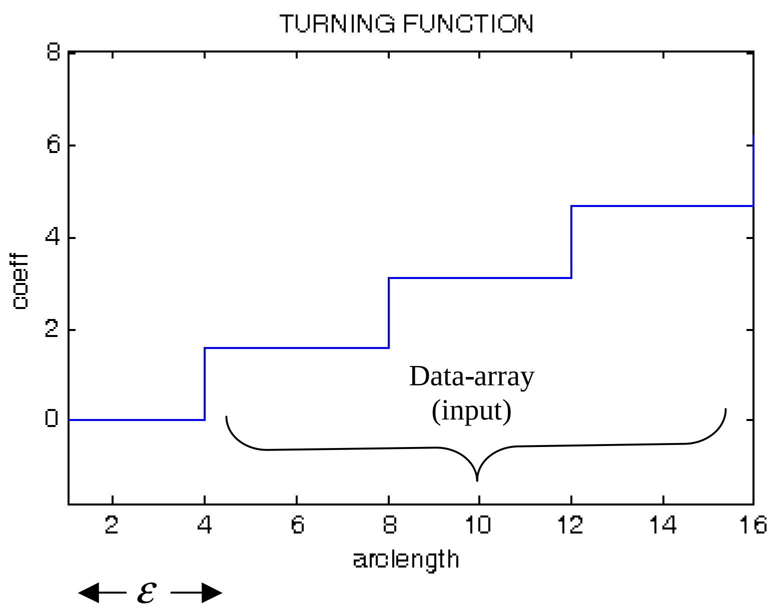


Figure 5.4: Tentacle case „offset-problem“. Turning function of an object. ε is the position information of the object according to the manipulator.

The first problem that is concerned with in this thesis is about the objects that are placed away from the manipulator. It is assumed that the object is entwined or partially entwined. To solve the offset-problem, the fourier transformation of the data inputs of the object information is used. It is very easy to predict that if the object is placed away from the manipulator, the fully entwined process could not be performed and the grasp information of the shape will give false results. The offset-problem was simulated with circles that have different location information. The aim of the object recognition is to define the shape information if the object is entwined or not.

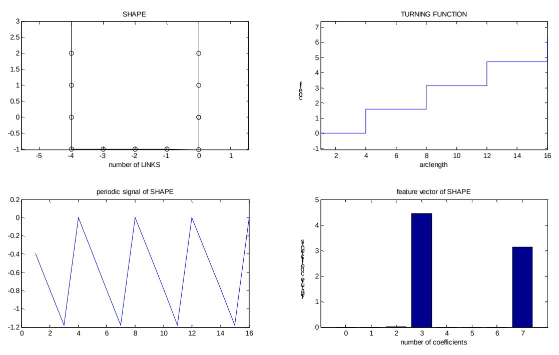
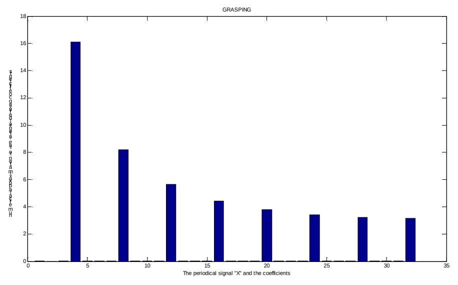
TEST OBJECT	**GRASPING RESULTS**
	
(a)	(b)

Figure 5.5: Based on tentacle case entwinement problem. The test array is plotted in two-dimensional plane and periodic signal of turning function, turning function and the fft coefficients are shown (a). The coefficients of periodisiced turning function are illustriated (b).

The solution of problem with a simulation is performed in order to achieve the grasping process even recognized with human eye. It is important to recognize the results not only with mathematical explanations also the figures must help to interpret the grasping process. More detailed searching results will be founded in the next section.

The extraction of the feature vector consists of three important points

1. The turning function of shapes which are given manually and transformation to a periodic signal that is necessary for the fourier transformation

2. fourier transformation of the periodic signal

3. computing of the feature vector that comprises the magnitudes of the complex fourier coefficients

After feature extraction of shapes, it is able to make a classification with the help of two different mlp networks. In the following, we will deal with these two networks.

5.2.1 Investigating tentacle case problem

The given test sets as an input are one-dimensional data arrays and for the plotting of each set, they must be converted into two-dimensional data arrays in order to see how they look like. This is helpful to recognize if the object is entwined or not without any mathematical explanation. Each specified point is interpolated and the results are plotted in the graphical user interface.

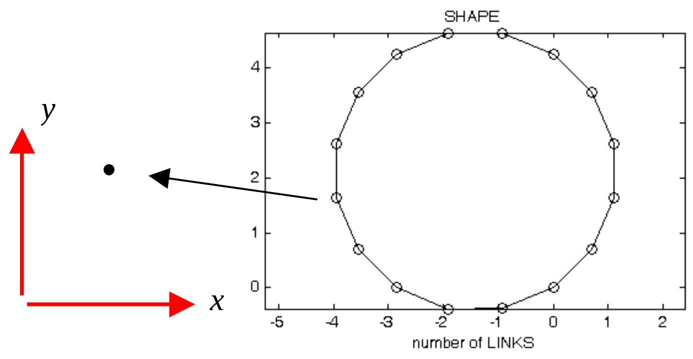
test = [pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8]


Fig 5.6: The test set and interpolated result of previously specified points.

The shape transformed from the points is done with the help of a little program. The coordinates (*x*, *y*) or each angle (pi/8) is computed and plotted as shown in fig. 5.6. As discussed later, the usage of plotting a shape is important for the grasping decision.

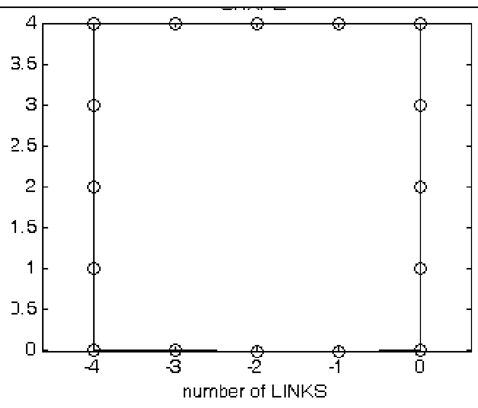
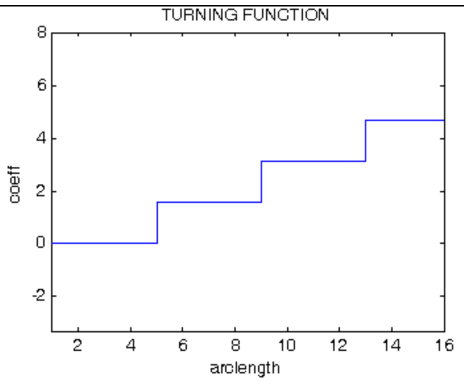
test = [0 pi/2 0 0 0 pi/2 0 0 0 pi/2 0 0 0 pi/2 0 0]	
	
The plotted shape	The turning function of plotted shape

Fig 5.7: The given test test and turning function of plotted shape.

The turning function is not invariant to rotation; rotation of the shapes causes a vertical shift of the turning function. Additionally, the course of the function depends on the choice the cumulation of the relative tangent angles namely the starting point of shapes. By plotting the test arrays into a shape, it is possible to make an interpretation if the objects are grasped or not.

One-dimensional data arrays are converted into two-dimensional data arrays and plotted to realize the shape information of object. The turning function of a polygon gives the cumulative sum of angles between the counterclockwise tangent to the sides of the polygon and the *x*-axis as a function of the arc length *s*.

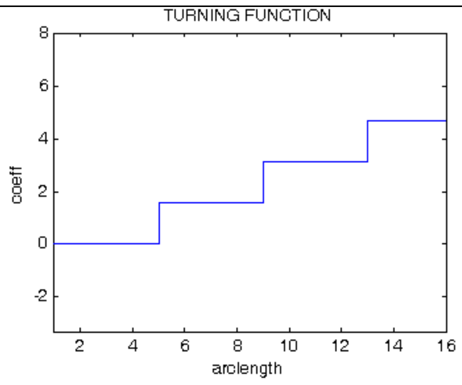
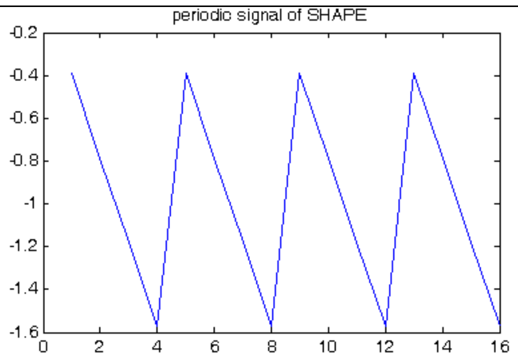
test = [0 pi/2 0 0 0 pi/2 0 0 0 pi/2 0 0 0 pi/2 0 0]	
	
The turning function of plotted shape	The periodic signal of turning function

Fig 5.8: Turning function of shape and periodic signal which is necessary for fourier serie.

The Fourier series as given in chapter 4 requires the sequence turning function *shape* to be periodic. Thus, the turning function of shape (*t*) can be transformed into a periodic function.

n = length(t)

periodic signal = t - 2*pi/n * [1:n]

where n is the number of links and turning function is converted to periodic signal that is necessary to apply fourier transformation and strip contents over nyquist frequency and static content.

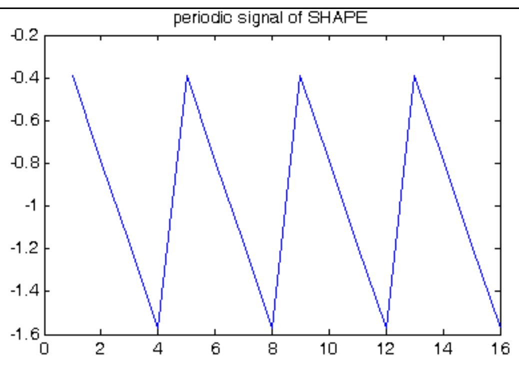
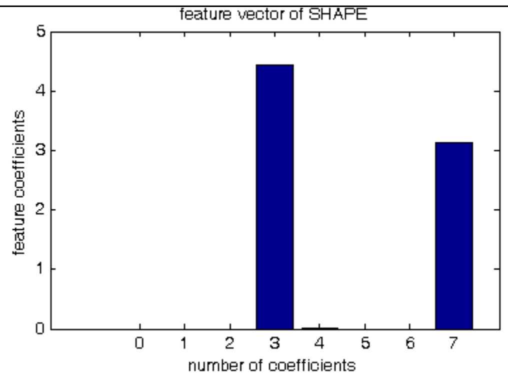
test = [0 pi/2 0 0 0 pi/2 0 0 0 pi/2 0 0 0 pi/2 0 0]	
	
The periodic signal of turning function	Applied fourier transformation and feature coefficients

Fig 5.9: Periodic signal of turning function and feature coefficients.

Fourier transformation is applied to the periodic signal of turning function of shape and contents are striped over nyquist frequency and static content as follows.

fta = abs(fft(periodic signal))

ft = fta(1,2:ceil(n/2)+1)

The absolute fourier transformation of periodic signal is computed and return feature vector *a*.

a = ft

The matching results will be exact by using more descriptors but on the other side the accuracy reduces. Each coefficient is calculated from boundary points and sensitive to all the points of shape. All of the objects assumed to be have different distance information to the manipulator. The fully entwined object, its turning function, feature vector, periodic signal of object and fourier transformation are given together with the result of the classification. The other kinds of training sets have different distance information to the manipulator and the objects are assumed to be fully entwined, partially entwined or several times entwined.

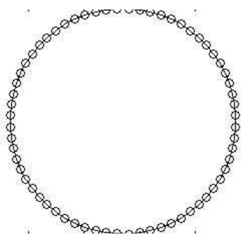
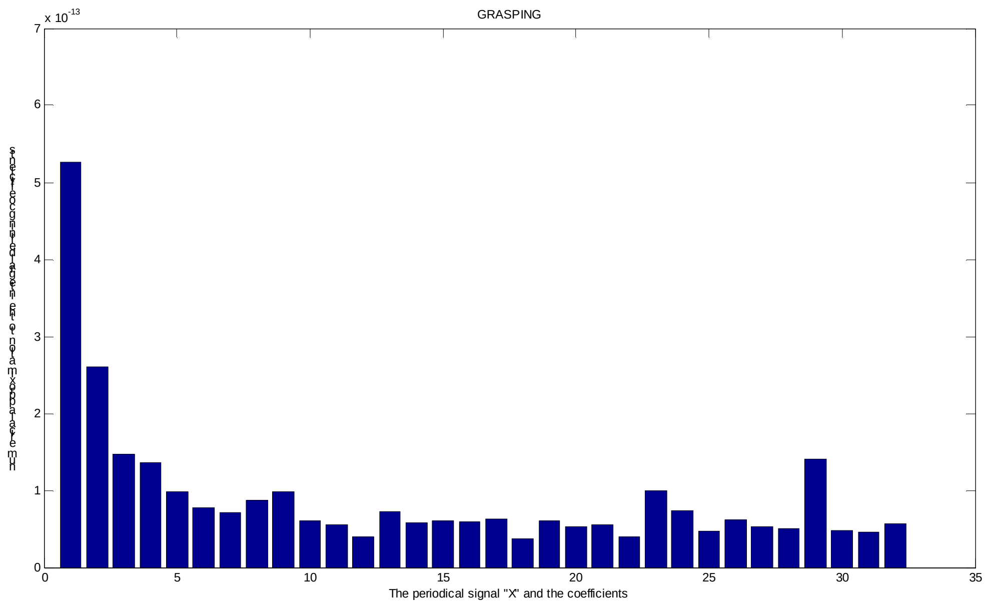
The training set of 4 types of square, circle, triangle and rectangle object information were utilized. The vectors that we gave manually for describing the objects consist of zero points that imply the smooth, the angles that imply the break points on the objects. The fft coefficients of these vectors are used for the classification results.

