LEARNING APPROACHES TO CONTACT ESTIMATION IN ASSEMBLY TASKS WITH ROBOTS

M. Nuttin*, J. Rosell, R. Suárez**, H. Van Brussel*, L. Basañez** and J. Hao***

* *K.U.Leuven-PMA, Celestijnenlaan 300 B, B-3001 Heverlee, BELGIUM.*

** *Instituto de Cibernética (UPC-CSIC), Diagonal 647, 08028 Barcelona, SPAIN.*

Abstract. Fast and accurate contact estimation during assembly operations is an important requirement to guarantee successful executions using fine-motion planners. In this context, learning methods provide an attractive alternative to complicated analytical approaches. In order to generate examples of correct contact classifications for the learning process, a method has been developed. Then, starting from a data base of examples, three inductive learning approaches have been followed: backpropagation nets, radial basis function nets and classification trees. The comparison between the approaches is made based on the following criteria: accuracy, convergence speed, on-line estimation speed, compactness of representation, ease of use, and the possibility of interpretation.

Key Words. Exemplar based Learning, Robotic Assembly, Fine-motion Planning, Contact States.

1. INTRODUCTION

Assembly is a typical field of robot applications where contact between the manipulated object and the environment must be taken into account, specially due to the geometric uncertainty affecting the position of the objects.

A first solution to deal with geometric uncertainties in assembly has been the use of passive compliance [34], successfully applied in industrial tasks. A second and more general solution is the use of active compliance, which avoids the lack in flexibility of the passive devices [17]. Nevertheless, the use of active compliance leads to two other problems which must be properly solved:

- the *control* of the reaction force/torque besides the position/orientation control of the gripper.
- the determination of a *strategy* or *plan* to use reaction force/torque information as a guide to perform the assembly despite geometric uncertainty.

These two problems are not completely disjoint; in fact, there exist some approaches (e.g. reactive control [10] [21] and specific compliance matrix generation [26] [12]) including the strategy in the controller behaviour level. On the other hand, several approaches build a plan independently of the control solution [15] [5] [7] [29], determining a proper robot command to proceed the assembly from any task state. These last approaches must be able to (figure 1):

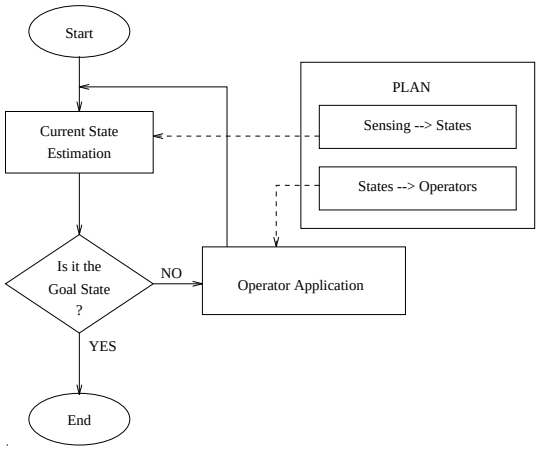

Figure 1 *Task execution*

- identify the current *task state* by using information from sensors,
- determine a proper robot command (*operator*) to successfully proceed the task from the current state.

This paper is related to the first of these two points, which is usually treated considering sens-

ing information describing the gripper configuration (position and orientation) and the generalized reaction force (force and torque) between the objects. Task states are defined here as the different contact situations between the objects to be assembled (i.e. the possible different sets of basic contacts) [30]. Previous works in this line are mainly based on the on-line execution of test movements to disambiguate contact situations [27] but they may change the task state in an undesired way, or on geometric reasonings [9] [35] by considering the models of the objects and the different sources of uncertainty. A new approach, including learning techniques, has been presented in [31], where its advantages and disadvantages in comparison with an analytical geometric approach are discussed. As a further step, in this paper, a comparison of three different learning techniques applied to the estimation of the current contact situation between the objects is presented, including some experimental results.

The rest of the paper is organized as follows: section 2 describes the framework and assumptions; section 3 presents three learning techniques and its experimental results; in section 4 a discussion and comparison of these techniques is presented and, finally, section 5 summarizes the conclusions of the work.

2. FRAMEWORK

The Task

The experiments have been performed over a problem with three degrees of freedom: the positioning of a block into a corner. Accepting that the block can be positioned closed enough to the corner by gross motion, nine different contact situations can be reached according to the nominal models of the objects (figure 2). The exact configurations in which each situation can be reached depend on the deviations of geometric characteristics: object shape and size, robot positioning, and undesired slippings of the object in the robot gripper. The effect of these sources of geometric uncertainty on the contact configurations has been modeled in [2], and the effect produced on the possible reaction forces is analyzed in [28].

From these models, a three degrees of freedom task simulator has been implemented, with the capability of showing the task evolution in the physical, the configuration and the force spaces simultaneously, considering uncertainty and friction forces [20].

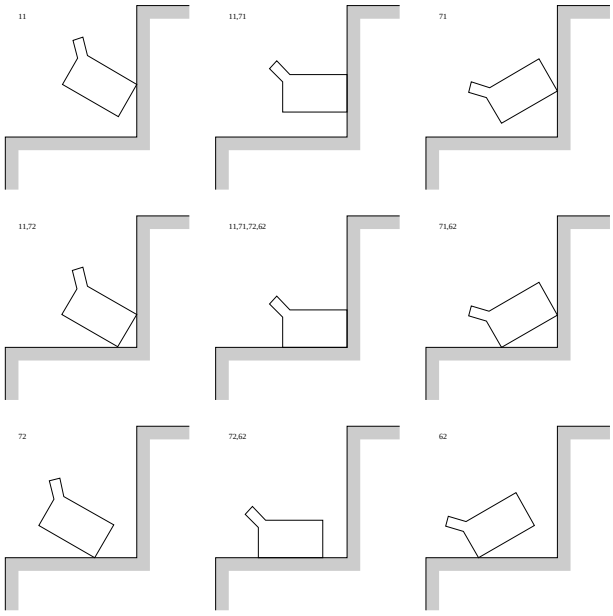

Figure 2 *Possible nine nominal states.*

Generation of training and test sample sets

The data sets of training and test data have been generated with the above simulator.

