Robot Learning - Three case studies in Robotics and Machine Learning

M. Kaiser, L. Camarinha-Matos, A. Giordana, V. Klingspor
J. del R. Millán, F.G.B. De Natale, M. Nuttin, R. Suárez

ESPRIT Basic Research Action No. 7274
B-Learn II

Contact address:
Prof. Dr.-Ing. R. Dillmann
University of Karlsruhe
Institute for Real-Time Computer Systems & Robotics
D-76128 Karlsruhe, Germany

ABSTRACT

This paper describes methodologies applied and results achieved in the framework of the ESPRIT Basic Research Action B-Learn II (project no. 7274). B-Learn II is one of the first projects working towards an application of Machine Learning techniques in fields of industrial relevance, which are much more complex than the domains usually treated in ML research. In particular, B-Learn II aims at easing the programming of robots and enhancing their ability to cooperate with humans.

The paper gives a short introduction to learning in robotics and to the three applications under consideration in B-Learn II. Afterwards, learning methodologies used in each of the applications, the experimental setups, and the results obtained are described.

In general, it can be found that providing good examples and a good interface between the learning and the performance components is crucial for success, so the extension of the "Programming by Demonstration" paradigm to robotics has become one of the key aspects of B-Learn II.

1 INTRODUCTION

The application of Machine Learning techniques in real-world applications is currently a topic gaining a lot of interest. However, the real world often poses much stronger requirements on the learning methods than problems considered in the Machine Learning community usually do. Missing or noisy, continuous-valued data, context- or time-dependent information and system behaviours have proven to be difficult to handle. While this is mainly a problem of symbolic learning techniques, the application of subsymbolic learning techniques such as neural networks is limited by their lack of ability to communicate and verify the knowledge that has been built up during the learning process.

Apart from this, it must be noted that real-world problems can usually not simply be classified as being completely located on the symbolic respectively the subsymbolic layer. In fact, most complex tasks require some *reflexive* skills, which are usually associated with a subsymbolic component, as well as *planning* or *reasoning* capabilities, typically to be realized by means of a symbolic module ([145]). This observation results in the natural demand for a functional integration of these components. In addition to the common practice of designing an hierarchical control system ([1]) and adding "learning capabilities" as some kind of module, *Multistrategy Learning* ([93]) and *Integrated Learning Architectures* ([115]) are currently investigated in order to achieve integration not only with respect to control but also regarding the learning tasks. Currently, approaches to Multistrategy Learning are mainly aiming at integrating empirical and analytical learning methods in order to speed up the empirical learning process ([56], [100], [21]). The integration in symbolic connectionist systems is usually limited to rule compilation ([149], [53], [148]) or the mapping of networks into rules ([147]), both closely related to the field of fuzzy neural networks ([17], [63], [50]).

In B-Learn II, the intgeration problem is tackled from both the application and the machine learning point of view. The conceptual design of the architectures employed is mainly influenced by the individual application, while the actual representation mechanisms used are selected according to the needs of the learning techniques.

2 LEARNING IN ROBOTICS

Considering the structure of a typical robot system, there are numerous opportunities to learn. They include learning of direct sensor–actor couplings on the lowest level of robot control as well as the acquisition of knowledge about the robot's environment or the continuous improvement of the robot's reasoning and planning capabilities, in order to enable it to handle increasingly complex tasks over time.

However, the traditional hierarchical structure of a robot control system (such as the NASREM architecture, see [1]) does typically not support the application of learning techniques on all its levels. Usually, it features distinct handling of object knowledge and action knowledge, with the latter being represented only implicitely in the form of executable code. In this implicit representation, knowledge is not learnable[1], hence most work concerning learning deals with the improvement of control knowledge by means of Neural Networks ([79], [135], [146]) or reinforcement learning ([19], [98], [94], [96], [113]). Especially the application of symbolic learning in Robotics usually considers only simple problems or takes place in simulation only (see [152], [77], or [78] for an overview).

The main conclusion that can be drawn from an analysis of the application of Machine Learning to Robotics is that a successful employment of learning techniques on all levels of robot control is not possible without deeply revising the design criteria that are usually underlying the robot control system ([77], [69]). In particular, it is necessary to identify both the tasks of the learning system and the tasks of the robot first and to design an architecture being able to host both the learning and the performance components afterwards.

2.1 Learning Tasks

There are numerous learning tasks that can be identified in the framework of robotics and robot control (see [77] for an overview). Principally, Machine Learning can be applied to support the following actions during robot development and deployment:

- Initial knowledge acquisition and program generation, i.e., initial robot programming and world-model acquisition.
- Action and world knowledge refinement, i.e., acquisition of new knowledge in order to be able to solve new tasks as well as refinement and correction of existing knowledge helping the robot to deal with an uncertain and to adapt to a changing environment.

[1]This means "learnability" in a practical sense and does not refer to the information theoretic definition of learnability as given in, e.g., [151] or [6].

Considering an hierarchical control system, both activities are located on all levels of the hierarchy. Also with respect to a functional decomposition of the tasks assigned to the robot, these activities are part of all functionally independent modules. Therefore, learning must become an integrated part of the robot control system.