The training set used in the program for the classification is a 16 data-array that gives the object boundary information. The classification process is achieved with the help of fft that is implemented to the training set. Fft computes the discrete fourier transform of the object boundary for each points up to number of the links. The lower fft values store the general information of the shape and the higher frequency the smaller details.

The discrete approximation has two important effects on the representation of the shape information. Firstly, it limits the number of frequencies in the fourier expansion. Our 16-data-array shape is illustrated with *n/2* number of frequencies. Secondly, it forces the numerical approximation to the integral defining coefficients.

In the following, implementation details and simulation results of the neural object recognition will be described. Furthermore, we claim the objects to be fully entwined, that is, the manipulator covers the whole object boundary. In our simulation of object recognition, it is assumed that the manipulator to have 16 links. Two different neural networks are trained in order to classify objects to be either rectangular, square, circular or a triangle. The turning function in the training sets as well as the test set used to verify the object recognition system was specified manually. All of the simulations were done in MATLAB using the Neural Networks Toolbox.

A linear mlp network that determines the fourier coefficients of the boundary function is implemented. The input to the network is a 16-dimensional vector is converted from the turning function. The output is organized in two layers, each of which holds the real respectively complex parts of the fourier coefficients beyond the Nyquist frequency with the exception of the zeroth order coefficient. The feature vector includes the magnitudes of the complex fourier coefficients. Thus, a two-layer mlp network is entrained to perform this task. The first layer includes six sigmoidal units and the second layer holds a single linear unit. For the detailed spectrum of fft, 64-dimensional vector set is used for the grasping process. By this way, the form of spectrum can be observed and sufficient results are interpreted. For the classification process, a pnn is entrained.

	circle= [pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32] This circle is fully entwined and the coefficients converge on a stationary value. Convergency shows that the grasping of objects is achieved.
	

f

Figure 5.10: The 64-dimensional data arrays for grasping process in mlp network. The discrete approximations with the limited number of frequencies in Fourier expansions are called “coefficients”.

Describing the results more detailed in fig 5.10. the whole-arm grasped circle is introduced. The value of each winkle is pi/8 (22.5°) and the perimeter of circle can be calculated as $16 \times pi/8 = 2.\pi$. All of the winkles of our objects that we used in this thesis is proportional to $2.\pi$.

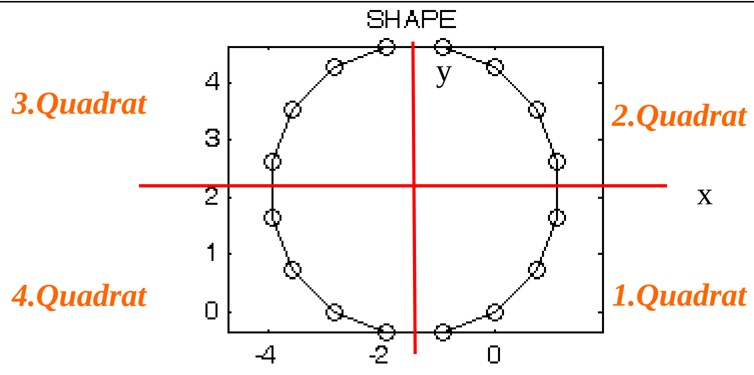
The MLP network for grasping process has two layers consisting of the first layer {1} and the second layer {2}. The first layer {1} maps the cos part of plotted shape “E” and the coefficients that is implemented in Matlab program as “coeff”. The second layer {2} maps the sin part of plotted shape “E” and the imaginary part of coefficients. The 32 frequencies are symbolized with “E” and taken after the turning function is transformed to a periodic signal. In Matlab the periodic signal and the coefficients are taken as follows.

```
y = sim(MLP, {(E)'})
coeff = (abs(y{1}+i*y{2}))'
```

First the periodic signal "E" is obtained from the turning function. "y" outputs the sim('mlp') that will simulate mlp model using all simulation parameter dialog settings including Workspace I/O options. The results of "y" are logged and the absolute values of coefficients are taken. The last command "*A*" displays the parameters in the same window to enable to observe the results.

For the coefficients, the absolute values of real and imaginary part of y are calculated in layer {1} and layer {2}.

As seen below, test object is a circle and whole-arm grasped. To observe the results of a plotted test arrays, "x(1,:),x(2,:)"is written in the command window of Matlab.



X: The value of plotted turning function of shapes.
The values of plotted shape on x-plane
0 0.7071 1.0898 1.0898 0.7071 -0.0000 -0.9239 -1.9239 -2.8478 -3.5549 -3.9375 -3.9375 -3.5549 -2.8478 -1.9239 -0.9239
The values of plotted shape on y-plane
0 0.7071 1.6310 2.6310 3.5549 4.2620 4.6447 4.6447 4.2620 3.5549 2.6310 1.6310 0.7071 -0.0000 -0.3827 -0.3827

Fig 5.11: The plotted turning function "x" of shape performed in MLP.

As shown above, the cos and sin of x are illustrated to receive the shape information. The first row describing the cosines values of angle (pi/8) of x increases in the first quadrat and reduces in the second and third quadrat while the sin values are increasing. In the fourth quadrat both the cos and sin values decrease. The value 0.7071 is the cos of (pi/8) the first angle. For each point of shape the sin and cos values are computed and added one another until the whole shape information is taken.

Fig 5.11 includes the necessary information of the grasping process of a circle. It is also easy to predict that the plotted shape give all the information that is needed.

test = [pi/2 0 pi/2 0 0 0 0 pi/2 0 pi/2 0 0 0 0 pi/2 0]

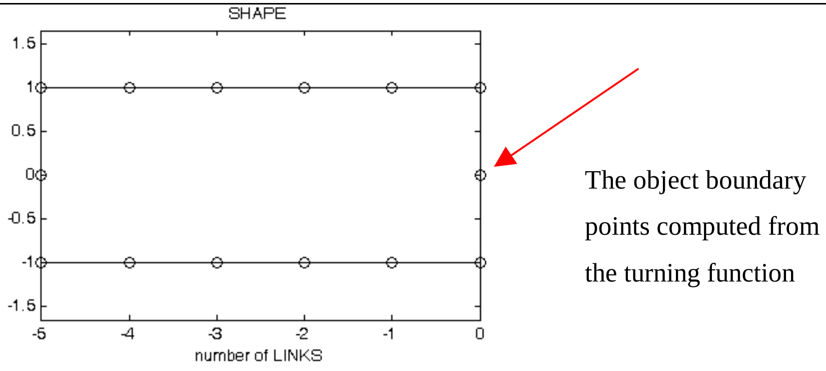


Fig 5.12: The plotted shape of one-time grasped rectangle.

Writing down the cos and sin values of plotted turning function of shape one under the other, the coordinates can be computed (*x*,*y*).

The number of grasping process is obtained by counting the points on the object boundary. In fig 5.12 the number of boundary points (15) is smaller than the number of links (16).

the number of points > 16	not fully grasped
the number of points < 16	more than one-time grasped
the number of points = 16	whole arm grasped

If the points are less than the number of links, the object is assumed to be grasped more than one-time. If the object boundary (8) has as the half number of links (16), in this case the object is said to be two-times grasped. The plotted turning function of shape “x” includes the cos and sin information of the angles between points.

In the next section, we will discuss the grasping results for different object cases and try to explain a general idea by using fourier transformation with the help of mlp.

5.2.1.1 Not fully grasped objects

In the following, we will study the grasping results for objects that are not entwined. The missing grasping cases of all objects will be searched and the results will be discussed.

Not fully grasping process is valid if the objects are placed away from the manipulator. In the fig. 5.13, it will be searched the necessary parameters that play important role to decide in which condition our object is.

RECTANGLE

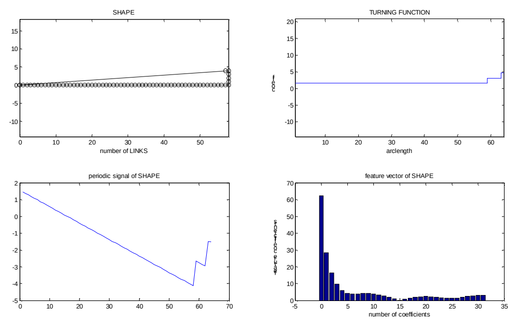
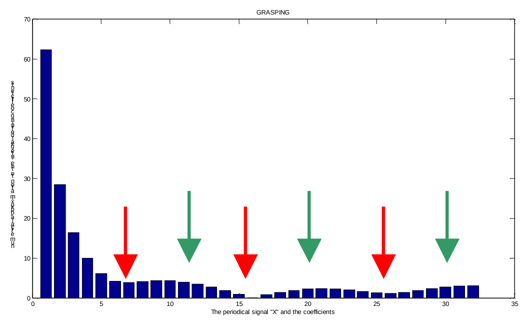
test = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 pi/2] Classified as **CIRCLE**	
	
62.2025 28.5078 16.3591 9.8899 6.0995 4.2147 3.8211 4.0784 4.3193 4.3100 4.0141 3.4654 2.7236 1.8594 0.9457 0.0701 0.7630 1.4385 1.9402 2.2437 2.3421 2.2466 1.9900 1.6357 1.3056 1.2071 1.4560 1.8977 2.3579 2.7386 2.9860 3.0715	

Fig. 5.13: Not-fully entwined object.

The real and imaginary parts of the coefficients are computed from the plotted shape of turning function of each shape. For the grasping process, the absolute value of the summation of coefficients are taken. In case of not fully grasped object, the integral defining coefficients of periodisiced turning function of shape do not decrease with the high frequencies showing the signal is not periodic and thus the object is not fully grasped.

SQUARES

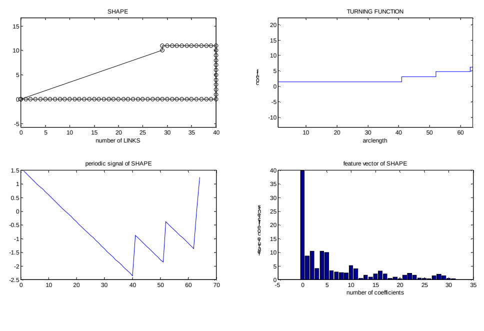
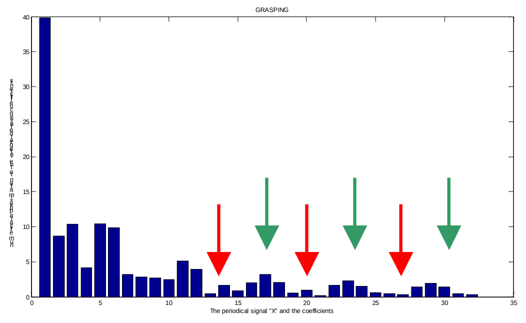
test = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 0 0 0 pi/2] Classified as **CIRCLE**	
	
39.8039 8.6634 10.3422 4.1597 10.4083 9.8530 3.1959 2.7923 2.6510 2.4468 5.0947 3.9461 0.4759 1.6400 0.8374 2.0005 3.1485 2.0566 0.4989 0.9092 0.1702 1.6392 2.2331 1.5144 0.5827 0.4649 0.3301 1.4301 1.9302 1.4083 0.4754 0.3321	

Fig. 5.14: Not-fully entwined object.

The values of coefficients decrease up to seventh frequency and begin to increase irregularly. This condition continues after one after as the spectrum larges. As seen below, the spectrum gives irregular coefficient values that a periodic data arrays can not be observed. The periodic signal degrease with the zero points in one-dimensional data array which implies that there is “0” angle between the links of the manipulator. The red-plotted array shows the pit and the green arrays the peak points of spectrum. The spectrums have wavy form of coefficient arrays and in case of not fully grasped objects these results are observed.