Any parameter subject to uncertainty is considered to have a uniform probability density function. In the same way, any random choice is equiprobable.

A state-sample is composed by three configuration data (x_0, y_0, ϕ_0), three force data (fx_0, fy_0, τ_0) and the corresponding task state label number.

Generation of the configuration

Instances of the deviations of all the parameters subject to uncertainty are randomly chosen and the actual C-surfaces for each basic contact are determined.

For one basic contact: first, an orientation ϕ_0 and then a point (x_0, y_0, ϕ_0) of the segment representing the section of the C-surface for $\phi = \phi_0$ are randomly chosen.

For two basic contacts involving different contact edges: first, an orientation ϕ_0 is randomly chosen, and then the intersection point (x_0, y_0, ϕ_0) of the two segments representing the sections of the C-surfaces for $\phi = \phi_0$ is computed.

For two basic contacts involving the same contact edge: first, the orientation ϕ_0 for which the C-surfaces of both basic contacts intersect is computed and then a point (x_0, y_0, ϕ_0) of the segment

representing this intersection is randomly chosen.

For four basic contacts:

- One of the four possible states of three basic contacts is randomly chosen (due to deviations the nominal four contact goal state will rarely appear; instead any non-nominal three contact state will be considered as goal state).
- One of the two edges of the corner as the edge involved in two basic contacts is randomly chosen.
- The orientation ϕ_0 that allows two basic contacts involving the edge selected in the previous step is computed.
- The pair (x_0, y_0) for which the configuration (x_0, y_0, ϕ_0) corresponds to three basic contacts is computed.

Generation of the reaction force

Once a contact configuration has been determined, a random reaction force is equiprobably selected between all the possible reaction forces compatible with the contact configuration. First the direction (from now direction meaning direction and sense) of the generalized reaction force is determined considering unitary module; then the actual module is randomly chosen within a predefined range.

For one basic contact:

- The two generalized forces bounding the generalized friction cone for the actual contact configuration are computed.
- A direction within the generalized friction cone is randomly chosen.

For two or three basic contacts:

- The two generalized reaction forces bounding the friction cone of each basic contact in the actual contact configuration are computed.
- Directions of the generalized force space are randomly chosen (Appendix A) until one selected direction can be expressed as a linear combination with positive coefficients of the directions computed in step 1.

For more than three basic contacts: A subsets of three contacts is randomly chosen and the previous algorithm is applied.

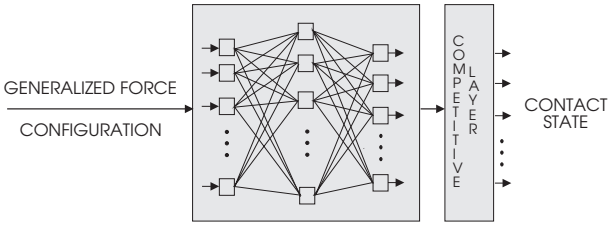

Figure 3 *A backpropagation neural net to estimate the contact state. The competitive layer selects the neuron with the highest output. Each output neuron represents a contact state.*

General Considerations

The main priority of the classification algorithm is not to confuse non-contiguous states, that is states such that movement from one to another requires crossing some other state. The reason is that the transition operators of contiguous states likely make the task evolve in a similar way, whereas two non-contiguous states usually not; therefore, in the latter case a confusion is much worse than in the former.

A confusion matrix will be generated to evaluate the classification results (table 1 shows the contiguity of the confusion matrix elements).

3. LEARNING METHODS FOR CONTACT IDENTIFICATION

Backpropagation Neural Nets

According to the connections among the neurons, artificial neural nets (ANN) are classified into feedforward nets and recurrent nets. The multilayer perceptron (MLP) net is one of the most important types of the class of feedforward nets. Since in most cases an MLP is trained using a

	1	2	3	4	5	6	7	8	9
1	.	c	n	n	n	n	n	c	c
2	c	.	c	n	n	n	n	n	c
3	n	c	.	c	n	n	n	n	c
4	n	n	c	.	c	n	n	n	c
5	n	n	n	c	.	c	n	n	c
6	n	n	n	n	c	.	c	n	c
7	n	n	n	n	n	c	.	c	c
8	c	n	n	n	n	n	c	.	c
9	c	c	c	c	c	c	c	c	.

Table 1 *Contiguity of the confusion matrix elements. A 'c' means the states are contiguous, a 'n' means the states are non-contiguous.*

backpropagation algorithm or its variants, it is also called a backpropagation net.

A backpropagation net features a layered structure and has weighted feedforward connections only between neurons in the adjacent layers. It is composed of a layer of input nodes, one up to several hidden layers of neurons and an output layer of neurons (figure 3). Each neuron in the network takes as input the sum of the weighted outputs from other neurons connected to it, and then passes the value through a nonlinear function. Typical examples for such function are a sigmoid and a hyperbolic tangent function, which are also used in our approach.

One of the most important properties of backpropagation nets relevant to the problems of classification and identification is the so-called universal approximation [13], which means that with enough but finite number of hidden neurons there always exist a set of weights such that the network can approximate any nonlinear function to a desired accuracy. Since any classification problem can be considered as a multi-input multi-output function approximation problem, this property virtually provides a theoretical foundation for using backpropagation nets to solve the problem of assembly contact estimation.