3 THREE CASE STUDIES IN ROBOTICS

The three scenarios chosen for B-Learn II exhibit typical characteristics of robotic applications. Especially in compliant motion, heavy real-time constraints are posed upon the performance elements, and stability issues are becoming very important. The monitoring domain mainly deals with the very important problem of error detection and recovery. Here, the main task is to classify from noisy and continuously changing data. In navigation, the focus is especially on the architectural aspects of the application of Machine Learning techniques. Apart from solutions to single problems, the main achievement there is due to the integration of individual learning techniques in a real mobile robot.

3.1 Compliant motion

Compliant motion ([91]) refers to tasks in which a robot manipulates an object while being constrained by contacts with other objects or the environment in general. The motion freedom of the manipulated object is limited and contact forces are generated. The robot has to deal with the constraints and the occuring forces and torques, too.

There are several ways to cope with problems in compliant motion that are caused by uncertainty. The first approach is to reduce these uncertainties by improving the accuracy of the environment and of the robot. However, the best achievable accuracy is limited and the costs for such a solution are extremely high. A second approach is to improve the design of the parts to be assembled, so that the assembly process is simplified. Although this is a good approach, it is not always possible and rarely sufficient. The third approach is to use an intelligently designed passive compliance ([163]). Here, the programmed path of the end-effector is not changed by the contact forces, but it is modified by the deflections of a flexible structure between the manipulated object and the end-effector: the compliance. In active compliance, the robot actively generates motions based on the task and the measured reaction forces ([127], [126], [24]). Two classes of active compliance techniques can be distinguished, namely fine-motion planning and reactive control. Experimental results related to both techniques are shown in section 5.

3.2 Machining and Monitoring

CNC machines play an important role in Flexible Manufacturing Systems: they produce the parts to be assembled. Solving the problem of making products close to the anticipation of defects will considerably contribute to improve the system's overall productivity and product quality. Anticipating accidents, wherever and whenever possible, is a convenient method for preventing intervention. Faults can occur randomly or as a consequence of internal structural degradation. Little hope exists for the first case. However, the second one opens the field for prediction of deviation of quality, enhancing the commonly used methodologies. Prediction of faults can become an important issue as an effective technique aimed at operating in run time and not based on an a posteriori analysis of samples ([5], [136]).

To build a suitable model of the machining process, it is necessary to identify a set of low-level building blocks (process steps). These blocks are commonly called *machining features* ([121]). The machining of a specific workpiece can therefore be understood as the execution of a sequence of these blocks in proper order. The machine's behaviour is mainly determined by the NC instructions it executes. Therefore, it is plausible to expect a characterization of the machining process based on machining features, independent of a specific workpiece. The machined material must be considered in the model because it influences the machine's behaviour, too. For example, machining the same pocket on plastic, brass, or steel produces different sensor patterns, because the necessary cutting forces are different. Hence, the basic components involved in the machining process characterization are a set of machining features, a set of NC instructions, a set of specific machines, and a set of materials.

A characterization of a machining process for each machine, machining feature, and material consists of the specification of the sensorial data pattern captured when the execution of the related NC instructions takes place. Assuming that there exists a machining process characterization for some specific machine, material, and machining feature, it is possible to define a monitoring program for pieces involving those machining features.

In diagnosis and recovery, reasoning about errors is a central activity. Errors can be described on different levels of abstraction, forming a hierarchy or taxonomy, and can be divided into three families: system faults, external exceptions, and execution failures.

Cause-effect relations between these errors can be established. Typically, execution failures are caused by system faults, external exceptions or other past execution failures, although, in general, errors of the three kinds may cause each other. So, if the taxonomy is the vertical organization of errors, the set of causal relations between them, forming causal networks at various levels of abstraction, is its horizontal organization. Determining explanations for detected execution failures can become very complex when errors are propagated. The proposed approach to modeling errors in terms of taxonomic and causal links aims at handling this complexity.

3.3 Transportation and Navigation

In addition to actual manufacturing tasks occuring in a factory, transportation of both raw material and fabricated workpieces is an important issue. Driver-less transport systems are designed to handle these tasks. However, to achieve flexibility and efficiency within the typical constraints of a manufacturing process, simple transport systems are not sufficient, and autonomous mobile robots are to be taken into consideration.

The basic task a mobile robot has to solve is to plan and execute collision-free motions. Moreover, successful navigation and task execution by a mobile robot in a partially unknown environment is strongly dependent on the robot's ability to acquire knowledge about its environment in form of a map or directly object-related information and to use this acquired knowledge for efficient navigation and self-localization ([29], [38]). Therefore, **autonomy** becomes an important aspect ([28]). Generally, an autonomous system should provide several advanced features. First of all, the system should be able to adapt itself to changing conditions in the environment while performing a commanded task. This **indirect** adaptation results in slight changes of the system's behaviour in order to accomplish the given task, even if the a-priori information about the environment is incomplete or incorrect. Second, the robot should be able to acquire knowledge, both about itself as about the environment in general. This acquisition should result in a continuous improvement of the system's capabilities. It is also important to note that an autonomous system should have at least a limited ability to reason about the status and the usability of its own knowledge, in order to determine what kind of information is missing and what knowledge could possibly be acquired by experimentation, retrieved from a user demonstration, or directly be derived from user-given information.