CIRCLES

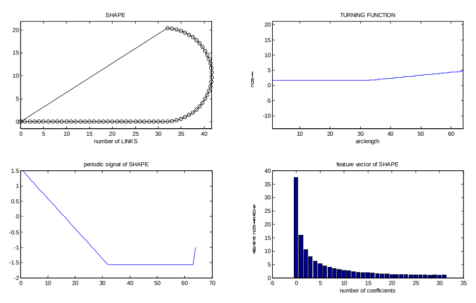
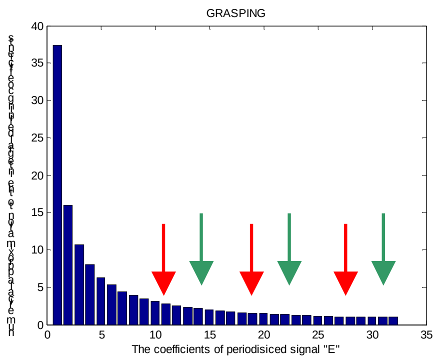
test = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32] Classified as **CIRCLE**	
	
60.9521 26.1909 13.3423 6.7743 3.9242 3.7453 4.1111 4.0701 3.6074 2.9467 2.3818 2.1524 2.2192 2.3314 2.3190 2.1568 1.9208 1.7380 1.6992 1.7712 1.8440 1.8385 1.7449 1.6148 1.5309 1.5428 1.6188 1.6829 1.6813 1.6114 1.5203 1.4780	

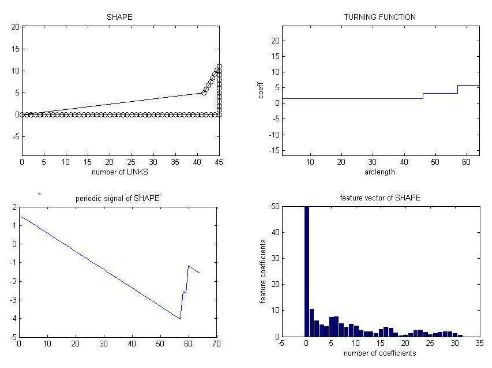
Fig. 5.15: Not-fully entwined circle

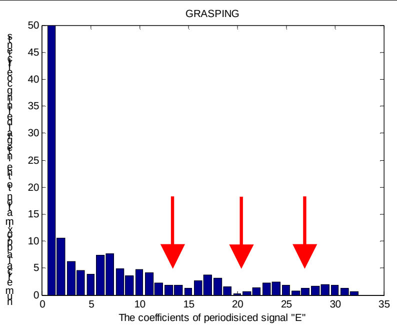
As mentioned above, the spectrum of not fully grasped circle has a wave form as the frequencies increase. The data arrays are 64-dimensional but the results have 32-dimensional data information. According nyquist theorem, the half of the coefficients are used in order to decrease the effects of the curse of dimensionality and thus the feature vector contains 32-dimensional data arrays. The absolute value of coefficient that is performed with mlp computes only the half of coefficients; both the static component and the coefficients above the nyquist frequency are removed. As mentioned in section 4.5, the classification of objects that are entwined several times is imperfectly if the nyquist requirements are not satisfied.

TRIANGLE

test = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 0 0 0 5*pi/6 0 0 0 0 0 0]

Classified as **CIRCLE**





49.9383 10.5842 6.1651 4.6303 3.9207 7.3695 7.6384 4.9253 3.6258 4.7307 4.1359 2.3049 1.8908 1.7776 1.2451 2.7137 3.6997 3.1282 1.5421 0.2643 0.6444 1.3206 2.2385 2.4849 1.7575 0.8338 1.2596 1.7468 1.9196 1.8384 1.2823 0.6420

Fig. 5.16: Not-fully entwined triangle

Note that object grasping information is related with the circumference of shapes. As depicted before, our manipulator is a shape definer and actually the manipulator is interpreted as an object. Grasping is to compute the circumference of related object. The circumference of all kind of objects to be either square, rectangle, triangle and circle is equal to $2.\pi$. Therefore, in case of whole-arm object grasping, the last arm of manipulator scans $360°$. Whole-arm grasping information of the object is the sum of the angles which gives the edge information of the object and circumference results must be equal to $2.\pi$.

If the product of the angles and the number of the entwined links is higher than $2.\pi$, the object is said to be more than one-time grasped.

$0 <$ the sum of angles $< 2.\pi$	not fully grasped
the sum of angles $= 2.\pi$	whole arm grasped
$2.\pi <$ the sum of angles $< 4.\pi$	more than one-time grasped
the sum of angles $= 4.\pi$	two times grasped
$4.\pi <$ the sum of angles $< 6.\pi$	more than two times grasped
the sum of angles $= 6.\pi$	three times grasped

5.2.1.2 One-time grasped objects

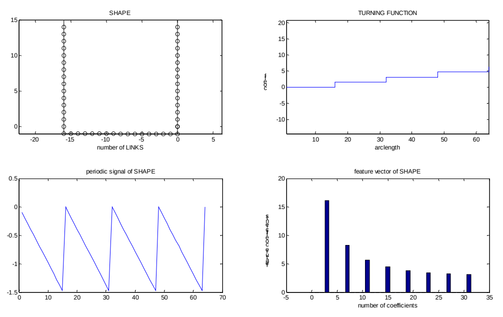
In following we will give the whole-arm grasped object results and at the end of this section, the relation of the coefficients (*coeff*) is discussed with an example. As mentioned previously, the periodisiced turning function (*E*) carry the important information for the achievement of grasping process. The edge-information is performed and observed in the periodic signal.

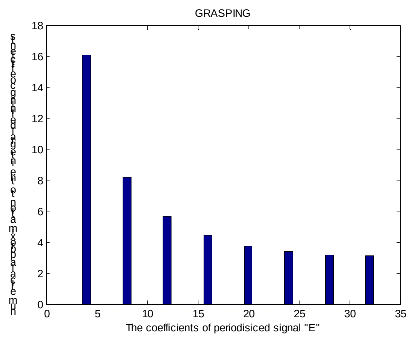
The spectrum of fourier coefficients in case of one-time grasped objects has important feature over not fully grasped objects. The periodic decreasing values of coefficients are repeated every certain values like 4.,8.,…28.,32. feature vector.

SQUARES

test = [pi/2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

Classified as **SQUARE**





0.0000 0.0000 0.0000 16.1033 0.0000 0.0000 0.0000 8.2094 0.0000 0.0000 0.0000 5.6547 0.0000 0.0000 0.0000
4.4429 0.0000 0.0000 0.0000 3.7784 0.0000 0.0000 0.0000 3.4004 0.0000 0.0000 0.0000 3.2031 0.0000
0.0000 0.0000 3.1416

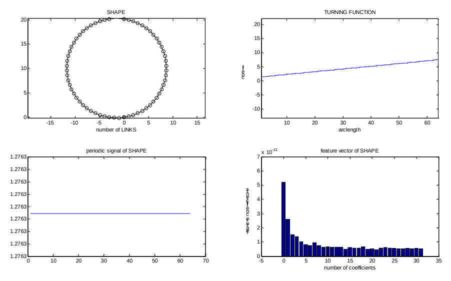
Fig. 5.17: The state of grasping process for whole-arm grasped object of square with coefficients.

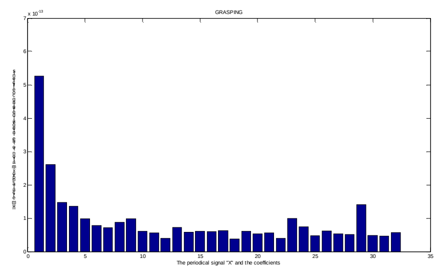
At the right side of the figure, the grasping results are shown. The whole arm grasped square has a periodic decreasing coefficient spectrum. The feature vector of shape reaches a steady state as the frequency increases. The spaces between coefficients are equally distributed and related with the periodic signal of shape.

CIRCLES

test = [pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32]

Classified as **CIRCLE**





0.5259 0.2603 0.1469 0.1357 0.0977 0.0778 0.0715 0.0872 0.0984 0.0607 0.0551 0.0396 0.0729 0.0578 0.0606 0.0587 0.0623 0.0373 0.0608 0.0535 0.0556 0.0395 0.0996 0.0735 0.0469 0.0612 0.0527 0.0502 0.1404 0.0480 0.0454 0.0570

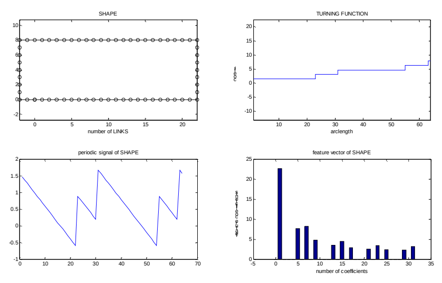
Fig. 5.18: The state of grasping process for whole-arm grasped object of circle.

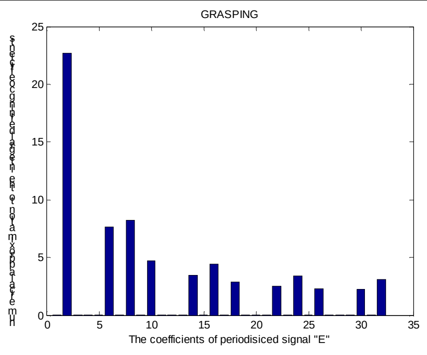
The periodic signal of whole-armed grasped circle is constant and related with the angles between the links of the manipulator.

RECTANGLES

test = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 pi/2]

Classified as **RECTANGLE**





0.0000 9.3040 0.0000 13.3894 0.0000 8.3659 0.0000 3.1416 0.0000 6.6323 0.0000 1.1032 0.0000 4.3674 0.0000 3.1416 0.0000 1.9158 0.0000 3.7058 0.0000 0.3492 0.0000 3.1416 0.0000 2.0827 0.0000 1.7796 0.0000 3.0209 0.0000 0.0000

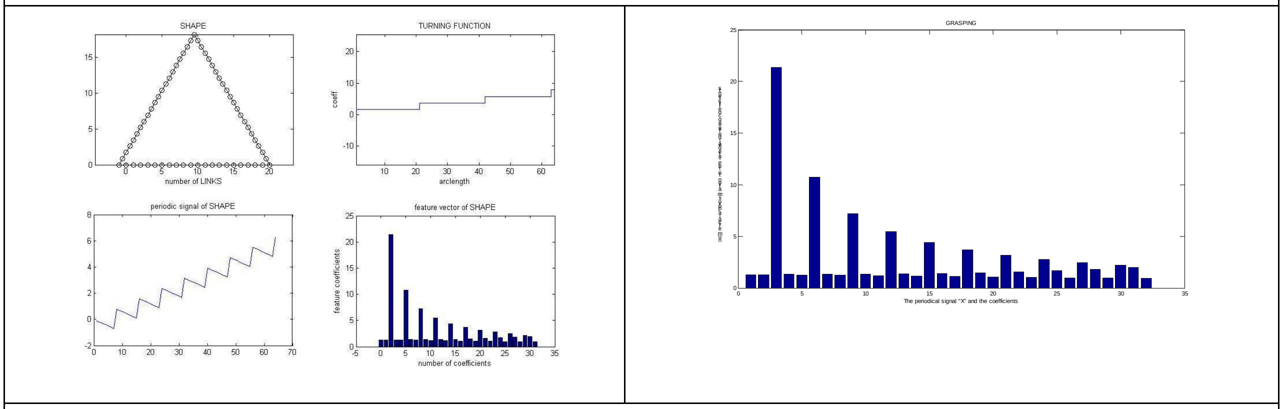
Fig. 5.19: The state of grasping process for whole-arm grasped object of rectangle.

As mentioned for squares, the whole arm grasped rectangle has a periodic decreasing coefficient spectrum. The spaces between coefficients are equally distributed and related with the periodic signal of shape.

TRIANGLES

test = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2*pi/3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2*pi/3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2*pi/3]

Classified as **TRIANGLE**



1.2995	1.2799	21.3745	1.3154	1.2546	10.7452	1.3304	1.2230	7.2196	1.3487	1.1853	5.4622	1.3749	1.1429	4.4051
1.4145	1.0978	3.6927	1.4729	1.0532	3.1739	1.5549	1.0125	2.7747	1.6647	0.9797	2.4559	1.8056	0.9583	2.1955
1.9811	0.9509													

Fig. 5.20: The state of grasping process for whole-arm grasped object of triangle with coefficients.

The results for triangles are the same with square shape. The coefficients decrease after increasing periodic frequencies. As see in the next section, if the number of grasping process increase, the periodic coefficient width wides. For one-time grasped triangle, every third coefficient has the highest value but for the more than one-time grasped shapes, the space between the periodicised highest frequencies increases.

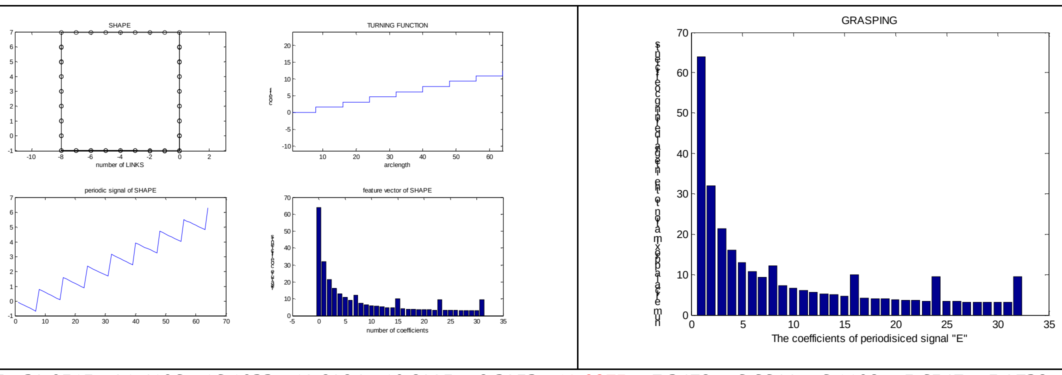
5.2.1.3 More than one-time grasped objects

In this section we will study the grasping results for objects that are more than one time entwined. If compared with other situations of grasping process, the spectrum of coefficients gives always a periodicised array and decreases to a constant value.

In the following, for each object, the results are illustriated and at the end of this section a brief summary of more than one-time grasping process is given.