The backpropagation net is one of the earliest and most extensively studied networks in the artificial neural network field. It has been very successfully used in applications of nonlinear system identification and control [18][11], function approximation and various classification problems [14]. The use of a backpropagation net for a particular classification problem, needs a reasonably large set of training data which is composed of inputs and the corresponding desired outputs and a set of test data which is used to validate the network after training. It is also necessary to know the relative complexity of the given problem in order to come up with a network with an appropriate number of hidden neurons, while the number of hidden layers is normally chosen to be one or two. The number of hidden neurons is very important since if it is too small, the network will not be powerful enough to solve the problem, while if it is too large, the network will have the so-called generalization or over-fitting problem, i.e., the network fits well on the training set but very poor on the test set. Presently there is no systematic method to predetermine the number of hidden neurons, so the *trail-and-error* method is usually used.

Results for the backpropagation net

For the application mentioned above, the neural net has 6 inputs (configuration and force components) and 9 outputs (one for each state). The classical backpropagation algorithm was used, modified with momentum and a variable learning rate [33]. The networks were initialized with the Nguyen-Widrow initialization method [19]. Network with various number of hidden neurons were trained in order to find a network with the best generalization capability. MATLAB's neural toolbox [8] was used for this part of the experiment. Best results on the test set were obtained using a single layer, 15 hidden neurons with hyperbolic tangent activation function, and linear output neurons. In this case, the misclassification rate on an independent test set was 3.2%. Figure 4 shows the misclassification errors on training and test set, during the learning process. Table 2 shows the confusion matrix on the independent test set.

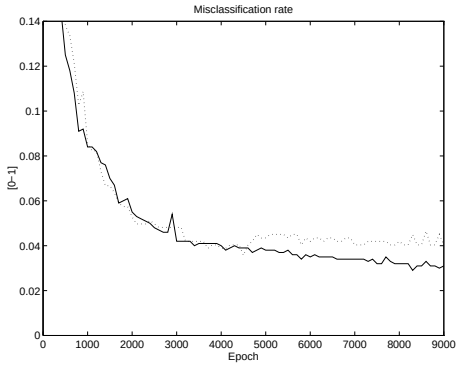

Figure 4 *Evolution of the classification error during the learning process of a backpropagation net on the training set (full line) and on the test set (dotted line).*

Radial Basis Function Neural Nets

%	1	2	3	4	5	6	7	8	9
1	96.3	0.0	0.0	0.0	1.3	0.0	1.8	0.0	0.0
2	0.0	100.0	0.0	0.0	0.0	5.4	0.0	0.0	0.0
3	0.0	0.0	96.3	0.0	0.0	2.8	0.0	0.0	0.0
4	0.0	0.0	0.0	98.7	1.3	0.0	0.0	0.0	0.0
5	3.7	0.0	0.0	1.2	97.4	0.0	0.0	0.0	1.8
6	0.0	0.0	3.7	0.0	0.0	91.8	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	98.2	0.0	3.5
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	7.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	87.7

Table 2 *The confusion matrix calculated on a test set, using a trained backpropagation net, with 15 hyperbolic tangent hidden neurons and 9 linear output neurons. The columns represent the true states, the rows give the output of the classifier. The classifications are reported in percentages.*

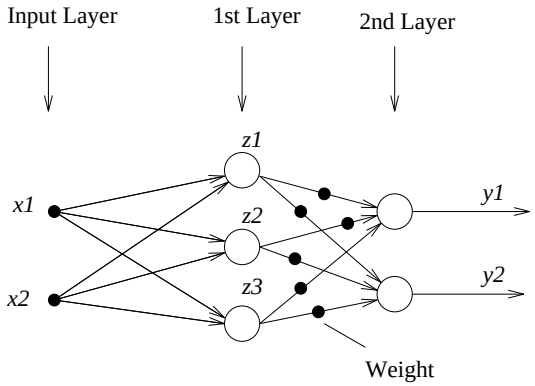

Figure 5 *The structure of a RBF neural net.*

A Radial Basis Function (RBF) net is yet another feedforward type of neural net and is attracting more and more attention from researchers for its unique properties in solving problems like function approximation, prediction and classification. A RBF neural net [4] is a two-layer feedforward network with the first layer of RBF neurons and the second layer of linear neurons as shown in figure 5. Adjustable weights among the neurons only exist from the neurons in the first layer to those in the second layer. The inputs are directly connected to the neurons in the first layer via unity weights.

Referring to figure 5, the ith RBF neuron in the first layer has the following transfer function

$$z_i(x) = r(||x - p_i||/\sigma_i), \tag{1}$$

where $z_i(x)$ is the ith output of the neuron in the first layer. The argument x is the input vector of the form $x = [x_1 \quad x_2 \quad \cdots \quad x_q]^T$; $r(\cdot)$ is a radial basis function; p_i, σ_i (a positive scalar[1]) are, respectively, the center and the width for the ith RBF neuron. p_i, σ_i may be different for different neurons. Finally, the operator $||\cdot||$ denotes the standard vector norm.

Denoting by y_i the output of the ith linear neuron in the second layer, and assuming that the number of RBF neurons is n, its transfer function is simply the weighted sum of the outputs from the first layer

$$y_i = \sum_{j=1}^{n} w_{ij} z_j(x),$$

where $y_i, w_{ij} \in \mathbb{R}$. The equation above clearly is a linear operation. For m such linear neurons, the following matrix expression can be written in

$$y = W z(x), \tag{2}$$

where $y = [y_1 \quad y_2 \quad \cdots \quad y_m]^T \in \mathbb{R}^m, z(x) = [z_1(x) \quad z_2(x) \quad \cdots \quad z_n(x)]^T \in \mathbb{R}^n$ and $W \in \mathbb{R}^{m \times n}$ with w_{ij} being the ijth element of W.