Obviously, these requirements have a direct influence on the design of an architecture for an autonomous system, or, more specifically, for an autonomous mobile robot.

Especially the combination of directly task-related activities, long term strategy, and the incremental generation of knowledge about the world defines a complex behaviour that must be accounted for, both during the design as during the realization phase of such a system. These aspects give a strong evidence for the necessity of **learning components** that must be integrated parts of the system ([119], [34]).

Summarizing, the architecture of an autonomous mobile robot must contain performance components, such as planning modules, but it must also incorporate learning capabilities such as the incremental generation and improvement of motoric skills, or extraction of symbolic knowledge about the world from sensory data. This combination of learning and performance components is a typical feature of an **integrated learning architecture** ([75], [115]). Additionally, the identification of the learning tasks shows the need to employ several different learning techniques. Their integration and cooperation is another important aspect that must be considered and qualifies the developed architecture as an integrated learning architecture, too.

4 APPROACH AND METHODOLOGIES

In this section, the general approaches and learning methodologies being the basic tools used in B-Learn II are described. The applicability of a learning technique with respect to a given task depends on the characteristics of both the task and the learning techniques. For a detailed description of the individual learning tasks, the learning techniques, and the classification of both see [44].

Task characteristics that directly influence the selection of the learning technique are the **kind and amount of learning events**, the **structure of the examples**, and the **amount of available background knowledge.** Other restrictions are given by the **representation and intended use of the target knowledge.** Additionally, the tasks may differ regarding constraints given by the application. **Real-time constraints** might be existing, the **function evaluating the learning result** might require additional knowledge or can directly be derived from the examples.

The selected learning approach is characterized by the **method** (according to the classification scheme developed in [75]), the **representation language**, and, of course, **the learning algorithm.**

4.1 Example preprocessing

All supervised learning algorithms require the existence of a set of preclassified examples. Moreover, these examples are often assumed to be noise-free or even discrete valued. In real-world applications such as robotics, both assumptions do not hold. Therefore, "intelligent" example preprocessing is crucial for successful applications of machine learning algorithms. In particular, two cases must be considered:

1. If a continuous function is to be learned, examples must be preprocessed in order to identify the complexity of the function and the inherent structure of the task.

2. For classification tasks, the important features must be calculated and extracted from the examples prior to feeding the learning algorithm with the features.

4.1.1 Example generation for learning continuous functions

For the initial design of a robot controller (i.e., for the approximation of a continuous function), the existence of examples is the main requirement of the proposed exemplar-based approach. A particular appealing idea is to let a human operator perform the control task in order to generate examples. This approach is in fact an extension of a new programming paradigm, namely *Programming by Human Demonstration* ([31]) respectively *Robot Programming by Demonstration* (RPD, [55], [107]), to the acquisition of elementary skills ([9], [25], [32], [66]).

In the robotics community, the concept of skills can be found throughout a number of different applications. In telerobotics, robots provide the capability to autonomously execute certain operations and relieve the operator from difficult control tasks. These individual capabilities are referred to as skills, the concept itself is also known as shared autonomy ([23], [52], [58], [137], [164]). In robot programming, SKills-Oriented Robot Programming (SKORP, see [7]) relies on the existence of a set of skills as the building blocks of a robot program (see also [37], [55], [69], [107], [131]). The approaches to skill learning are mainly aiming at identifying a control function for a given task. They can be found both in the robotics ([9], [25], [32], [87], [125]) and the machine learning community ([49], [84], [89], [133]). In general, all works share the same principal view on skills as the ability of the robot to safely change the world from a given state to a defined one in the presence of uncertainty, with the individual control functions applied using only initialization data and direct sensorial information at runtime.

For the learning procedure itself, preprocessing of the recorded examples is usually necessary for two reasons. First, there might be differences between the values that can actually be sampled and those that can be used for control afterwards. For instance, it might be necessary (due to a desired sampling frequency) to directly record the robot's joint angles. However, for the analyzation phase and the training following afterwards, it might be better to use the robot's position in cartesian coordinates, hence requiring an a priori transformation step. The second reason is the generally noisy character of the sampled signals, which originates from both uncertainties in the sensors and the only limited "continuous" control that can be provided by a human operator. While the former can be taken care of by, e.g., low pass filtering of the recorded data, the latter requires to eliminate those samples that would become counterproductive during training, such as those created by "breaks" in the robot's motion due to the slowness of the operator.

4.1.2 Feature calculation and extraction

In most machine learning applications, the features are given. In robot applications, sensory data are given. This is what the robot perceives from the environment. However, sensory data are too specific to apply to more than one object perceived during one operation of the robot. Because of this, features have to be calculated from the sensory data. This feature construction is a difficult task. Depending on the calculated features, concept learning may become more or less efficient. Up to the present, there is no theory of representation that tells us which features to construct.