SQUARES

test = [pi/2 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0]



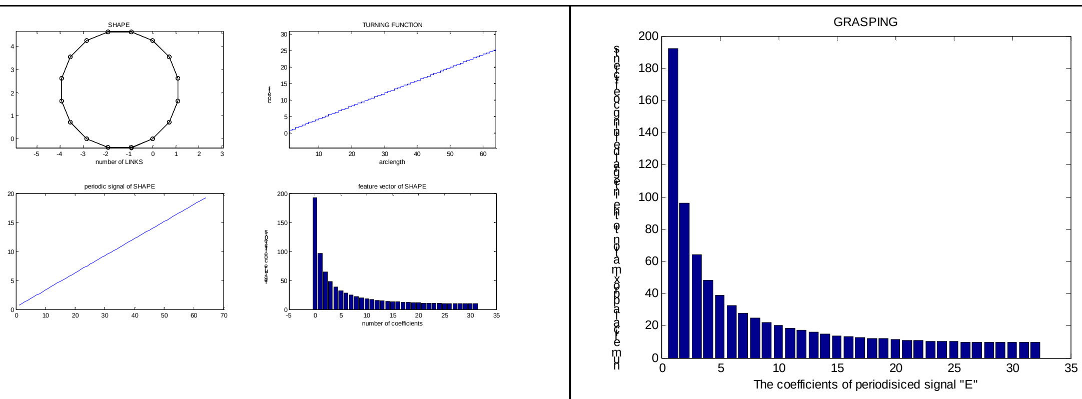
64.0257 32.0515 21.4106 16.1033 12.9294 10.8225 9.3253 12.0975 7.3478 6.6644 6.1108 5.6547 5.2738 4.9521
4.6781 9.9346 4.2399 4.0641 3.9113 3.7784 3.6627 3.5622 3.4753 9.5142 3.3366 3.2830 3.2387 3.2031
3.1760 3.1568 3.1454 9.4248

Fig. 5.21: More than one-time grasped square. The coefficient values decrease with the increasing number of frequencies.

The spectrum of fourier coefficients in case of more than one-time grasped objects become more linear and difference between the coefficients do not vary sharply as seen in the spectrum of one-time grasped object.

CIRCLES

test = [pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8]

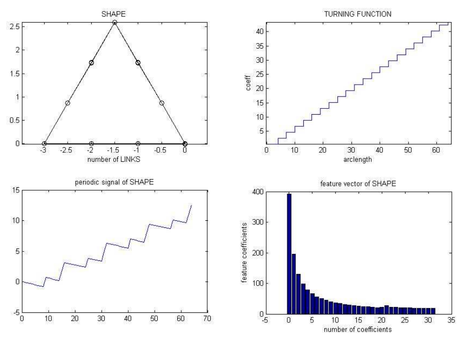


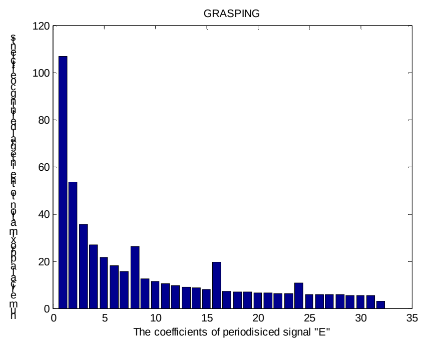
192.0771 96.1544 64.2319 48.3098 38.7883 32.4674 27.9758 24.6281 22.0434 19.9933 18.3325 16.9642 15.8214 14.8564
14.0342 13.3286 12.7198 12.1923 11.7339 11.3351 10.9881 10.6866 10.4258 10.2013 10.0099 9.8489 9.7160 9.6094
9.5279 9.4704 9.4361 9.4248

Fig. 5.22: More than one-time grasped circle.

TRIANGLES

test = [0 0 0 0 0 0 0 2*pi/3 0 0 0 0 0 0 0 2*pi/3 0 0 0 0 0 0 0 2*pi/3 0 0 0 0 0 0 0 2*pi/3 0 0 0 0 0 0 0 2*pi/3 0 0 0 0 0 0 0 2*pi/3 0 0 0 0 0 0 0 2*pi/3 0 0 0 0 0 0 0 2*pi/3]



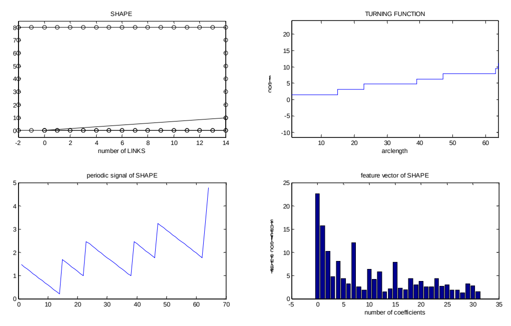


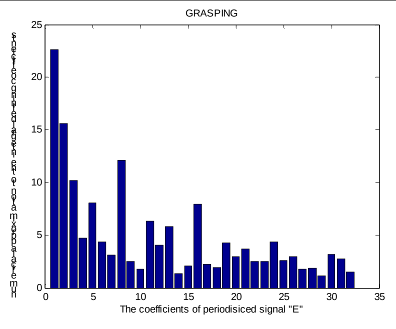
106.7237 53.4474 35.7267 26.8950 21.6190 18.1210 15.6391 26.0929 12.3691 11.2426 10.3320 9.5835 8.9599
8.4346 7.9882 19.4522 7.2773 6.9930 6.7467 6.5328 6.3472 6.1863 6.0473 10.7432 5.8264 5.7411
5.6708 5.6145 5.5714 5.5411 5.5230 2.8606

Fig. 5.23: More than one-time grasped triangle.

RECTANGLES

test = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2]





22.5923 15.6424 10.1999 4.7027 8.1120 4.3186 3.1308 12.0800 2.5020 1.8049 6.2906 4.0808 5.8281 1.3869 2.1188
7.9111 2.1955 1.9647 4.2726 2.9879 3.6887 2.5577 2.5150 4.3271 2.6052 2.9747 1.7992 1.8614 1.1657 3.1863
2.7497 1.4995

Fig. 5.24: More than one-time grasped rectangle.

If we summarize the grasping process, the signal that is transformed from the turning function of the shapes is necessary for the fourier transformation. With the help of MLP network, the cosines and the sinus values of plotted turning function, the real and the imaginary part of coefficients are computed using the two layers. The absolute value of summation of sin and cos part carries the key information for grasping process. Three different object features are introduced and spectrums of coefficients are performed. As a result, if the object is entwined the spectrum of fourier coefficients decrease and convergence to a constant value. If not, the spectrum gives a wave form of coefficients. In case of several entwinement process, the value of coefficients are bigger than the coefficients which are observed with one-time entwined objects.

5.2.2 Investigating the „Linear object scanner case“

In this section of the thesis, we will study out the possibility of the recognizing the deformed part of the objects passing through the conveyor belt using neural networks as a recognizer of patterns which is within the field known as quality control applications that are designed to find that one in a hundred or one in a thousand part that is defective. Linear object scanner case data inputs are $n \times n$ matrices. The main problem which is desired to be solved from a sensor system is that the sensor that lies on the conveyor-belt must recognize the faults on the shape. If there is a hole in the object or a part of is destructed, the object must be noticed out by the system.

As shown below, the construction of conveyor belt is seen. In this procedure, the objects must be recognized if they are deformed partially.

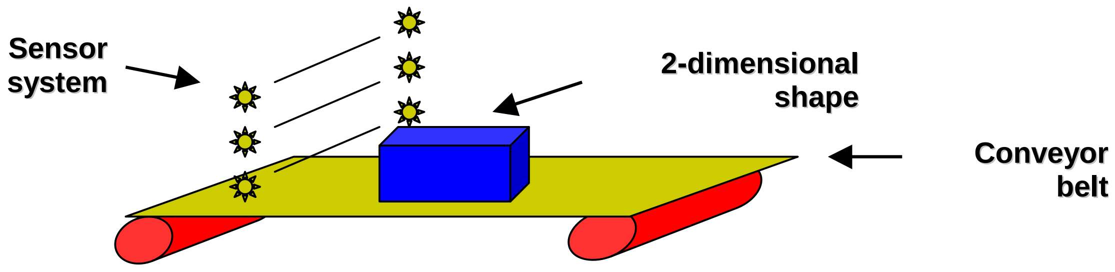


Figure 5.25: Illustration of „Linear object scanner case“ problem.

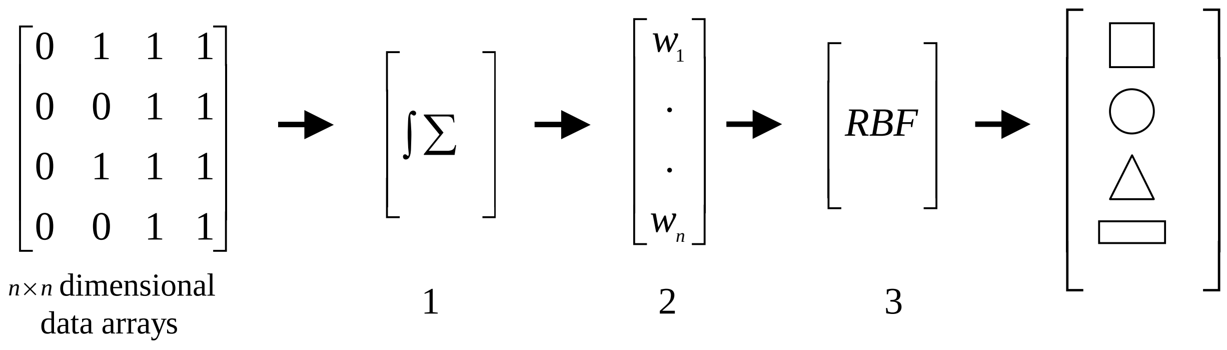

$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
$n \times n$ dimensional data arrays
$\int \Sigma$
1
w_1
w_n
2
RBF
3

Figure 5.26: The sub-problems of linear object scanner case problem.

square = [1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1]	circle = [0 0 1 0 0; 0 1 1 1 0; 1 1 1 1 1; 0 1 1 1 0; 0 0 1 0 0]

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

0	0	1	0	0
0	1	1	1	0
1	1	1	1	1
0	1	1	1	0
0	0	1	0	0

triangle = [0 0 0 0 0; 0 0 0 0 0; 0 0 1 0 0; 0 1 1 1 0; 1 1 1 1 1]	rectangle = [0 0 0 0 0; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1]

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	1	1	1	0
1	1	1	1	1

0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Figure 5.27: The $n \times n$ data arrays shown as inputs and as two dimensional shape information.

Fig. 5.26 shows the method that was used in linear object scanner case and fig. 5.27 reveals the inputs as $n \times n$ data arrays that give the object boundary information. “0” indicates the holes or the faults, “1” indicates the actual boundary points on the shape boundary that are given manually as a training set, the inputs, are said to be one dimensional-data arrays. Then, the fourier serie is computed for the classification. After implementation of fourier serie, the necessary coefficients are taken for the feature vector. The classification, which here pnn used, is given with the help of the coefficients of feature vector. As a result, the objects are classified as square, circle, triangle, rectangle.

The classification results in linear object scanner case problem can be explained as follows;

1. The objects which are partially deformed can be classified correctly. The partially deformed shapes which have different dimensions were recognized properly. The dimension of shapes that were recognized properly have smaller matrix dimension than $n \times n$ matrix.
2. The holes on the shapes cause problems in classification process and give false results.

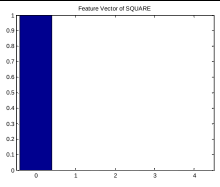
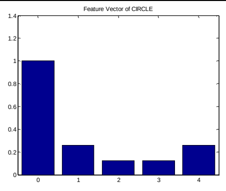
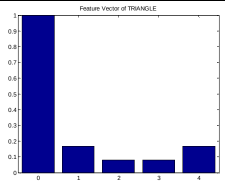
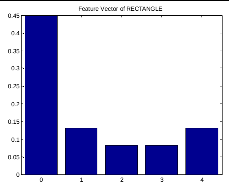
SHAPE																			
1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	1	1	1	0	0	0	0	0	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	1	1	1	1	1
1	1	1	1	1	0	1	1	1	0	0	1	1	1	0	1	1	1	1	1
1	1	1	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1
SQUARE					CIRCLE					TRIANGLE					RECTANGLE				
																			

Figure 5.28: The $n \times n$ data arrays with different object information. The feature vectors of two dimensional data arrays are shown.

As seen above, the objects have $n \times n$ dimensions that pass through the conveyor belt. These inputs are used as a training set and the classification is supervised according to these $n \times n$ data arrays.

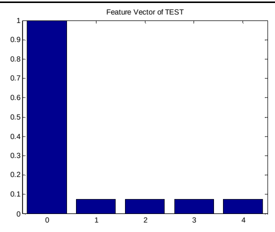
TEST SHAPE							
SHAPE					Feature vector	Comment	Result
1	1	1	1	1		Shape with a hole	Triangle
1	1	0	1	1			
1	1	0	1	1			
1	1	1	1	1			
1	1	1	1	1			

Figure 5.29: The $n \times n$ shape with a hole in the center.

The classification results of the $n \times n$ data arrays show that it is not possible to take correct classification results with the newpnn method for the defected objects passing through the conveyor belt.

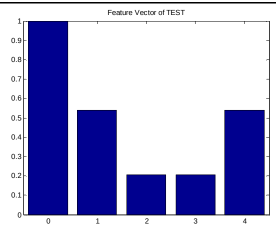
TEST SHAPE							
SHAPE					Feature vector	Comment	Result
0	0	0	0	0		Partially deformed shape	Square
0	0	0	0	0			
1	1	1	0	0			
1	1	1	0	0			
1	1	1	0	0			

Figure 5.30: The $n \times n$ data arrays with different object information.

We see that the classification procedure with deformed objects do not give the true results but on the other hand the small shapes are recognized and classified properly by our neural network. Fig 5.30 shows actually a small square comparing with the real shape value.