For the sake of definiteness, we shall choose the RBF function $r(\cdot)$ to be the Gaussian function, shown in figure 6 of the form

$$z_i(x) = e^{-||x-p_i||^2/\sigma_i^2}, \tag{3}$$

where the parameters p_i, σ_i are termed, respectively, the center and the standard deviation of the Gaussian function. σ_i is also called the spread parameter.

Using the preceding notations, the universal approximation property of RBF neural nets can be stated specifically as follows. With an appropriate, but finite, number of RBF neurons in the first layer and a proper setting of the parameters p_i and σ_i, there exists a weight matrix $\bar{W}$ such that the neural net with $\bar{W}$ can approximate, to an arbitrary precision, any continuous nonlinear function on a compact set (bounded and closed) [22]. This property is seen as a justification for RBF neural nets to be used as a generic model structure for classification problems. Another important feature of RBF nets is their linear-in-the-weights type of structure as shown in (2). This structural feature allows to separate the setting of the structural parameters n, p_i and σ_i from that of the weight matrix W.

Quite a few learning algorithms exist for the determination of the structural parameters (n, p_i and σ_i) and the weight matrix W, and basically, these algorithms can be classified into two categories. One class of algorithms perform the design task in two steps; first determine the structural parameters in some way (e.g. random selecting from the data points, using a clustering method, or using the spatial Fourier transform theory [25]) and then calculate the weight matrix using, e.g., error model based method [32]. The other class of algorithms integrates the determination of the structural parameters and the weight matrix in one process. Examples of these type of algorithms are the Orthogonal Least Squares (OLS) learning algorithm [6] and the Resource-Allocating Network (RAN)[23]. In this work, the OLS learning algorithm (Appendix B) has been adopted for the RBF net.

Results with RBF nets

The only design parameter as input of the algorithm described above, is the spread parameter σ_i in equation 3. If the spread is chosen too small, the network will not generalize enough. If the spread is chosen too large, many RBF neurons will respond essentially in the same way, which makes the output less dependent of the input.

[1] In general, σ_i need not be a scalar. It can be a vector or even a matrix as in [24] [16].

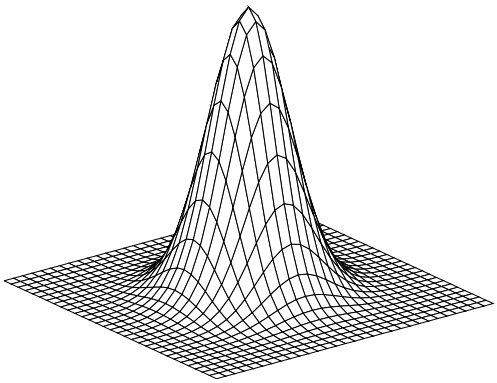

Figure 6 *A typical activation function for a radial basis function net: the Gauss function* $z_i(x) = e^{-||x-p_i||^2/\sigma_i^2}$ *for* $x = [x_1, x_2]^T$

state	average distance	minimum distance
1	0.38	0.088
2	0.36	0.089
3	0.31	0.080
4	0.47	0.106
5	0.48	0.099
6	0.45	0.085
7	0.46	0.093
8	0.45	0.091
9	1.61	0.207

Table 3 *Average and minimum distance between points of a same state. At this stage the numbers are considered to be dimensionless.*

This will result in a poor approximation of the training data.

We can choose an initial value for the spread parameter in the following way. The spread parameter must be larger than the minimum distance between any arbitrary pair of data points in the input space, belonging to the same state (see table 3). The spread parameter should also be of the same order of magnitude than the average distance between any two points of the input space, belonging to the same state. The spread parameter should be much smaller than the maximum distance between any two data points. The maximum distance between two data points is 5. If other units are used for force or configuration space, the network must be retrained, and the spread parameter will have to be optimized again. Having obtained a rough value for the spread in this way, we further optimized this parameter in an experimental way as shown in table 4 and in figure 7 that report the classification results. The minimum misclassification error on the test set is 2.8%. Table 5 shows the confusion matrix.

spread	class. error train. set	class. error test set	no epochs total	epoch min test set
0.1	0.0030	0.073	478	300
0.3	0.0030	0.034	449	100
0.5	0.0020	0.029	562	90
0.9	0.0020	0.028	635	100
1.1	0.0020	0.028	653	120
1.3	0.0020	0.035	655	110
1.5	0.0020	0.0389	699	90

Table 4 *Spread parameter (column 1), the classification error (column 2) on the training set after the number of epochs (column 4), the smallest classification error on the test set (column 3) that occurred after the number of training epochs listed (column 5).*

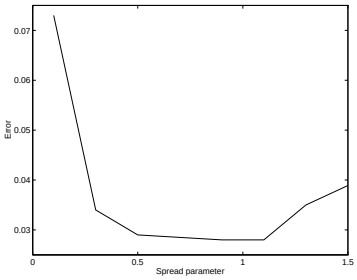

Figure 7 *Classification error on the test set as a function of the spread design parameter.*

Figures 8 and 9 show that typically the sum-squared error and the misclassification error on the training set can continuously be reduced, whereas the errors on the test set have a clear minimum.

Classification Trees

A classification (decision) tree is another interesting approach to the problem of contact estimation. Figure 10 shows an example of a simple classification tree. It represents a deterministic mapping $Y = T(X)$ from a set of feature vectors X to a set of classes Y. This classification tree is a binary tree but not necessarily balanced. A test, based on a single feature vector, is associated to every non-terminal node. This test is also called the split criterion. Each terminal node is associated with a target class.