An idea similar to the one followed in B-Learn II was proposed by Wrobel as an answer to symbol grounding problems [162]. In Wrobel's paper, however, learning features from sensory data was modeled as a direct segmentation task of a stream of real-valued sensory input. In contrast, we provide the algorithm with more complex calculations such as, the gradient of two values measured consecutively, or the difference of two angles. However, we do not aim at learning such functions. What is to be learned is the appropriate degree of granularity of the abstraction result, i.e., which function to choose.

In machining, sensors are selected according to the physical nature of the process to be monitored. The complexity of the collected data as well as the respective sampling rate are strongly depending on the selection of low level processing and feature extraction methods ([27], [33], [76]).

A set of sensor data processing procedures and feature extraction methods were developed. After getting knowledge about the sensor data behaviour, the sensor processor model can be implemented. That model includes the representation of the nature of the low level data processing and the feature extraction method. Also the correlation of the resulting features and the process status related to the collected data must be evaluated. After this evaluation decisions can be made about which features are good or not. That leads us to the selection of a correct feature extraction method. In [117] a tool developed for feature extraction evaluation is presented. For the FFT case, it can be determined which subset of harmonics represents the best correlation with the variable class values ([47]). Some results of applying this procedure are also presented in [117] and [10].

Building a map of the environment

Successful autonomous navigation and task execution by a mobile robot in an unknown environment is strongly dependend on the robot's ability to acquire knowledge of its environment in form of a map and to use this acquired knowledge for efficient navigation and self-localization ([29], [38], [92],[118]). Therefore, mobile systems are usually equipped with a set of sensors ([82]), used to build and maintain a sufficiently accurate model of the environment. However, several aspects must be taken care of:

- Due to several factors such as wheel slipping, the knowledge about the robot's position and orientation is inaccurate.
- The sensor inputs are noisy.
- The sensor characteristics (esp. when using ultrasonic sensors) might cause additional uncertainties.
- Positions of objects that are detected in the environment are not known exactly.

Therefore, the task of building a map of the environment can not be seen separately, but should also comprise methods to deal with the uncertainties with respect to the robot's position (see, for instance, [120] and [30] for recent developments in this research area).

4.2 Learning of control functions

Following an example-based approach for learning reactive behaviours in general and control functions mapping sensorial inputs to primitive robot actions in particular, the basic task that has to be solved is to approximate the optimal control as close as possible. The machine learning community developed several methods for approximating continuous numerical functions by means of algorithms which can be trained on data. The most popular methods are the ones based on Neural Networks ([123], [54]). Many network topologies have been investigated and many learning algorithms, both one-step and incremental, have been proposed. In B-Learn II, especially extensions to the classical backpropagation paradigm ([122]), such as the Time-Delay Neural Network ([154]), have been under investigation, as well as networks based on local receptive fields, such as Radial Basis Function networks ([103]).

From the research community in statistics, another family of function approximators based on regression trees ([80]) has been proposed. Recently, a growing interest grew around this alternative approach. However, in comparison to neural networks, regression trees look less suitable to approximate control functions because the current formalisms do not allow incremental training. However, the incremental learning capability is a fundamental requirement in the kind of applications that are considered here. Finally, a third proposal, came from the Fuzzy Set community ([165]), i.e. the fuzzy controllers. Since the initial proposal by Zadeh, a lot of effort has been put in the further development of fuzzy control. Nowadays, fuzzy controllers are a tool which is industrially exploited both in Japan and in U.S, being easy and cheap to apply in comparison to nonlinear controllers offered by classical control theory.

As well as many kinds of neural networks, fuzzy controllers are universal function approximators. Both formalisms share many topological characteristics and some of the training algorithms such as, for instance, the back-propagation algorithm, with neural networks. Also, fuzzy controllers are to be considered a family more than a unique architecture, owing to the many variants they show. For instance, considering the architecture proposed by Berenji ([17]), more similarities to the multi-layer perceptron than to the classical fuzzy logic approach can be found.

In addition to learning control functions for robots, neural networks are also employed to solve the basic problem of regulating the intrinsic parametersetting of a camera. Here, the neural controller's task can be summarized as follows:

1. input data transformation (i.e., acquisition),
2. quality evaluation of the transformed data, and
3. adjustment of the transformation parameters.

Such operations can obviously be iterated until an acceptable average quality of the current image has been obtained. It must be noted that the reduction of complexity performed by applying quality functions to images is actually a necessary step which must be done in order to avoid feeding neural networks directly with images. Concerning the off-line training strategy, the error back-propagation technique has been adopted. Two accelerated implementations have been tested, which feature a good speed-up in comparison to the basic algorithm ([153], [144]).