5.3 Conclusion

The fourier coefficients of turning functions that are performed in two mlp networks makes possible an interpretation about grasping of an object. The zeroth order of fourier transformation is the important key for grasping results.

The classification is realized with three networks, respectively two mlp networks and pnn. A probabilistic neural network structure is able to classify the objects with one dimensional data arrays but the results are not successful enough for the right classification if the object is translated, rotated or scaled. The reason of false results is the different side lengths of objects.

The two dimensional data arrays can not be classified with our pnn network correctly. The defaulted shapes that are produced during the production process must be recognized for the quality work of a firm. Thus, instead of pnn, the other neural networks must be used.

In this chapter, we have shown turning functions as a way to describe the shapes that are recognized during the grasping process. We developed two neural networks in order to make a classification of the objects that the feature vectors from the turning function are extracted. The shapes as test set which have different side lengths could not be classified correctly. The grasping process is introduced with the help of plotted object and coefficients performed in mlp.

In case of not fully entwined objects, the spectrum of coefficients gives a wavy form. For one-time and more than one-time entwined objects, the coefficients are observed as they decrease with the high frequencies. Also the value of coefficients increases if the objects are several times grasped.

6 Implementation in Matlab

This chapter describes implementation details with the help of MATLAB and the main results obtained with neural object classification. Section 6.1 starts with implementation details in neural object classification and continues to describe MATLAB realizations of related neural networks based algorithm.

6.1 Implementation in MATLAB

The MATLAB codes was written and tested with MATLAB Version 7.0. The software package was used with Windows XP on a Pentium Celeron M processor with 1.5 Ghz clock frequency and 996 MB RAM.

MATLAB code listings

6.1.1 Tentacle case

```
%
% tentaclecase.m
%
% Classification with RBF and grasping with MLP
%
% input:
%   SQUARE
%   CIRCLE
%   TRIANGLE
%   RECTANGLE
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The function uses inputs as a training set and outputs the grasping process for each object. The manipulator has 16 number of links and is used as a shape definer. Therefore one-dimensional data array has 16 values which describe the angles between the links.

```
[obj,n,t,ts]=DRAWTESTSET()
```

One-dimensional data arrays are converted into two-dimensional data arrays and plotted

```
[t,coefficient,x,s]=turningfunction(obj,n)
```

Performing the turning function of plotted object

```
[a,E]=periodic(t)
```

Transforming the turning function into periodical signal “E”

```
subplot(2,2,1);
plot(x(1,:),x(2,:),'ko-');
axis equal;
title('SHAPE');
Xlabel('number of LINKS');
```

Plotting the shape of one-dimensional data arrays

```
subplot(2,2,2);
stairs(s(1:n-1),t);
title('TURNING FUNCTION');
axis equal;
Xlabel('arclength');
Ylabel('coeff');
```

Plotting the turning function in respect with arclength

```
subplot(2,2,3);
plot(E);
title('periodic signal of SHAPE')
```

Plotting the periodical signal of shape

```
subplot(2,2,4);
bar([0:((n/2)-1)],a);
title ('feature vector of SHAPE');
Xlabel('number of coefficients');
Ylabel('feature coefficients');
```

Plotting the feature vector of periodical signal over nyquist theorem

```
[A,fig] = transformationX(E,fig)
```

Implementing the periodisiced signal in mlp network

```
[circle,triangle,rectangle,square] = classify()
```

Input sets of each object for the classification

```
P=[];
y = sim(MLP, {circle'});
coeff = abs(y{1}+i*y{2});
P = [P, coeff];

y = sim(MLP, {triangle'});
coeff = abs(y{1}+i*y{2});
P = [P, coeff];

y = sim(MLP, {square'});
coeff = abs(y{1}+i*y{2});
P = [P, coeff];

y = sim(MLP, {rectangle'});
coeff = abs(y{1}+i*y{2});
P = [P, coeff];
```

The periodical signal and coefficients of each class of object performed in two mlp networks respectively

```
T = ind2vec([1 2 3 4]);
```

The classes of objects are performed

```
net = newpnn(P,T);
```

Pnn, A kind of radial basis network is created

```
y = sim(MLP, {E'});
coeff = (abs(y{1}+i*y{2}))';
```

Test object is given to mlp networks in order to make classification

```
fig = fig + 1;
figure(fig)
bar(A);
title ('GRASPING PROCESS')
Xlabel('The periodical signal "E" and the coefficients');
Ylabel('numerical approximation to the integral defining coefficients');
```

The grasping results performed in mlp networks are plotted. Here the turning function of shapes and coefficients are figured

```
class = sim(net, coeff');
vec2ind(class)
yc_test = vec2ind(class)
```

The class indices are converted into target vectors and a Pnn network is designed and tested

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
switch yc_test
case 1, display('SQUARE');

case 2, display('CIRCLE');

case 3, display('TRIANGLE');

case 4, display('RECTANGLE');
end
```

This small program is written in C program language to see the classification results with words instead of the numbers. A little C program outputs the classification results.
Note, that the PNN will always output circle if there is no match.

6.1.2 Linear object scanner case problem

```
% twodimensional.m
%
% Classification with RBF
% input:
%   SQUARE
%   CIRCLE
%   TRIANGLE
%   RECTANGLE
% output:

clear
fig = 0;
n = 5;
```

Two-dimensional data arrays were used describing the objects as input.
The dimension of matrix is $5x5$ and also as test set the matrixes which have small dimensions were used.

```
% square
square1 = [1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1]';
square2 = [1 1 1 1 0; 1 1 1 1 0; 1 1 1 1 0; 1 1 1 1 0; 1 1 1 1 1]';
square3 = [0 0 0 0 0; 0 1 1 1 0; 0 1 1 1 0; 0 1 1 1 0; 0 0 0 0 0]';
square4 = [0 0 0 0 0; 0 0 1 1 0; 0 0 1 1 0; 0 0 0 0 0; 0 0 0 0 0]';

%%%%%%%%%%%%%%%%
% circle
circle1 = [0 0 1 0 0; 0 1 1 1 0; 1 1 1 1 1; 0 1 1 1 0; 0 0 1 0 0 ]';
circle2 = [0 0 0 0 0; 0 0 1 0 0; 0 1 1 1 0; 0 0 1 0 0; 0 0 0 0 0 ]';
circle3 = [0 1 1 1 0; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 0 1 1 1 0 ]';
circle4 = [0 0 0 0 0; 0 0 0 0 0; 0 0 1 0 0; 0 0 0 0 0; 0 0 0 0 0 ]';

% triangle
triangle1 = [0 0 0 0 0; 0 0 0 0 0; 0 0 1 0 0; 0 1 1 1 0; 1 1 1 1 1]';
triangle2 = [1 1 1 1 1; 0 1 1 1 0; 0 0 1 0 0; 0 0 0 0 0; 0 0 0 0 0]';
triangle3 = [1 0 0 0 0; 1 1 0 0 0; 1 1 1 0 0; 1 1 0 0 0; 1 0 0 0 0]';
triangle4 = [0 0 0 0 0; 0 0 0 0 0; 0 0 0 0 0; 0 0 1 0 0; 0 1 1 1 0]';

% rectangle
rectangle1 = [0 0 0 0 0; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1]';
rectangle2 = [1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 0 0 0 0 0]';
rectangle3 = [1 1 1 1 0; 1 1 1 1 0; 1 1 1 1 0; 1 1 1 1 0; 1 1 1 1 0]';
rectangle4 = [0 1 1 0 0; 0 1 1 0 0; 0 1 1 0 0; 0 1 1 0 0; 0 1 1 0 0]';
```

Four type of objects that have different bigness are used.

```
% FFT
% computes the fast fourier transform of the data arrays
% classification
% This function results the FFT of objects for classification
[net,Y,Yc,T,P,norm_abs_FFT_alpha_square_1,
norm_abs_FFT_alpha_square_2,norm_abs_FFT_alpha_square_3
,norm_abs_FFT_alpha_square_4,norm_abs_FFT_alpha_circle_1,
norm_abs_FFT_alpha_circle_2,norm_abs_FFT_alpha_circle_3
,norm_abs_FFT_alpha_circle_4,norm_abs_FFT_alpha_triangle_1,
norm_abs_FFT_alpha_triangle_2,norm_abs_FFT_alpha_triangle_3
,norm_abs_FFT_alpha_triangle_4,norm_abs_FFT_alpha_rectangle_1
,norm_abs_FFT_alpha_rectangle_2,norm_abs_FFT_alpha_rectangle_3
,norm_abs_FFT_alpha_rectangle_4] =
FFTscanner(square1,square2,square3,square4,circle1,circle2,circle3
,circle4,triangle1,triangle2,triangle3,triangle4,rectangle1,rectangle2
,rectangle3,rectangle4)
```

Discrete Fourier Transformation of the objects are computed with this function.
Classification of the input vectors is computed with the help of this neural network function.
Indices are converted into vectors to create the number of classes.
Here there are 4 type of classes.

A neural network is created for the classification.
The input arrays are tested with the help of neural network in order to give out the results of the classification.

```
fig = fig + 1;
figure(fig)
bar([0:n-1],norm_abs_FFT_alpha_square_1);
title('Feature Vector of SQUARE');

fig = fig + 1;
figure(fig)
bar([0:n-1],norm_abs_FFT_alpha_circle_1);
title('Feature Vector of CIRCLE');

fig = fig + 1;
figure(fig)
bar([0:n-1],norm_abs_FFT_alpha_triangle_1);
title('Feature Vector of TRIANGLE');

fig = fig + 1;
figure(fig)
bar([0:n-1],norm_abs_FFT_alpha_rectangle_1);
title('Feature Vector of RECTANGLE ');
```

```
% classifying a new vector with the network

%test = [0 0 0 0 0; 1 1 0 1 1; 1 1 0 1 1; 1 1 1 1 1; 1 1 1 1 1]';
%test = [1 1 0 0 0; 1 1 0 0 0; 1 1 0 0 0; 1 1 0 0 0; 1 1 1 1 1]';
%test = [1 1 1 1 1; 1 1 0 1 1; 1 1 0 1 1; 1 1 1 1 1; 1 1 1 1 1]';
%test = [1 1 1 1 1; 1 1 0 0 1; 1 1 0 0 1; 1 1 1 1 1; 1 1 1 1 1]';
%test = [0 0 0 0 0; 0 0 1 1 1; 0 0 1 1 1; 0 0 1 1 1; 0 0 0 0 0]';
%test = [1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 0 0 0 0 0]';
%test = [1 1 1 1 1; 1 1 0 1 1; 1 1 0 1 1; 1 1 1 1 1; 1 1 1 1 1]';
%test = [1 1 1 1 1; 1 1 0 1 1; 1 1 0 1 1; 1 1 1 1 1; 1 1 1 1 1]';
```

The objects being defaulted are used as test set in order to achieve the classification process.

```
% The classification of TEST set with the help of RBF.
[yc_test]= TESTFFT(test,net)
```

The classification of TEST set with the help of RBF.

```
switch yc_test
case 1, display('SQUARE');

case 2, display('CIRCLE');

case 3, display('TRIANGLE');

case 4, display('RECTANGLE');
end
```

This small program is written in C program language to see the classification results with words instead of the numbers. A little C program outputs the classification results.
Note, that the PNN will always output circle if there is no match.

6.2 Functions

The functions which are briefly described below, include the following main file and first level subfunctions, respectively

TURNINGFUNCTION.m	Computes turning function of plotted shape
PERIODIC.m	Returns turning function to periodic signal
NEWPNN.m	Computes the classification process
MLP.m	Returns the fourier transformation of periodic signals
TESTFFT.m	Returns the fourier transformation of test objects
CLASSIFY.m	Classifies each object
FFTscanner.m	Returns discrete fourier transformation of two-dimensional objects

---

TURNINGFUNCTION.m Computes turning function of plotted shape

```
function [t,coefficient,x,s] = turningfunction(obj,n)
%GRAPHS : plots of turning function and frequency contents
%  obj : object boundary as spline in ppform
%  n   : number of points on object contour
% gather object pnts
%equally spaced
s = linspace(obj.breaks(1),obj.breaks(end),n);

%equally spaced, partially occluded:
%s = linspace(obj.breaks(end)*0.2,obj.breaks(end)*0.8,n);

x = fnval(obj, s);
% compute turning function
% df = fnval(fnder(obj),s);
%
% %first tangent's angel relative to y = const:
% %absolute angels in [-pi,pi]
% t = atan2(df(2,:), df(1,:));
% %correct for pi -jumps
% if (abs(t(1) - t(n)) >= pi)
%   t(1) = t(1) + 2*pi;
% end;
% for i = 2 : n
%   %this condition certainly is not quite correct - but it works as long as the differnces of
%   %tangent angels are small => arclengths between pnts must be small
%   if (abs(t(i) - t(i-1)) >= pi)
%     t(i) = t(i) + 2*pi;
%   end;
% end;
vec = [[0;1],diff(x,1,2)]; %diff does not account for the last vector that closes the boundary!
len = [sqrt(sum(vec.^2,1))];
for i = 1 : n - 1
  cosphi = vec(:,i)' * vec(:,i+1) / (len(1,i) * len(1,i+1));
  phi(i) = acos(cosphi);
```

```
end;

t = cumsum(phi);
%---------------------------------------------------------------------
% compute frequency contents
%---------------------------------------------------------------------
coefficient = abs(fft(t));
%---------------------------------------------------------------------
% plots
%---------------------------------------------------------------------
return
```