Algorithms to build the trees from data are described in [3] which solve the following problems: (1) how to find good splits, (2) when to stop splitting and (3) how to assign a class to a terminal node. A tree is built by repeated splitting of a data set into subtrees. A criticism to this

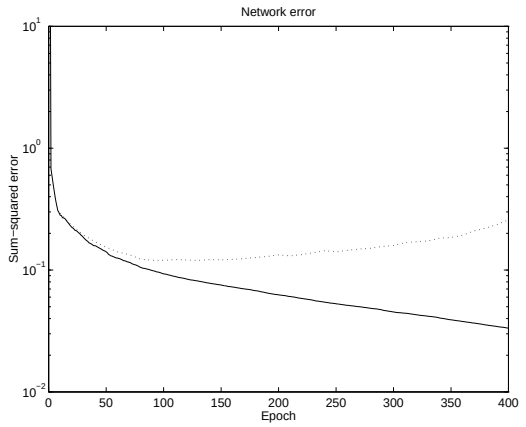

Figure 8 *Sum-squared error on the training set (full line) and on the test set (dotted line) as a function of the number of training epochs (spread parameter 0.5).*

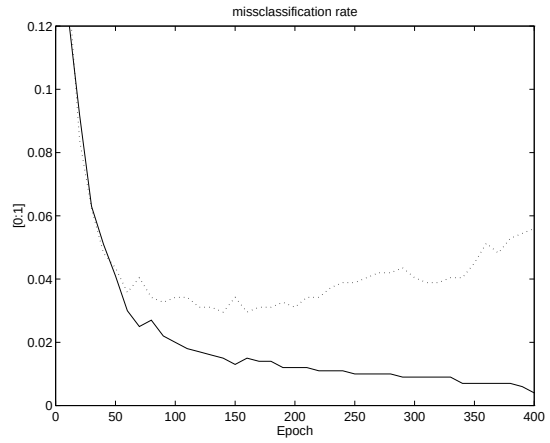

Figure 9 *Classification error on the training set (full line) and on the test set (dotted line) as a function of the number of training epochs (spread parameter 0.5).*

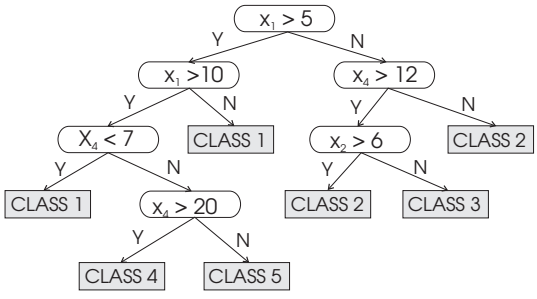

Figure 10 *Example of a simple classification tree with 5 target classes and vectors of 4 features.*

approach is that the split criteria are optimized at every step; but the splits are not overall optimal. However, by continuously splitting a tree, it is very easy to obtain accurate results on the training set; but generalization is not straightforward. In order to solve this problem, a pruning phase is introduced after the building phase. In the pruning phase, the tree is explored bottom up, and every inner node is eliminated if the error on independent data set doesn't increase. For a fair comparison, the training set used for the ANN approach must still be split up into a training set and a pruning set for the tree approach. Refer to [3] and [1] for details on the algorithms used.

Results with classification trees

Using the classification tree approach, (without any tuning of parameters), an accuracy of 98% is obtained on the training set, and an accuracy of 84% on the test set. The trees has typical 400 nodes, with the longest path of 21. Table 6 shows an example of a confusion matrix obtained with classification trees.

4. COMPARISON

The nature of the three learning approaches discussed in section 3 has important differences: the activation functions of RBF nets act locally, whereas the activation functions of backpropagation nets are not localized (i.e. they are 'open'). With respect to this point, classification trees are closer to backpropagation nets. It is not immediately clear which approach could be the best for this application. Therefore, a comparison has been made using the following criteria: (1) the accuracy on the test set, (2) the speed of convergence, (3) the on-line estimation time, (4) the compactness of representation, (5) the ease of use, (6) the possibilities of interpreting the classifier

%	1	2	3	4	5	6	7	8	9
1	97.5	0.0	0.0	0.0	1.3	0.0	1.8	0.0	0.0
2	0.0	98.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	96.2	0.0	0.0	5.6	0.0	0.0	0.0
4	0.0	0.0	0.0	98.7	0.0	0.0	0.0	0.0	0.0
5	1.3	0.0	0.0	1.3	98.7	0.0	0.0	0.0	3.4
6	0.0	1.3	2.5	0.0	0.0	94.4	0.0	0.0	0.0
7	1.2	0.0	0.0	0.0	0.0	0.0	98.2	0.0	1.8
8	0.0	0.0	1.3	0.0	0.0	0.0	0.0	98.4	8.8
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.6	86.0

Table 5 *The confusion matrix calculated on a test set using a trained RBF network (with as spread parameter 0.9). The columns represent the true states, the rows give the output of the classifier. The classifications are reported in percentages.*

%	1	2	3	4	5	6	7	8	9
1	93.8	1.8	0.0	0.0	0.0	0.0	1.2	14.3	0.0
2	2.5	90.9	13.8	0.0	0.0	0.0	0.0	1.3	0.0
3	0.0	1.8	65.0	9.6	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	21.2	71.2	8.8	1.6	0.0	0.0	1.8
5	0.0	0.0	0.0	19.2	86.2	4.8	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	5.0	84.1	5.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	92.5	3.9	0.0
8	3.8	0.0	0.0	0.0	0.0	0.0	1.2	76.6	7.0
9	0.0	5.5	0.0	0.0	0.0	9.5	0.0	3.9	91.2

Table 6 *Confusion matrix calculated on a test set using classification trees: the columns represent the true states, the rows give the classification that the classifier makes. The classifications are reported in percentages.*

Comparison	BP-ANN	RBF-ANN	TREE
accuracy	2	1	3
convergence speed	2	3	1
estimation speed	2	3	1
compactness	1	3	2
interpretability	3	1	2
ease in use	3	2	1
extendability	3	2	1

Table 7 *The three classification methods are compared: an entry '1' corresponds to the best method.*

and (7) the extendability to more complex tasks. The results are summarized in table 7.