4.3 Learning and refinement of reactive behaviours

Reinforcement Learning (RL) is a kind of learning between the supervised and the fully unsupervised paradigms. It deals with the problem of learning to do prediction in sequences of interrelated events from a reinforcement signal (reward or punishment) sent back by the world. A major contribution comes from a family of algorithms based on *temporal differences*, called TD(λ) ([140]). The earliest method based on temporal differences is represented by Samuel's checker player ([124]). Later on, similar methods have been used by Holland ([60]) in the Bucket Brigade algorithm, by Sutton, Barto and Anderson ([13]) in the Adaptive Heuristic Critics, and also by Witten ([161]), Booker ([20]), and Hampson ([51]). Moreover, TD methods have also been proposed as a model of *classical conditioning* ([14], [142], [41], [105], [74]).

The most popular algorithm for dealing with the prediction problem is *Q-learning*, developed by Watkins ([158]). Q-learning is strictly related to TD(0) algorithms, since it is based on an implicit Markovian model of the world, too. Other strict relations exist between Q-learning and TD on one hand, and classical Dynamic Programming (DP, [16]) on the other. The fundamental difference is that Q-learning and TD do not build an explicit Markovian model of the prediction process, whereas DP does.

Significant development in the framework of Q-learning are due to Kaelbling ([65]), and to Mahadevan and Connell ([90]), who applied Q-learning to a real robot in order to learn to react in a box pushing task. Other practical developments are due to Lin ([84], [85], [86]), who showed the importance of tutoring in structuring the learning problem into sub-problems in order to speed up a task which is inherently very complex, and to Singh ([134]), who tackles the problem of complexity by combining solutions of simple subtasks, too.

One of the most interesting results in the field of RL application is the *Dyna class of architectures* ([141]). Dyna architectures integrate RL and execution-time planning, and apply the same algorithm alternately to the world and to a previously learned model of the world. Particularly appealing versions are Dyna-Q, that uses Q-learning as a RL method, and Singh's hierarchical Dyna ([132]). Other recent developments in the framework of Dyna architectures have been carried out by Peng and Williams ([114]), who proposed improvements in order to speed up the learning process.

Recently an important variant of the TD and Q-learning algorithms has been proposed by Moore and Atkeson ([104]) with *prioritized sweeping*. This new algorithm proved experimentally to be faster than any other on several test cases. However, a convergency proof as for TD still does not exist.

Further development in the framework of RL comes from authors who work outside the main stream of Q-learning, but that are worth to be mentioned. In particular, Berenji's work on fuzzy controllers, can potentially be very useful: the *GARIC architecture* ([17]) is an evolution of the ARIC architecture proposed by Anderson ([3], [4]) and is based on the learning rule proposed by Sutton, Barto and Anderson ([13]). Berenji proved to be able to learn effective heuristics in difficult control tasks. The GARIC architecture is conceptually stricly related to Dyna and other architectures of the main stream. Finally, in robot navigation, an important contribution comes from Millán ([95], [96], [97], [98], [99]), who proposes an original architecture for combining reactive planning with high level symbolic planning. In conclusion, the Reinforcement Learning community proposes an interesting methodology which could potentially solve the problem of on-line refinement. Obviously (see, for instance, [143]), Reinforcement Learning tackles problems that are also closely related to those of adaptive control (e.g., [61]). Formalisms such as Albus' CMAC ([2]) have already been adopted by the RL community ([83]). Also, Gullapalli's work exploits these similarities ([48]). Millán ([96]) reviews the use of reinforcement learning for robot navigation.

4.4 Concept learning

Most often, if all features defining a concept are true for an object, it is concluded that the object is a member of the concept. This means that a concept is completely determined by its features. Think, for instance, of a representation for the everyday concept "cup". The flat bottom, the concave form, and the handle could be features of the concept "cup", but features alone are not sufficient to define a cup. A particular object could be described by these three features without actually being a cup, because you cannot drink from such a receptacle. Adding additional features to the characterization of a cup will not solve the problem, because, in an infinite number of ways, however, a given receptacle can be such that it is impossible to drink from it. All these ways cannot be excluded by features in the concept and object descriptions. Presumably, for any list of features that a cup cannot have, we could construct an additional exceptional feature which hinders drinking from a receptacle. This is the qualification problem.

The qualification problem indicates that observational features alone are not adequate. What is most important about a cup is that one can drink from it. Drinking is a verifying action for a cup. As many ways as there are to disturb the functionality of a cup, as many exist to preserve its functionality even if the mentioned features are not existing. A concept description should therefore not only consist of perceptual features but also of a typical action. If this action is successful for a particular object, it belongs to the concept. If the action is not successful, it does not belong to the concept. In this way, actions are integrated into concept descriptions and into their application as recognition functions.

Perceptual features describe patterns which are perceived while the robot performs an action. Even features that seem to be purely observational without any link to an action are, in fact, action-oriented. The perception of a flat bottom, for instance, is only possible when looking from a particular angle with respect to the object. Looking straight down upon the receptacle does not allow to determine whether the bottom is flat or not. A perceptual feature (e.g, flat bottom) is constructed with reference to an action (e.g. looking from the side). Action features, in turn, require perceptual features. Actions are represented in terms of the following sensor patterns: what is sensed before a particular action can be performed, what is sensed during successful performance of the action, and what is sensed as the outcome of the action. Hence, perception and action are closely interrelated at all levels of abstraction.