PERIODIC.m		Returns turning function to periodic signal

```
function [a,E]=periodic(t)
%FEATURE extract feature vector from turning funciton for classification
%
%requests:
%  t : turning function of shape
%returns:
%  a  : feature vector
%transform turning function to periodic signal:
n = length(t);
E = t - 2*pi/n * [1:n];
% where n is the number of links.
%apply fourier transformation and strip contents over nyquist frequ. and
%static content
% compute frequency contents
fta = abs(fft(E));
ft = fta(1,2:ceil(n/2)+1);

%return feature vector:
a = ft;
return
```

GRASPING.m		computes actual transformation

```
function [A,coeff,fig] = grasping(E,fig)

%MLP : the actual transformation
% E is the periodisiced signal of turning function of shapes.
% "coeff" results the absolute value of the real and the imaginary part of
% fft values of the shape that are computed distinctly in layer {1} and
% layer {2}.(see function MLP)
%the coefficients are determined with
A = [];
y = sim(MLP, {E'});
coeff = (abs(y{1}+i*y{2}))';
% A outputs the result
A = [coeff];

%The colors are set by the colormap.
```

```
fig = fig + 1;
figure(fig)
bar(A);
title ('GRASPING')
Xlabel('The coefficients of periodisiced signal "E"');
Ylabel('numerical approximation to the integral defining coefficients');

return
```

---

newpnn.m Computes the classification process

```
function [net] = newpnn(p,t,spread)
%NEWPNN Design a probabilistic neural network.
%
%  Synopsis
%
%    net = newpnn
%    net = newpnn(P,T,SPREAD)
%
%  Description
%
%    Probabilistic neural networks are a kind of radial
%    basis network suitable for classification problems.
%
%   NET = NEWPNN creates a new network with a dialog box.
%
%    NET = NEWPNN(P,T,SPREAD) takes two or three arguments,
%      P      - RxQ matrix of Q input vectors.
%      T      - SxQ matrix of Q target class vectors.
%      SPREAD - Spread of radial basis functions, default = 0.1.
%    and returns a new probabilistic neural network.
%
%    If SPREAD is near zero the network will act as a nearest
%    neighbor classifier.  As SPREAD becomes larger the designed
%    network will take into account several nearby design vectors.
%
%  Examples
%
%    Here a classification problem is defined with a set of
%    inputs P and class indices Tc.
%
%      P = [1 2 3 4 5 6 7];
%      Tc = [1 2 3 2 2 3 1];
%
%    Here the class indices are converted to target vectors,
%    and a PNN is designed and tested.
%
%      T = ind2vec(Tc)
%      net = newpnn(P,T);
%      Y = sim(net,P)
%      Yc = vec2ind(Y)
%
%  Algorithm
%
%    NEWPNN creates a two layer network. The first layer has RADBAS
%    RADBAS neurons, and calculates its weighted inputs with DIST, and
%    its net input with NETPROD.  The second layer has COMPET neurons,
%    and calculates its weighted input with DOTPROD and its net inputs
%    with NETSUM. Only the first layer has biases.
%
%    NEWPNN sets the first layer weights to P', and the first
%    layer biases are all set to 0.8326/SPREAD resulting in
%    radial basis functions that cross 0.5 at weighted inputs
%    of +/- SPREAD. The second layer weights W2 are set to T.
%
%  References
%
%    P.D. Wasserman, Advanced Methods in Neural Computing, New York:
%       Van Nostrand Reinhold, pp. 35-55, 1993.
%
%  See also SIM, IND2VEC, VEC2IND, NEWRB, NEWRBE, NEWGRNN.

% Mark Beale, 11-31-97
% Copyright 1992-2002 The MathWorks, Inc.
% $Revision: 1.9 $ $Date: 2002/03/25 16:53:29 $

if nargin < 2
  net = newnet('newpnn');
  return
end
```

```
% Defaults
if nargin < 3, spread = 0.1; end

% Error checks
if (~isa(p,'double') & ~islogical(p)) | (~isreal(p)) | (length(p) == 0)
  error('Inputs are not a non-empty real matrix.')
end
if (~isa(t,'double') & ~islogical(t)) | (~isreal(t)) | (length(t) == 0)
  error('Targets are not a non-empty real matrix.')
end
if (size(p,2) ~= size(t,2))
  error('Inputs and Targets have different numbers of columns.')
end
if (~isa(spread,'double')) | ~isreal(spread) | any(size(spread) ~= 1) | (spread < 0)
  error('Spread is not a positive or zero real value.')
end

% Dimensions
[R,Q] = size(p);
[S,Q] = size(t);

% Architecture
net = network(1,2,[1;0],[1;0],[0 0;1 0],[0 1]);

% Simulation
net.inputs{1}.size = R;
net.inputWeights{1,1}.weightFcn = 'dist';
net.layers{1}.netInputFcn = 'netprod';
net.layers{1}.transferFcn = 'radbas';
net.layers{1}.size = Q;
net.layers{2}.size = S;
net.layers{2}.transferFcn = 'compet';

% Weight and Bias Values
net.b{1} = zeros(Q,1)+sqrt(-log(.5))/spread;
net.iw{1,1} = p';
net.lw{2,1} = t;

return
```

```
function [net] = setWeightMatrix(net,W)
  inputSizes = net.hint.inputSizes;
  layerSizes = net.hint.layerSizes;
  I = net.hint.totalInputSize;
  U = net.hint.totalLayerSize;

  % W(i,j) weights to layer unit i from input, layer or bias unit j

  for i=1:net.numLayers
    indRow = sum(layerSizes(1:i-1)) + (1 : layerSizes(i));

    for j=find(net.inputConnect(i,:))
      indCol = sum(inputSizes(1:j-1)) + (1 : inputSizes(j));
      net.IW{i,j} = W(indRow, indCol);
    end

    if net.biasConnect(i)
      indCol = I + 1;
      net.b{i} = W(indRow, indCol);
    end

    for j=find(net.layerConnect(i,:))
      indCol = I + 1 + sum(layerSizes(1:j-1)) + (1 : layerSizes(j));
      net.LW{i,j} = W(indRow, indCol);
    end
  end

%function [net_new] = trainrtrl(net,P,T,epochs,show,lr)
% trainrtrl - encapsules the RTRL training algorithm in rtrl.mex.
%
%function [net] = trainrtrl(net,P,T,epochs,show,lr)
%
% requests
%    net       : neural network structure (Neural Network toolbox)
%    P         : input signal (sequence of T timesteps)
%    T         : teacher signal (sequence of T timesteps)
%    epochs    : number of epochs
%    show      : number of epochs, after which a report of the training progress is output
%                at the command line
%    lr        : learning rate
%
```

```
% returns
%     net_new    : neural network structure after training
%
% remarks
% This function converts the neural network structure to the form the mex file demands, calls the
% mex file and reconverts the outcome of the training to a neural network structure. In addition,
% it plots the training signal and the network output at the end of the training.

net=network;


epochs = 50;
lr = 0,6;
%prepare weight matrix:
 % w = getWeightMatrix(net);



net.trainParam.epochs = 50;
net.trainParam.goal = 0.01;
net = train(net,p,t);
y2 = sim(net,p)
plot(p,t,'o',p,y1,'x',p,y2,'*')
%-----------------------------------------------------------------------------------------------
% utility fcns - conversion of neural network structure to and from the form the mex file
% demands
%-----------------------------------------------------------------------------------------------

%function [W] = getWeightMatrix(net)
  inputSizes = net.hint.inputSizes;
  layerSizes = net.hint.layerSizes;
  I = net.hint.totalInputSize;
  U = net.hint.totalLayerSize;

  % W(i,j) weights to layer unit i from input, layer or bias unit j

  W = zeros(U, U+I+1);

  for i=1:net.numLayers
    indRow = sum(layerSizes(1:i-1)) + (1 : layerSizes(i));

    for j=find(net.inputConnect(i,:))
      indCol = sum(inputSizes(1:j-1)) + (1 : inputSizes(j));
      W(indRow, indCol) = net.IW{i,j};
    end

    if net.biasConnect(i)
      indCol = I + 1;
      W(indRow, indCol) = net.b{i};
    end

    for j=find(net.layerConnect(i,:))
      indCol = I + 1 + sum(layerSizes(1:j-1)) + (1 : layerSizes(j));
      W(indRow, indCol) = net.LW{i,j};
    end
  end

%prepare array with transfer fcn indices:
transferFcn = zeros(net.hint.totalLayerSize,1);
j = 0;
for i = 1:net.numLayers
  n = net.layers{i}.size;
  switch (net.hint.transferFcn{i})
      case 'logsig'
        transferFcn(j+1:j+n,1) = 0*ones(n,1);
      case 'tansig'
        transferFcn(j+1:j+n,1) = 1*ones(n,1);
      case 'purelin'
        transferFcn(j+1:j+n,1) = 2*ones(n,1);
      otherwise
        transferFcn(j+1:j+n,1) = 0*ones(n,1);
  end;
  j = j+n;
end;

%target indices
targInd = [];
for i = net.hint.targetInd
  targInd = [targInd sum(net.hint.layerSizes(1:i-1)) + (1 : net.hint.layerSizes(i))];
end;

%pass inputs to mex file
w = rtrl(w, transferFcn, P, T, targInd, epochs, show, lr);
```

```
%update net
net_new = net;
net_new = setWeightMatrix(net_new,w);

%simulation of net (initial state set to first sample of teacher signal)
%y = sim(net, con2seq(P), {}, {T(:,1)});
y = sim(net, con2seq(repmat(P,1,4)), {}, {T(:,1)});
y = seq2con(y);
y = y{:};



%time - sequence
figure;
plot([T(:,1) y]','k-');
hold on;
plot(repmat(T,1,4)','k--');
hold off;

%phase plane
figure;
plot([T(1,1),y(1,:)],[T(2,1),y(2,:)],'k-');
hold on;
plot([T(1,:),T(1,1)],[T(2,:),T(2,1)],'k--');
hold off;

return
```

MLP.m   Returns the fourier transformation of periodic signals

```
function [net] = MLP(N);
%dftnet - set up a MLP network that performs discrete fourier transformation
%
%requires:
%  N  : number of samples
%returns:
%  net : Neural Network Toolbox network structure.
%remarks:
%MLP computes only ceil(N/2) coefficients; both the static component and the
%coefficients above the nyquist frequency are stripped. The two layers output
%the real and imaginary parts of the coefficients.
%the weight matrix is determined analytically; no need to perform additional
%training on the net.
N=16;
net = network;
net.numInputs = 1;
net.numLayers = 2;
net.inputs{1}.size = N;

net.biasConnect = zeros(2,1);
net.outputConnect = ones(1,2);

%DFT layer (outputs complex fourier coefficients)
net.layers{1}.size = ceil(N/2);
net.layers{2}.size = ceil(N/2);

net.layers{1}.transferFcn = 'purelin';
net.layers{2}.transferFcn = 'purelin';

net.inputConnect = ones(2,1);

for i = 1 : ceil(N/2)
  for j = 1 : N
    kn(i,j) = i*(j-1);
  end;
end;

W1 = cos( 2*pi/N * kn );
W2 = - sin( 2*pi/N * kn );

net.IW{1,1} = W1;
net.IW{2,1} = W2;

net.inputWeights{1,1}.learn = 0;
net.inputWeights{2,1}.learn = 0;

return
```