The training set contained 1000 examples. RBF nets gives the best classification accuracy (97.2%), calculated on an independent test set of 600 examples, closely followed by backpropagation nets (96.8%). The classification trees perform worst (84%).

RBF nets generally require a smaller number of training epochs than backpropagation nets do. However, the particular algorithm used here for training RBF nets is very slow, because the optimization strategy to determine the clusters centers described in appendix B is very time consuming. Using an HP-9000 workstation, the average convergence times were 8 hours for a backpropagation net, 18 hours for a RBF-net, and 1/3 hour for a classification tree. The trees were implemented in C, and they work remarkably fast and are therefore very handy to experiment with for various data sets.

As the trained classifiers are used on-line, it is also important to compare the estimation time, i.e., the time necessary to determine the contact state, given the configuration and the generalized force. The following comparison is valid assuming software implementation of all classifiers. Classification trees are the fastest as they need to evaluate a maximum of 20 inequalities. RBF nets require more time than backpropagation nets, because they have more neurons (120 versus 15). All these learning approaches have better estimation times than the analytical approach [31]. This real-time aspect is also related to the compactness of representation. As memory requirement is not a problem for the application, the compactness of representation is only important with respect to the generalization capability.

The design of a RBF net is easier than the design of a backpropagation net. Using the methods described above, only the value of the spread parameter must be optimized. A rough value for this parameter can even be estimated from the training data. For a backpropagation net more trials are necessary to determine an appropriate network topology. For some topologies, the learning algorithm didn't even converge. The easiest and most robust method clearly is the classification tree approach. The user only needs to choose non critical values concerning the desired accuracy, maximum tree dimension and one of the possible alternative algorithms. Even the first trial is successful.

For tasks involving more tasks states (more outputs) or more degrees of freedom (more inputs), the dimension of the problem increases, and so the complexity of the classifier also increases. The classification tree seems the approach more easily extendable to higher dimensionality.

A final aspect of comparison is the possibility of interpreting the resulting classifier. With RBF nets it is possible to interpret the cluster centers as characteristic points in the feature space. The decision trees can be displayed graphically. Interpretation of the split criteria (as shown in figure 10) is possible. The weights of a backpropagation net are almost impossible to interpret.

5. CONCLUSIONS

The execution of fine-motion plans requires the estimation of the current contact state in order to determine the robot commands to apply. The learning approach has shown as an efficient way to contact estimation. A procedure has been developed to gather, in an automated way, a set of examples to learn from. Three learning approaches for this application has been compared: backpropagation nets, RBF nets and classification trees. RBF nets are more accurate in contact estimation, and are easier to design than backprop-

agation nets. However, RBF nets require most computing time for real-time contact estimation. Classification trees are even easier to use and more robust in convergence than RBF nets. Classification trees are additionally very fast for real-time estimation. However, their generalization capabilities are bad in comparison with neural nets. If the accuracy is enough for the application, classification trees seem to be the more promising approach for the integration of the classifier into a fine-motion planner because of its estimation speed, robustness, ease of use and extendability.

*

APPENDIX A: EQUI-PROBABLE GENERATION OF A 3D DIRECTION The problem of randomly generating with a uniform probability density function a direction of the generalized force space is equivalent to the equiprobable selection of a point $s = (\phi, \theta)$ on the unit sphere, which can be described as:

$$\begin{array}{ll} x = sin(\theta)cos(\phi) & 0 \leq \phi < 2\pi \\ y = sin(\theta)sin(\phi) & 0 \leq \theta \leq \pi \\ z = cos(\theta) & \end{array}$$

The probability density function of the spheric coordinates, $p(\phi)$ and $p(\theta)$, to be used for selecting the point, can be obtained from the uniform probability density function $p(s)$ desired for the sphere surface.

Let $ds = sin(\theta)d\theta d\phi$ be a differential of surface. Since $\int\int_{sphere} p(s)ds = 1$, then $p(s) = \frac{1}{4\pi}$ and $p(s)ds = \frac{1}{4\pi}sin(\theta)d\theta d\phi$.

$p(\theta)d\theta$ and $p(\phi)d\phi$ can be obtained by integrating $p(S)dS$ over a curve of constant ϕ and constant θ, respectively:

$$\begin{array}{l} p(\theta)d\theta = \int_0^{2\pi}(\frac{1}{4\pi}sin(\theta)d\theta)d\phi = \frac{1}{2}sin(\theta)d\theta \\ p(\phi)d\phi = \int_0^{\pi}(\frac{1}{4\pi}sin(\theta)d\phi)d\theta = \frac{1}{2\pi}d\phi \end{array}$$

Therefore to uniformly select a point on the unit sphere, it is necessary to select the spheric coordinates ϕ and θ with probability density functions $p(\phi) = \frac{1}{2\pi}$ and $p(\theta) = \frac{1}{2}sin(\theta)$ in the ranges $0 < \phi < 2\pi$ and $0 < \theta < \pi$.

*

APPENDIX B: OLS LEARNING ALGORITHM

The Orthogonal Least Squares learning algorithm integrates the determination of the structural parameters and the weight matrix in one process [6].