4.5 Learning of diagnostic knowledge

The basic input for the inference engine handling diagnostic knowledge are sensory patterns obtained from the machine or the robot to be monitored. Some patterns can be learned as associated to represent situations of normal or abnormal system behaviour. Depending on the sensing complexity, more general or more specific situations can be addressed. Related knowledge can be organized and made available for monitoring purposes. Learning is associated with the following aspects: training concerning adequate feature selection for the processes to be characterized and generation of real examples associated with good and bad behaviour.

Richard Forsty ([40]) proposes a general structure for a learning system. The four components of Forsty's system are: the performer, the rules, the learner, and the critic. Learning takes place by applying one of the learning paradigms presented in [40], [159], [108], [64], [59], [12], or [18]. They are summarized in the following categories: black box methods, learning structural descriptions, and evolutionary learning strategies. The first category comprises artificial neural networks, the second one consists of symbolic inductive methods and decision trees, the latter uses genetic algorithms.

The general framework for the learning system employed in the machining application consists of the Machining Process, modelled by the available sensors, and the Monitoring & Prognostic System, implementing the overall performer. Because of the complexity of the process, the performer is decomposed into a set of sub-performers called Specific Monitors and the Machine Supervisor. The output of the performers is collected in the facts database *MonitoringResults*. In order to enable a starting point for learning and performance evaluation, the collected sensor data and related features are stored in the database *Sensor Collected Data & Features*. These two databases form the input for the critic and learning processes. The critic and learning processes are human assisted. Here, the human assistance is done both for criticizing some relevant aspects and also to evaluate the learned knowledge. The output of the learning process is stored in the knowledge base *Learned Rules & Characterizations*. This knowledge is organized in order to relate the machining features and respective CNC program segments to the learned rules. The *Learned Rules & Characterizations* knowledge base enables the monitoring program generation. These programs are the rules and related information necessary for the several performers to carry out their job for the specific machining process that is going to take place.

Some of the experiments in the assembly domain used a typically inductive learning algorithm, presented in [57] (an application of this algorithm in the area of robot manipulation of moving objects was also described in this reference). The algorithm is simple, and has the advantage of dealing with numerical data. Numerical training data are preprocessed in order to produce discrimination diagrams (a discrimination diagram of an attribute shows, for each value, the classes that the objects of the training set may have) ([130]). Producing discrimination diagrams from numerical training examples is done in three steps. The first step is to produce histograms for all pairs of classes and attributes. In the second step, each histogram will be approximated to several well known statistical distributions. The distribution that matches the histogram best will be selected. The last step is to apply a rate of significance to the distribution in order to ignore values of the attribute that do not occur significantly in the objects of the considered class.

The output of this inductive learning algorithm is a decision tree ([116], [150]), which is generated in two phases. In the first phase, a minimal set of attributes is determined. The best attributes are those with greatest discrimination power and the smallest computational costs. In the second phase, the tree is recursively created. In each step of the recursion, considering the discrimination power and the importance of the discriminated classes, a new attribute is selected for branching. A leaf node is created when there is only one possible class.

4.6 Learning of planning knowledge

In principle, "planning knowledge" is a term that might refer to a lot of different things, such as layout planning, assembly planning, motion planning, etc. The next two sections are, however, explicitely referring to robot motion planning ([62]), as it occurs in planning of fine motions for assembly robots and in path planning for mobile robots.

4.6.1 Fine motion planning

The assembly strategies associated to the use of active compliance could be generated by a human operator, but they are task-dependent and frequently they imply great skill and effort. Consequently, the interest of an automatic fine motion planner capable of establishing a sequence of movements that ensures assembly task success despite uncertainty becomes evident.

The approach described here assumes that the involved objects are polyhedral and rigid under the forces developed during manipulation. It is valid for a general 6-dof problem, but it has been developed and implemented in detail for planar movements, i.e. two translational and one rotational degrees of freedom, such that polyhedral objects can be considered as polygonal objects moved in a plane parallel to that of the real movement. It is also assumed that the robot is able to work with an impedance position/force control and that the movements are slow enough to make inertias negligible. The plan is built by using information directly available from a CAD database system. The main input data to the planner are the geometric model of the objects, the initial and the final desired configurations of the objects, and the parameters of each uncertainty source.

In order to perform an assembly task with small clearances, the uncertainty affecting the parameters and variables describing objects or objects' behaviour in the real world must be considered. Uncertainty has been modeled in different ways given rise, for instance, to probabilistic and worst case models. In this work, a worst case model has been considered, defining the *uncertainty* of a parameter or variable as the domain containing all possible actual values for an observed value subject to deviations; thus, the actual value may be any inside the uncertainty domain with the same probability. The following table enumerates the uncertainty sources that affect an assembly task.