CLASSIFY.m classifies each object

```
function [circle,triangle,rectangle,square] = classify()

%%%%%%%%%%%%%%%%
%Shapes in the training set
% square
%square1 = [0 0 0 pi/2 0 0 0 pi/2 0 0 0 pi/2 0 0 0 pi/2]';
square1 = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2]';

%%%%%%%%%%%%%%%%
% circle
%circle1 = [pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8 pi/8]';
circle1 = [pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32
pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32
pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32
pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32 pi/32]';
%%%%%%%%%%%%%%%%
% triangle
triangle1 = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2*pi/3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2*pi/3 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2*pi/3]';

%%%%%%%%%%%%%%%%
% rectangle
```

```
rectangle1 = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 pi/2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 pi/2 0 0 0 0 0 0 0 pi/2]';

test2= [];
L=[0;0];

n = length(circle1);

test2(1)= circle1(1);

for i = 2:n
   test2(i) = test2(i - 1) + circle1(i);
end
L(1,1) = 0;
L(2,1) = 0;
L(1,n+1) = 0;
L(2,n+1) = 0;
for i = 2 : n
            L(1,i)= cos(test2(i)) + L(1,i-1);
            L(2,i)= sin(test2(i)) + L(2,i-1);
end

n = length(L);
% Interpolate with a spline curve and finer spacing.
t = 1:n;
ts = 1: 0.1: n;
obj = csapi(t, L);


%function [t,coefficient,x,s] = turningfunction(obj,n)
%GRAPHS : plots of turning function and frequency contents
%  obj : object boundary as spline in ppform
%  n   : number of points on object contour
% gather object pnts
%equally spaced
s = linspace(obj.breaks(1),obj.breaks(end),n);

%equally spaced, partially occluded:
%s = linspace(obj.breaks(end)*0.2,obj.breaks(end)*0.8,n);

x = fnval(obj, s);
% compute turning function
% df = fnval(fnder(obj),s);
%
% %first tangent's angel relative to y = const:
% %absolute angels in [-pi,pi]
% t = atan2(df(2,:), df(1,:));
% %correct for pi -jumps
% if (abs(t(1) - t(n)) >= pi)
%   t(1) = t(1) + 2*pi;
% end;
% for i = 2 : n
%   %this condition certainly is not quite correct - but it works as long as the differnces of
%   %tangent angels are small => arclengths between pnts must be small
%   if (abs(t(i) - t(i-1)) >= pi)
%     t(i) = t(i) + 2*pi;
%   end;
% end;
vec = [[0;1],diff(x,1,2)]; %diff does not account for the last vector that closes the boundary!
len = [sqrt(sum(vec.^2,1))];
for i = 1 : n - 1
  cosphi = vec(:,i)' * vec(:,i+1) / (len(1,i) * len(1,i+1));
  phi(i) = acos(cosphi);
end;

t = cumsum(phi);
%--------------------------------------------------------------------
%function [a,E]=periodic(t)
%FEATURE extract feature vector from turning funciton for classification
%
%requests:
%  t : turning function of shape
%returns:
%  a  : feature vector
%transform turning function to periodic signal:
n = length(t);
circle = t - 2*pi/n * [1:n];


%function [triangle] = triangle1()

test2= [];
L=[0;0];

n = length(triangle1);

test2(1)= triangle1(1);

for i = 2:n
    test2(i) = test2(i - 1) + triangle1(i);
end
L(1,1) = 0;
L(2,1) = 0;
L(1,n+1) = 0;
L(2,n+1) = 0;
for i = 2 : n
            L(1,i)= cos(test2(i)) + L(1,i-1);
            L(2,i)= sin(test2(i)) + L(2,i-1);
end

n = length(L);
% Interpolate with a spline curve and finer spacing.
t = 1:n;
ts = 1: 0.1: n;
obj = csapi(t, L);


%function [t,coefficient,x,s] = turningfunction(obj,n)
%GRAPHS : plots of turning function and frequency contents
%  obj : object boundary as spline in ppform
%  n   : number of points on object contour
% gather object pnts
%equally spaced
s = linspace(obj.breaks(1),obj.breaks(end),n);

%equally spaced, partially occluded:
%s = linspace(obj.breaks(end)*0.2,obj.breaks(end)*0.8,n);

x = fnval(obj, s);
% compute turning function
% df = fnval(fnder(obj),s);
%
% %first tangent's angel relative to y = const:
% %absolute angels in [-pi,pi]
% t = atan2(df(2,:), df(1,:));
% %correct for pi -jumps
% if (abs(t(1) - t(n)) >= pi)
%   t(1) = t(1) + 2*pi;
% end;
% for i = 2 : n
%   %this condition certainly is not quite correct - but it works as long as the differnces of
%   %tangent angels are small => arclengths between pnts must be small
%   if (abs(t(i) - t(i-1)) >= pi)
%     t(i) = t(i) + 2*pi;
%   end;
% end;
vec = [[0;1],diff(x,1,2)]; %diff does not account for the last vector that closes the boundary!
len = [sqrt(sum(vec.^2,1))];
for i = 1 : n - 1
  cosphi = vec(:,i)' * vec(:,i+1) / (len(1,i) * len(1,i+1));
  phi(i) = acos(cosphi);
end;

t = cumsum(phi);
%----------------------------------------------------------------------
%function [a,E]=periodic(t)
%FEATURE extract feature vector from turning funciton for classification
%
%requests:
%  t : turning function of shape
%returns:
%  a  : feature vector
%transform turning function to periodic signal:
n = length(t);
triangle = t - 2*pi/n * [1:n];

%function [square] = square1()

test2= [];
L=[0;0];

n = length(square1);

test2(1)= square1(1);

for i = 2:n
    test2(i) = test2(i - 1) + square1(i);
end
L(1,1) = 0;
L(2,1) = 0;
L(1,n+1) = 0;
L(2,n+1) = 0;
for i = 2 : n
            L(1,i)= cos(test2(i)) + L(1,i-1);
            L(2,i)= sin(test2(i)) + L(2,i-1);
end

n = length(L);
% Interpolate with a spline curve and finer spacing.
t = 1:n;
ts = 1: 0.1: n;
obj = csapi(t, L);


%function [t,coefficient,x,s] = turningfunction(obj,n)
%GRAPHS : plots of turning function and frequency contents
```

```
%  obj : object boundary as spline in ppform
%  n   : number of points on object contour
% gather object pnts
%equally spaced
s = linspace(obj.breaks(1),obj.breaks(end),n);

%equally spaced, partially occluded:
%s = linspace(obj.breaks(end)*0.2,obj.breaks(end)*0.8,n);

x = fnval(obj, s);
% compute turning function
% df = fnval(fnder(obj),s);
%
% %first tangent's angel relative to y = const:
% %absolute angels in [-pi,pi]
% t = atan2(df(2,:), df(1,:));
% %correct for pi -jumps
% if (abs(t(1) - t(n)) >= pi)
%   t(1) = t(1) + 2*pi;
% end;
% for i = 2 : n
%   %this condition certainly is not quite correct - but it works as long as the differnces of
%   %tangent angels are small => arclengths between pnts must be small
%   if (abs(t(i) - t(i-1)) >= pi)
%     t(i) = t(i) + 2*pi;
%   end;
% end;
vec = [[0;1],diff(x,1,2)]; %diff does not account for the last vector that closes the boundary!
len = [sqrt(sum(vec.^2,1))];
for i = 1 : n - 1
  cosphi = vec(:,i)' * vec(:,i+1) / (len(1,i) * len(1,i+1));
  phi(i) = acos(cosphi);
end;

t = cumsum(phi);
%----------------------------------------------------------------------
%function [a,E]=periodic(t)
%FEATURE extract feature vector from turning funciton for classification
%
%requests:
%  t : turning function of shape
%returns:
%  a  : feature vector
%transform turning function to periodic signal:
n = length(t);
square = t - 2*pi/n * [1:n];

%function [rectangle] = rectangle1()

test2= [];
L=[0;0];

n = length(rectangle1);

test2(1)= rectangle1(1);

for i = 2:n
    test2(i) = test2(i - 1) + rectangle1(i);
end
L(1,1) = 0;
L(2,1) = 0;
L(1,n+1) = 0;
```

```
L(2,n+1) = 0;
for i = 2 : n
        L(1,i)= cos(test2(i)) + L(1,i-1);
        L(2,i)= sin(test2(i)) + L(2,i-1);
end

n = length(L);
% Interpolate with a spline curve and finer spacing.
t = 1:n;
ts = 1: 0.1: n;
obj = csapi(t, L);


%function [t,coefficient,x,s] = turningfunction(obj,n)
%GRAPHS : plots of turning function and frequency contents
%  obj : object boundary as spline in ppform
%  n   : number of points on object contour
% gather object pnts
%equally spaced
s = linspace(obj.breaks(1),obj.breaks(end),n);

%equally spaced, partially occluded:
%s = linspace(obj.breaks(end)*0.2,obj.breaks(end)*0.8,n);

x = fnval(obj, s);
% compute turning function
% df = fnval(fnder(obj),s);
%
% %first tangent's angel relative to y = const:
% %absolute angels in [-pi,pi]
% t = atan2(df(2,:), df(1,:));
% %correct for pi -jumps
% if (abs(t(1) - t(n)) >= pi)
%   t(1) = t(1) + 2*pi;
% end;
% for i = 2 : n
%   %this condition certainly is not quite correct - but it works as long as the differnces of
%   %tangent angels are small => arclengths between pnts must be small
%   if (abs(t(i) - t(i-1)) >= pi)
%     t(i) = t(i) + 2*pi;
%   end;
% end;
vec = [[0;1],diff(x,1,2)]; %diff does not account for the last vector that closes the boundary!
len = [sqrt(sum(vec.^2,1))];
for i = 1 : n - 1
  cosphi = vec(:,i)' * vec(:,i+1) / (len(1,i) * len(1,i+1));
  phi(i) = acos(cosphi);
end;

t = cumsum(phi);
%--------------------------------------------------------------------
%function [a,E]=periodic(t)
%FEATURE extract feature vector from turning funciton for classification
%
%requests:
%  t : turning function of shape
%returns:
%  a  : feature vector
%transform turning function to periodic signal:
n = length(t);
rectangle = t - 2*pi/n * [1:n];
end
```

TESTFFT.m Returns the fourier transformation of test objects

```
function [yc_test]= TESTFFT(test,net)
% The classification of TEST set with the help of RBF. This function
% computes the max absolute
% The maximum absolute value of FFT of test ignores the phase angle.
% The FFT value of TEST is normalized because to make them independent of
% translation, scale and rotation.

FFT_test = fft(test);
abs_FFT_test = abs(FFT_test);
max_abs_FFT_test = max(abs_FFT_test);
norm_abs_FFT_test = abs_FFT_test/max_abs_FFT_test;
y_test = sim(net,norm_abs_FFT_test)
yc_test = vec2ind(y_test)

return
```

TESTTURNING.m Returns turning function to a periodic signal

```
function [n,X,fig] = TESTOBJECT(Test,fig)
%shape plot shape from the turning function square

n = length(Test);

X = [0;0];

for c = 1 : n - 1
  X(:,c+1) = X(:,c) + [cos(Test(c)); sin(Test(c))];
end;

fig = fig + 1;
figure(fig)
plot(X(1,:),X(2,:),'k-');
axis equal;
axis([-1,10,-1,10]);
Xlabel('arclength');
Ylabel('\theta');
axis([1,32,-pi/18,2*pi]);

label = {'0','PI/2','PI','3*PI/2','2*PI'};
set(gca, 'YTick' ,2*pi/4*[0:4]);
set(gca, 'YTickLabel' ,label);
title ('Turning function of Test')

%function [c,fig,X] = FEATUREsquare(X,fig)

%transform turning function to periodic signal:
n = length(X(1,:));
```

```
X(1,:)= X(1,:) - 2*pi/n * [1:n];


%%%%%%%%%%%%%%%%%%%%%%%%
%apply fourier transformation and strip contents over nyquist frequ. and
%static content
ftc = abs(fft(X(1,:)));
ftw = ftc(1,2:ceil(n/2)+1);

%return feature vector:
c = ftw;
% The figure of feature vector of TEST
fig = fig + 1;
figure(fig)
bar([0:(n-1)/2],c);
title('feature vector of Test');

% The figure of periodic signal of TEST
fig = fig + 1;
figure(fig)
plot(X(1,:));
title ('PERIODIC SIGNAL of Test')


return
```