It is first assumed that out of N given data, a number of M inputs have been selected as the centers for the RBF net and all the centers have a known constant as their widths, then we have

$$d = P\Theta + E, \tag{4}$$

where $d = [d(1)\cdots d(N)]^T$, $P = [p_1 \cdots p_M]$, $p_j = [p_j(1)\cdots p_j(N)]^T$, $\Theta = [\theta_1 \cdots \theta_M]^T$, $E = [\epsilon(1)\cdots\epsilon(N)]^T$, in which $d(i)$ is the desired output of the ith training sample (suppose there is only one output neuron), $p_j(i)$ is the output of the jth RBF neuron when the input of the network is the ith sample, θ_j is the weight from the jth RBF neuron to the output neuron, ϵ_i is the modeling error for the ith sample, and $i = 1, 2, \cdots, N$, $j = 1, 2, \cdots, M$.

With these notations, it can be immediately seen that the Least Squares solution $\hat{\Theta}$ to (4) is the projection of d onto the space spanned by the vectors $p_1 \cdots p_M$. Since these vectors $\{p_i\}$ are generally correlated, it is not clear how an individual vector contributes to the final solution.

The OLS method proceeds by performing the following transformation to the matrix P

$$P = WA, \tag{5}$$

where A is an $M \times M$ triangular matrix with 1's on the diagonal and 0's below the diagonal and W is an $N \times M$ matrix with orthogonal columns w_i such that

$$W^T W = H, \tag{6}$$

where H is diagonal with elements h_i.

The space spanned by the set of orthogonal basis vectors w_i is the same space spanned by the set of p_i, and (4) can be rewritten as

$$d = Wg + E. \tag{7}$$

Then the orthogonal LS solution $\hat{g}$ is given by

$$\hat{g} = H^{-1}W^T d \tag{8}$$

or

$$\hat{g}_i = w_i^T d/(w_i^T w_i), \quad 1 \leq i \leq M. \tag{9}$$

Vectors $\hat{g}$ and $\hat{\Theta}$ satisfy the triangular system

$$A\hat{\Theta} = \hat{g}.$$

As stated earlier, this orthogonal LS solution depends on the predetermined set of M centers out of the N training samples, which is not necessarily the best one. In the following a recursive method for picking up the M optimal centers is described. To proceed, from (7),

$$d^T d = \sum_{i=1}^{M} g_i^2 w_i^T w_i + E^T E. \tag{10}$$

Thus $g_i^2 w_i^T w_i$ is the increment to the desired output introduced by w_i, and an error reduction ratio due to w_i can be defined as

$$[\mathrm{err}]_i = g_i^2 w_i^T w_i/(d^T d), \quad 1 \le i \le M. \qquad (11)$$

Since this ratio offers a simple and effective means of seeking a subset of significant samples as centers, the algorithm is as follows:

- *At the first step, for* $1 \le i \le M$, *compute*

$$\begin{cases} w_1^{(i)} = p_i \\ g_1(i) = (w_1^{(i)})^T d/((w_1^{(i)})^T w_1^{(i)}) \\ [err]_1^{(i)} = (g_1^{(i)})^2 (w_1^{(i)})^T w_1^{(i)}/(d^t d) \end{cases}.$$

  Find

$$[err]_1^{(i_1)} = \max_{1 \le i \le M} \{[err]_1^{(i)}\}$$

  and select

$$w_1 = w_1^{(i_1)} = p_{i_1}.$$

- *At the* k*th step where* $k \ge 2$, *for* $1 \le i \le M$, $i \ne i_1, \cdots, i \ne i_{k-1}$, *compute*

$$\begin{cases} \alpha_{jk}^{(i)} = w_j^T p_i/(w_j^T w_j), \ \ 1 \le j < k \\ w_k^{(i)} = p_i - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} w_j \\ g_k(i) = (w_k^{(i)})^T d/((w_k^{(i)})^T w_k^{(i)}) \\ [err]_k^{(i)} = (g_k^{(i)})^2 (w_k^{(i)})^T w_k^{(i)}/(d^t d) \end{cases}.$$

  Find

$$[err]_k^{(i_k)} = \max_{1 \le i \le M, i \ne i_1, \cdots, i \ne i_{k-1}} \{[err]_k^{(i)}\}$$

  and select

$$w_k = w_k^{(i_k)} = p_{i_k} - \sum_{j=1}^{k-1} \alpha_{jk} w_j,$$

  where $\alpha_{jk} = \alpha_{jk}^{(i_k)}$, $1 \le j < k$.

- *The procedure is terminated at the* M_s*th step when*

$$1 - \sum_{j=1}^{M_s} [\mathrm{err}]_j < \rho,$$

  where $0 < \rho < 1$ *is a chosen tolerance. This gives rise to a subset model containing* M_s *significant centers.*

ACKNOWLEDGEMENTS

This work was partially supported by the ESPRIT Project B-LEARN II under contract n°7274 and the CICYT project TAP93-0415, and the Interuniversity Attraction Pole IUAP-50, initiated by the Belgian Ministry of Science Policy. J. Rosell has a grant from the Catalan Government. The authors would like to thank A. Giordana, M. Kaiser, C. Baroglio, R. Piola and G. Lo Bello for their help and support.

REFERENCES

[1] C. Baroglio, A. Giordana and R. Piola, "Learning control functions for industrial robots", *Proceedings of the "Workshop on Robotics", Machine Learning Conference*, New Brunswick, New Jersey. 1994.

[2] L. Basañez and R. Suárez, "Uncertainty Modelling in Configuration Space for Robotic Motion Planning", *Proc. of the SYROCO'91*, Vienna, Austria, pp. 675-680, 1991.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification And Regression Trees, *The Wadsworth Statistics/Probability Series, Wadsworth*, 1984.