Geometric	Manufacturing tolerances	(a) object shape and size
	Object observation	(b) point position measurement (c) configuration determination
	Robot related	(d) slipping of the object in the gripper (e) end effector position and orientation
Force measurement		(f) force/torque sensor
Velocity		(g) robot control

Deviations (a) through (d) determine the set of configurations of possible contact between the manipulated object and the environment. By merging all these uncertainties it is possible to establish an uncertainty domain, CUr, of possible configurations for the occurrence of each basic contact. Because the robot itself is used as a sensor to observe the manipulated object configuration, uncertainty (e) implies a set of possible sensed configurations for each real configuration. Considering all the uncertainties (a) through (e) (i.e. by adding uncertainty (e) to CUr) it is possible to establish an uncertainty domain of possible sensed configurations, CU, for each basic contact. The union of the sets CU of all possible basic contacts results in a set CcI in which a contact could be detected. CcI divides the Configuration Space, $\mathcal{C}$, giving rise to two other sets: ClI, the guaranteed free space configurations set, and CiI, the set of impossible configurations. Movements through ClI can be considered as gross-motion and movements through CcI implies fine-motion.

For polygonal objects moved in a plane, Basañez and Suárez [15] have modelled all the geometric uncertainty sources and have merged them to obtain the sets CUr and CU. An extension of this work was detailed in [111].

Concerning generalized reaction forces, deviations (a) through (d) determine the real shape of the $\mathcal{C}$-surfaces and therefore the set, FUr, of possible generalized reaction forces for each basic contact ([39]). When reaction forces are observed, the uncertainty of the corresponding sensor must be considered. By adding uncertainty (f) to FUr it is possible to obtain the set of all the possible measured generalized reaction forces, FU, for each basic contact.

For polygonal objects moved in a plane Suárez, Basañez and Rosell ([139]) have developed a procedure to determine the sets FUr and FU using dual representation of forces ([22]). Finally, uncertainty (g) affects the robot movements, and therefore it must be taken into account when the movements to perform the assembly are determined.

4.6.2 Mobile robot path planning

To speed up planning and to be able to take care of different evaluation criteria during the planning process, the planning process takes place on two levels. While a geometrical planner is responsible for short range planning on the base of known geometrical features of the environment, a topological planner operates on a graph representing existing subpaths in the environment. By means of a task-dependent cost function, the topological planner estimates the best possible path. This cost function is subject to adaptation due to the robot's experience and reflects random disturbances introduced in the environment as well as a simple exploration strategy to deal with mid-term environmental changes.

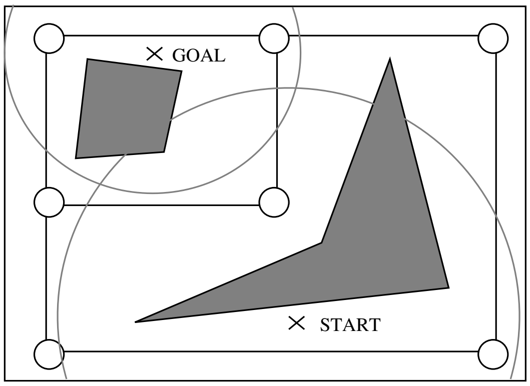

Figure 1: *Example of a topological map with areas associated to locations in the environment.*

Topological planning

The idea of introducing topological graphs (see figure 1) as a more abstract way to represent free spaces and existing passages in the environment is very appealing. The graph representation allows easy access to the existing information, and algorithms known from graph theory are directly applicable to the problem of finding the "best" path. However, an efficient application of topological maps in a real world requires that the following aspects are taken into account:

1. The topological map itself might be changing due to changes in the environment.
2. Even if the map itself does not change between the execution of two missions, it is possible that temporary changes in the environment (e.g., a passage being blocked by a human) require to use different paths to accomplish the same task.
3. Most of the time, the robot's start and goal position will not directly be located on the topological map. The selection of good or even optimal points to enter and leave the associated graph is not straightforward.
4. The cost of moving along an edge of the graph cannot completely be computed a-priori, since it depends heavily on dynamic features of the environment that influence, for instance, the security of an edge, or the average speed of the robot on the corresponding subpaths. Moreover, global constraints can be imposed on the whole system, changing the criteria an "optimal" path has to fulfill.

The problem of finding good entry points to move into and good exit points to leave the topological map is tackled by incrementally generating a set of possibly overlapping areas and connections between these areas. For each pair of areas, the best path (entry point, topological path, exit point) is known. Individual areas are enlarged or reduced according to the experience of the robot. A detailed description of the algorithm, which is based on incremental methods to build classification structures ([42]), can be found in [155].

To deal with temporary changes in the environment, the topological planner realizes a simple exploration strategy, which allows edges that have been found blocked during operation to be used again in the planning process, if the blocking doesn't occur too often. The calculation of the cost function is based on a cost model which includes both static (such as the length of the edge) and dynamic (such as density of obstacles, security of the edge, special actions that might be required) costs. The dynamic costs are known after the robot has passed an edge. This information is used to enhance the model of the cost function and to optimize the prediction of the edge costs at planning time. Since both the history of the cost function and the function's derivative is important in a complex environment, an appropriate learning method is temporal difference learning ([140]).