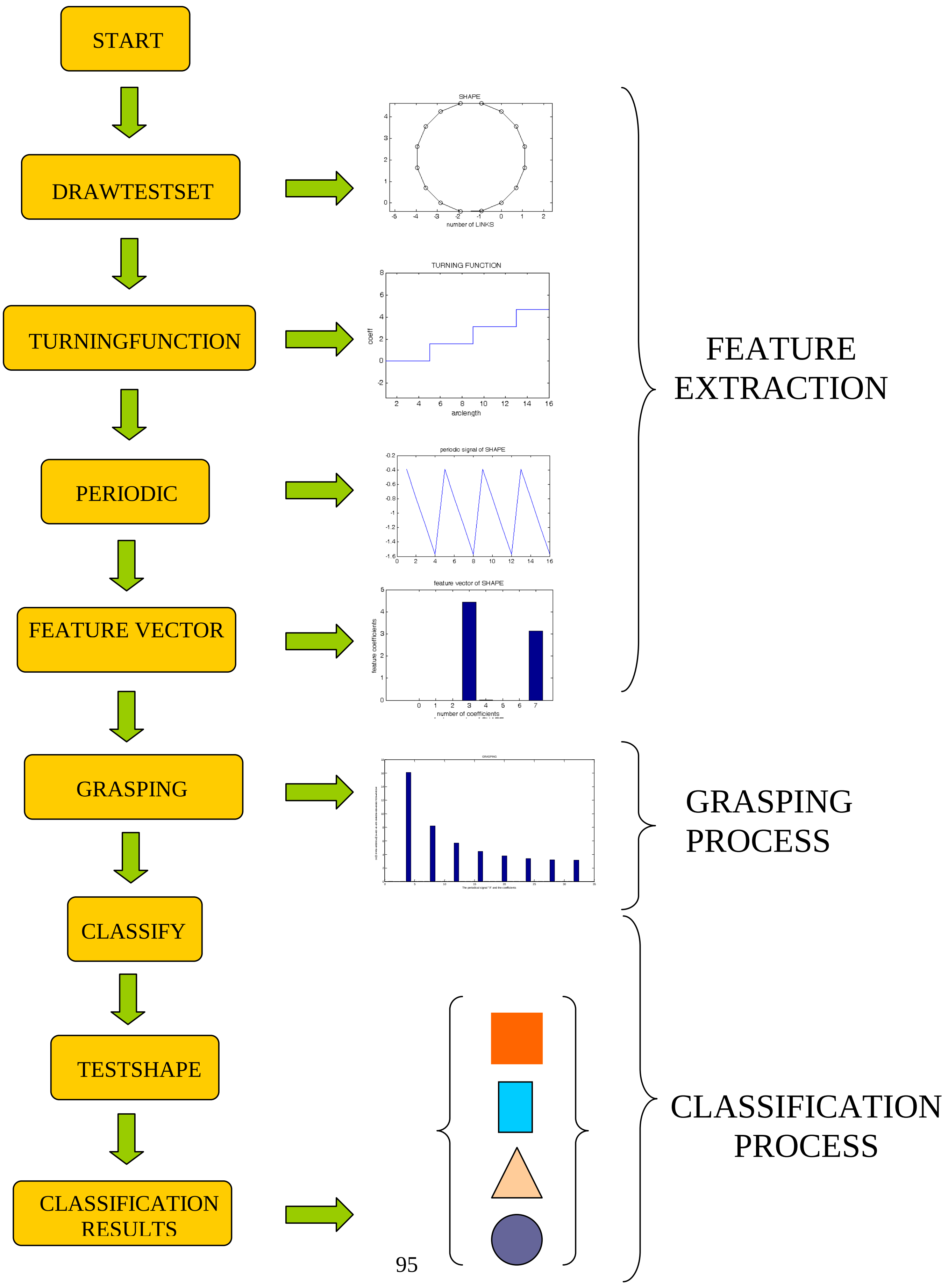
FFTscanner.m Returns discrete fourier transformation of two-dimensional objects

```
% FFT
% computes the fast fourier transform of the data arrays

function
[net,Y,Yc,T,P,norm_abs_FFT_alpha_square_1,norm_abs_FFT_alpha_square_2,norm_abs_FFT_alpha_square_3
,norm_abs_FFT_alpha_square_4,norm_abs_FFT_alpha_circle_1,norm_abs_FFT_alpha_circle_2,norm_abs_FFT
_alpha_circle_3,norm_abs_FFT_alpha_circle_4,norm_abs_FFT_alpha_triangle_1,norm_abs_FFT_alpha_triangl
e_2,norm_abs_FFT_alpha_triangle_3,norm_abs_FFT_alpha_triangle_4,norm_abs_FFT_alpha_rectangle_1,norm
_abs_FFT_alpha_rectangle_2,norm_abs_FFT_alpha_rectangle_3,norm_abs_FFT_alpha_rectangle_4] =
FFTscanner(square1,square2,square3,square4,circle1,circle2,circle3,circle4,triangle1,triangle2,triangle3,triangle
4,rectangle1,rectangle2,rectangle3,rectangle4)

FFT_alpha_square_1 = fft(square1);
FFT_alpha_circle_1 = fft(circle1);
FFT_alpha_triangle_1 = fft(triangle1);
FFT_alpha_rectangle_1 = fft(rectangle1);

FFT_alpha_square_2 = fft(square2);
FFT_alpha_circle_2 = fft(circle2);
FFT_alpha_triangle_2 = fft(triangle2);
FFT_alpha_rectangle_2 = fft(rectangle2);

FFT_alpha_square_3 = fft(square3);
FFT_alpha_circle_3 = fft(circle3);
FFT_alpha_triangle_3 = fft(triangle3);
```

```
FFT_alpha_rectangle_3 = fft(rectangle3);

FFT_alpha_square_4 = fft(square4);
FFT_alpha_circle_4 = fft(circle4);
FFT_alpha_triangle_4 = fft(triangle4);
FFT_alpha_rectangle_4 = fft(rectangle4);

% takes the absolute value of the data arrays

abs_FFT_alpha_square_1 = abs(FFT_alpha_square_1);
abs_FFT_alpha_circle_1 = abs(FFT_alpha_circle_1);
abs_FFT_alpha_triangle_1 = abs(FFT_alpha_triangle_1);
abs_FFT_alpha_rectangle_1 = abs(FFT_alpha_rectangle_1);

abs_FFT_alpha_square_2 = abs(FFT_alpha_square_2);
abs_FFT_alpha_circle_2 = abs(FFT_alpha_circle_2);
abs_FFT_alpha_triangle_2 = abs(FFT_alpha_triangle_2);
abs_FFT_alpha_rectangle_2 = abs(FFT_alpha_rectangle_2);

abs_FFT_alpha_square_3 = abs(FFT_alpha_square_3);
abs_FFT_alpha_circle_3 = abs(FFT_alpha_circle_3);
abs_FFT_alpha_triangle_3 = abs(FFT_alpha_triangle_3);
abs_FFT_alpha_rectangle_3 = abs(FFT_alpha_rectangle_3);

abs_FFT_alpha_square_4 = abs(FFT_alpha_square_4);
abs_FFT_alpha_circle_4 = abs(FFT_alpha_circle_4);
abs_FFT_alpha_triangle_4 = abs(FFT_alpha_triangle_4);
abs_FFT_alpha_rectangle_4 = abs(FFT_alpha_rectangle_4);

% computes the maximal absolute value of the data arrays

max_abs_FFT_alpha_square_1 = max(abs_FFT_alpha_square_1);
max_abs_FFT_alpha_circle_1 = max(abs_FFT_alpha_circle_1);
max_abs_FFT_alpha_triangle_1 = max(abs_FFT_alpha_triangle_1);
max_abs_FFT_alpha_rectangle_1 = max(abs_FFT_alpha_rectangle_1);

max_abs_FFT_alpha_square_2 = max(abs_FFT_alpha_square_2);
max_abs_FFT_alpha_circle_2 = max(abs_FFT_alpha_circle_2);
max_abs_FFT_alpha_triangle_2 = max(abs_FFT_alpha_triangle_2);
max_abs_FFT_alpha_rectangle_2 = max(abs_FFT_alpha_rectangle_2);

max_abs_FFT_alpha_square_3 = max(abs_FFT_alpha_square_3);
max_abs_FFT_alpha_circle_3 = max(abs_FFT_alpha_circle_3);
max_abs_FFT_alpha_triangle_3 = max(abs_FFT_alpha_triangle_3);
max_abs_FFT_alpha_rectangle_3 = max(abs_FFT_alpha_rectangle_3);

max_abs_FFT_alpha_square_4 = max(abs_FFT_alpha_square_4);
max_abs_FFT_alpha_circle_4 = max(abs_FFT_alpha_circle_4);
max_abs_FFT_alpha_triangle_4 = max(abs_FFT_alpha_triangle_4);
max_abs_FFT_alpha_rectangle_4 = max(abs_FFT_alpha_rectangle_4);

% the lower FFT values store the general information of the shape and the higher frequency
% the smaller details.The maximum absolute value of FFT of test ignores the phase angle.
% The FFT value of TEST is normalized because to make them independent of
% translation, scale and rotation.

norm_abs_FFT_alpha_square_1 = abs_FFT_alpha_square_1/max_abs_FFT_alpha_square_1
norm_abs_FFT_alpha_circle_1 = abs_FFT_alpha_circle_1/max_abs_FFT_alpha_circle_1
norm_abs_FFT_alpha_triangle_1 = abs_FFT_alpha_triangle_1/max_abs_FFT_alpha_triangle_1
norm_abs_FFT_alpha_rectangle_1 = abs_FFT_alpha_triangle_1/max_abs_FFT_alpha_rectangle_1

norm_abs_FFT_alpha_square_2 = abs_FFT_alpha_square_1/max_abs_FFT_alpha_square_2
```

```
norm_abs_FFT_alpha_circle_2 = abs_FFT_alpha_circle_1/max_abs_FFT_alpha_circle_2
norm_abs_FFT_alpha_triangle_2 = abs_FFT_alpha_triangle_1/max_abs_FFT_alpha_triangle_2
norm_abs_FFT_alpha_rectangle_2 = abs_FFT_alpha_triangle_1/max_abs_FFT_alpha_rectangle_2

norm_abs_FFT_alpha_square_3 = abs_FFT_alpha_square_1/max_abs_FFT_alpha_square_3
norm_abs_FFT_alpha_circle_3 = abs_FFT_alpha_circle_1/max_abs_FFT_alpha_circle_3
norm_abs_FFT_alpha_triangle_3 = abs_FFT_alpha_triangle_1/max_abs_FFT_alpha_triangle_3
norm_abs_FFT_alpha_rectangle_3 = abs_FFT_alpha_triangle_1/max_abs_FFT_alpha_rectangle_3

norm_abs_FFT_alpha_square_4 = abs_FFT_alpha_square_1/max_abs_FFT_alpha_square_4
norm_abs_FFT_alpha_circle_4 = abs_FFT_alpha_circle_1/max_abs_FFT_alpha_circle_4
norm_abs_FFT_alpha_triangle_4 = abs_FFT_alpha_triangle_1/max_abs_FFT_alpha_triangle_4
norm_abs_FFT_alpha_rectangle_4 = abs_FFT_alpha_triangle_1/max_abs_FFT_alpha_rectangle_4

%classificationHere a classification problem is defined with a set of inputs P and class indices T.
P = [norm_abs_FFT_alpha_square_1, norm_abs_FFT_alpha_square_2, norm_abs_FFT_alpha_square_3,
norm_abs_FFT_alpha_square_4, norm_abs_FFT_alpha_circle_1, norm_abs_FFT_alpha_circle_2,
norm_abs_FFT_alpha_circle_3, norm_abs_FFT_alpha_circle_4, norm_abs_FFT_alpha_triangle_1,
norm_abs_FFT_alpha_triangle_2, norm_abs_FFT_alpha_triangle_3, norm_abs_FFT_alpha_triangle_4,
norm_abs_FFT_alpha_rectangle_1, norm_abs_FFT_alpha_rectangle_2, norm_abs_FFT_alpha_rectangle_3,
norm_abs_FFT_alpha_rectangle_4];

% classes
% There are inclusive 4 classes of objects
T = ind2vec([1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4]);

% create a network. Probabilistic neural networks are a kind of radial
%basis network suitable for classification problems.
net = newpnn(P,T)

%Here the class indices are converted to target vectors,and a PNN is designed and tested.

Y=sim(net,P)
Yc=vec2ind(Y)
return
```

6.3 Flow Diagram



7 Discussion

The one-dimensional data arrays performed with the help of newpnn does not give sufficient results for the classification because of the extremely susceptible of the network to noise. It is alos possible to make an interpretation of being grasped or not from the manipulator by looking at the plotted shape. How many times it is entwined, could be only decided from the shape of the object and periodical signal of turning funciton. The object distance information is also avaliable with the help of turning function.

7.1 Future work

Being recognized and grasped from a manipulator were performed with the help of MLP network. The other neural network methods must be used to overcome this recognition problem with the entwined or not entwined objects that have not constant boundary point information.

The two-dimensional data arrays problem can not be solved with the help of radial basis functions. The corrupted objects were not classified correctly. Therefore it will be useful to study out this problem with the help of the other neural networks.

The future work could be the implementation of hand-drawn objects in recognizing and grasping process. In order to make a sufficient classification, the other neural networks must be used performing with pnn networks. Since classification is sensitive to object boundary and linearity of spaces between drawn points.

Bibliography

[Ark00] E.M.Arkin. Efficiently computable metric for comparing polygonal shapes. *IEEE transaction on Pattern Analysis and Machine Intelligence,* Vo1.13, no.3, 209-216, 2000.

[Bay] http://www.cs.mcgill.ca/~mcleish/644/main.html.

[Bec91] Suzanna Becker. Unsupervised learning procedures for neural networks. *International Journal of Neural Systems,* Vols 1&2, p 17-33, 1991.

[Bis95] Bishop, C. M. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK 1995.

[Burks] http://burks.brighton.ac.uk/burks/foldoc/30/87.htm.

[Bus02] Christian Busch. *Whole-arm grasping with hyper-redundant planar manipulators using neural networks.* Wien, 2002.

[Cast04] Castleman ch. *Pattern Recognition: Object Measurement & Classification*, 2004.

[Comp422] Mengjie Zhang. *Neural Networks for Object Recognition Applications,*1999.

[Cov65] T. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition . In *IEEE Transactions on Electronic Computers* EC-14,326-334, June 1965.

[Died03] Diedrich Wolter. *Shape: Representation & Recognition,* 2003.

[Guo01] Dengsheng Zhang and Guojun Lu Gippsland. A Comparative Study on Shape Retrieval Using Fourier Descriptors with Different Shape Signatures. School of Computing and Information Technology Monash University, 2001.

[Leo00] Leon Zlajpah. Planar Manipulators Toolbox For Use with Matlab/Simulink, 2000.

[L02] Irina Rish. Bayesian Decision Theory. BM T.J. Watson Research Center, 2003.

[Lec98] Bryan S. Morse, Shape Description (Regions) Brigham Young University, 1998–2000.

[Mar] Manolis Maragoudakis, Todor Ganchev and Nikos Fakotakis. Bayesian Reinforcement for a Probabilistic Neural Net Part-Of-Speech Tagger, Intelligent Systems Group, University of Patras Rion 26500, Patras, Greece, 2004.

[Mar04] Martin Brazda. *Neural Net Controlled Autonomous Hyper-Redundant Manipulator Systems*, diploma thesis, Institut für Handhabungsgeräte und Robotertechnik der Technischen Universität Wien, 2004.

[Meh97] B.M. Mehtre, M.S. Kankanhalli, and W.F. Lee, “Shape Measures for Content Based Image Retrieval: A Comparison,” *Information Processing & Management*, Vol. 33, No 3, pp. 319-337, 1997.

[m0039] Don Johnson. *Fourier Series* version 2.20, 2004.

[Mic04] Michel Verleysen. *Radial-Basis Function Networks*, 2004.

[pnn] http://www.netnam.vn/unescocourse/knowlegde/63.htm.

[Rip96] Pattern Recognition *via* Neural Networks, 1998.

[Sim88] Simon Haykin. *Neural Networks: A comprehensive foundation* 2nd International Ed. Prentice Hall, 1999.

[Smc00] Brian David Maciel and Richard Alan Peters. A Comparison of Neural and Statistical Techniques in Object Recognition. Center for Intelligent Systems Vanderbilt University School of Engineering, 2000.

[Sta] http://www.dcs.ex.ac.uk/ica/icapp/chapters.html

[Ste97] Stefan Kunze, Dr.Josef Pauli. *A Vision Based Robot System for Arranging Technical Objects.*1997.

[Velt99] Remco C. Veltkamp and Michiel Hagedoorn. *State of the art in Shape matching*. Utrecht University, Department of Computing Science Padualaan, 1999.

[wpnn] David Montana. *A Weighted Probabilistic Neural Network*,2002.