[4] D. S. Broomhead and D. Lowe (1988), "Multivariable functional interpolation and adaptive networks," *Complex Systems*, Vol.2, pp.321-355.

[5] S.J. Buckley, "Planning and Teaching Compliant Motion Strategies", MIT Artificial Intelligence Lab., report AI-TR-936 (Ph.D Thesis), 1987.

[6] S. Chen, C.F.N. Cowan and P.M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks,* Vol.2, No.2, pp.302-309, 1991.

[7] G. Dakin, "Fine-motion planning for robotic assembly in local contact space". Ph.D. Thesis, Univerity of Massachusetts, Amherst, MA, 1994.

[8] H. Demuth and M. Beale. *Neural Network Toolbox.* The Math Works Inc., Cochituate Place, 24 Prime Park Way, Natick, Massachusetts 01760, 1994.

[9] R.S. Desai and R.A. Volz "Identification and verification of termination conditions in fine-motion in presence of sensor errors and geometric uncertainties", *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, Arizona, USA, pp. 800-807.

[10] V. Gullapalli, A. G. Barto and R. A. Grupen, "Learning Admittance Mappings for Force-guided Assembly", *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp.2633-2638.

[11] J. Hao, S. Tan and J. Vandewalle, "Predictive control of nonlinear systems based on identification by backpropagation networks," *International Journal of Neural Systems,* in press, 1994

[12] S. Hirai and K. Iwata, "Derivation of Damping Control Parameters Based on Geometric Model", *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, pp. 87-92.

[13] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Net.*, Vol.2, pp.359-366, 1989.

[14] W.Y. Huang and R.P. Lippmann, "Neural net and traditional classifiers," *Neural Information Processing Systems,* Ed. D.Z. Anderson, pp.387-396, 1988.

[15] T. Lozano Perez, M. Mason and R. Taylor, "Automatic Synthesis of Fine-Motion Strategies for Robots", *The Int. Journal of Robotics Research*, 3 (1), pp. 3-24, 1984.

[16] S. Lee and R.M. Kil, "Multilayer feedforward potential function network," *Proc. IEEE Intern. Conf. Neural Net.*, Vol. 1, pp.161-171, 1988.

[17] M. Mason "Compliant Motion". *Robot Motion*, M. Brady et al. eds. Cambridge: MIT Press, 1983.

[18] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. on Neural Networks,* Vol.1, pp.4-27, 1990.

[19] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights", *International Joint Conference of Neural Networks*, volume 3, pages 21–26, July 1990.

[20] M. Nuttin, A. Giordana, M. Kaiser and R. Suárez R. B-learnII, workpackage 2, compliant motion in assembly. Deliverable 203 ESPRIT BRA Project No. 7274, Sept.1994

[21] M. Nuttin, H. Van Brussel, C. Baroglio and R. Piola, "Fuzzy Controller Synthesis in robotic assembly: Procedure and experiments", *FUZZ-IEEE94: Third International Conference on Fuzzy Systems, World Congress on Computational Intelligence, 1994.*

[22] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, Vol.3, pp.246-257, 1990.

[23] J. Platt, "A resource-allocation network for functional approximation," *Neural Computation*, Vol. 3, pp.213-225, 1991.

[24] M. Röscheisen *et.al*, "Incorporating prior knowledge in parsimonious networks of locally-tuned units," *Siemens Technical Report TR-FKI-155-91*, 1991.

[25] R. M. Sanner and J. E. Slotine, "Gaussian networks for direct adaptive control," *IEEE Trans. Neural Net.*, Vol. 3, pp.837-863, 1992.

[26] J. M. Schimmels and M. A. Peshkin, "Synthesis and Validation of Non-Diagonal Accommodation Matrices for Error-Corrective Assembly", *Proceedings of the IEEE 1990 International Conference on Robotics and Automation*, pp.714-719.

[27] M. Spreng, "A probabilistic Method to Analyze Ambiguous Contacts Situations", *Proceedings of the IEEE 1993 International Conference on Robotics and Automation*, Atlanta, USA, Vol. 3 pp. 543-548.

[28] R. Suárez, L. Basañez and J. Rosell, "Assembly Contact Force Domains in the Presence of Uncertainty", *Proc. of the SYROCO'94*, Capri, Italy, 1994.

[29] R. Suárez and L. Basañez, "Assembly With Robots in Presence of Uncertainty", *Proc. of the 22nd ISIR*, Detroit, USA, pp. 19/1–19/15, 1991.

[30] R. Suárez and L. Basañez, "Fine Motion Planning in Presence of Uncertainty", *II European Workshop on Learning Robots*, Torino, Italy, pp.143-164 ,1993.

[31] R. Suárez, L. Basañez, M. Nuttin, H. Van Brussel and J. Rosell, "Learning versus Analytical Approach to Contact Estimation in Assembly Tasks with Robots", *Technical Report IC-DT 9501, and KUL-PMA 95R02*, 1995.

[32] S. Tan, J. Hao and J. Vandewalle, "Stable and efficient neural network modeling of discrete multi-channel signals," *IEEE Trans. Circuits Syst.,* in press, 1994.

[33] T.P. Vogl, J.K. Mangis, A.K. Rigler, W.T. Zink and D.L. Alkon, "Accelerating the convergence of the backpropagation method", *Biological Cybernetics* Vol. 59, pp. 257-263, 1988.

[34] D. Whitney and J. Nevins, "What is the Remote Center of Compliance (RCC) and what can it do?", *Proceedings of the 9th International Symposium on Industrial Robots*, Washington DC, USA, pp. 135-152, 1979.

[35] J. Xiao, "Automatic Determination of Topological Contacts in the Presence of Sensing Uncertainties", *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, Atlanta, USA, Vol. 1 pp. 65-70.