The planning strategy itself can be described as follow: After good entry and exit points have been determined, generate temporary connections between the exact start and goal position and the entry respectively exit points. Predict the costs of all edges of the topological map and run a branch and bound algorithm to obtain a path with (estimated) least overall costs.

Geometrical planning

Paths outside the topological map such as the paths to enter and leave the graph must be planned geometrically. This geometrical planning is restricted to an area which is slightly larger than the area that can be perceived by the robot's ultrasonic sensors.

The local model which is used by the geometrical planner integrates a-priori available information (such as an incomplete description of the environment) with recently acquired sensorial inputs. The planning algorithm itself is based on the idea of potential distance fields ([156]) and is described in more detail in [155].

The integration of the topological and the geometrical planner (and the reactive module) realizes an adaptive behaviour on several levels. Especially the "experience" of the geometrical planner is not only altering the structure of the topological map (which might as well be seen as some sort of preference criterion), but also the prediction of edge costs and the set of areas that is defined over the whole working space of the robot. Moreover, also the activity of the reactive module, which is primarily responsible for guaranteeing collision-free motions, provides another contribution to the estimation of costs for individual (single-edge) movements, thus adapting the cost estimation means also to account for the robot's low-level capabilities. On the other hand, both the topological (via an a posteriori comparison of static costs such as the geometrical length of the edge and dynamic costs such as the required movement) and the geometrical planner (by detecting suboptimal short range movements) can explicitely request the generation of a new behaviour, expressed as a new situation-action rule, from the reactive module.

5 EXPERIMENTAL RESULTS

5.1 Test beds

Since the emphasis in B-Learn II is on the application of Machine Learning techniques to real robots and real machines, complex test beds featuring real-world conditions have been developed. They are shortly described in this section.

5.1.1 Robot Manipulators

For the experiments carried out in B-Learn II, four robot manipulators are available. At the Katholieke Universiteit Leuven[2], a KUKA-IR 361 is installed as a 6 degree of freedom force-torque and velocity controlled robot. Control programs are developed on a workstation and downloaded to a transputer cluster directly linked to the robot. The robot used at the University of Karlsruhe[3] is a standard Puma 260 manipulator (made by Staeubli Unimation). Providing a maximum load of 9 N, this robot is the smallest available industrial robot featuring six degrees of freedom, hence an ideal choice to perform experiments that involve direct interaction with human users. Is is controlled via a customized Unimation controller linked to workstations via Ethernet. The experiments taking place at the Instituto de Cibernética de Barcelona[4] are carried out on a Puma 560 (Unimation) with a Mark II controller. This robot is linked to a workstation via a serial connection and equipped with proximity and tactile sensors in addition to a force-/torque sensor. For the monitoring application, the Universidade Nova de Lisboa[5] is using a SCARA robot with four degrees of freedom.

5.1.2 Machinery

The experimental setup in the machining application consists of a Dendford STARTURN 4 lathe machine and a Dendford STARMILL milling machine[6]. These machines are semiprofessional NC machines, featuring a 0.5 HP DC permanent magnet spin-

[2]Contact: **Marnix Nuttin** (nuttin@mech.kuleuven.ac.be)
[3]Contact: **Michael Kaiser** (kaiser@ira.uka.de)
[4]Contact: **Raul Suárez** (suarez@ic.upc.es)
[5]Contact: **Luis Camarinha-Matos** (cam@fct.unl.pt)
[6]Contact: **Luis Camarinha-Matos** (cam@fct.unl.pt)

dle motor with rotating speeds of 0 to 2000 rpm. The mechanical resolution of the axis is 0.01 mm, with a feed rate range of 0-1200 mm per minute. These machines are equipped with additional sensors to monitor the machines' working conditions and the resulting workpiece.

5.1.3 Mobile robots

At the University of Karlsruhe[7], the mobile robot PRIAMOS ([36]) is used as a platform for experiments on perception, navigation, and on the application of both subsymbolic and symbolic learning techniques for navigation tasks. PRIAMOS is a mobile system with three degrees of freedom, i.e. motion in longitudinal and transverse direction and rotation around the center of the vehicle. This is accomplished by the use of four Mecanum wheels, each one driven separately. Currently, the robot is equipped with 24 ultrasonic sensors, of which three are mounted at each of the robot's sides and three at each of the robot's edges. The sensor control and processing system is able to process the input of all 24 sensors up to five times per second. Other sensors, such as a stereo vision system ([157]), can be mounted on the upper cover plate using a flexible mounting system. The Instituto de Cibernética de Barcelona[8] uses a commercial NOMAD 200 robot for experiments. This robot has a radius of 9 inches and a height of 30 inches and three independent motors. The first motor translates the three wheels of the robot together. The second one steers the wheels together. The third motor rotates the turret of the robot. The robot can only translate along the forward and backward directions along which the three wheels are aligned. The robot has a zero gyro-radius, i.e. it can steer around its center. The version of the Nomad 200 in use has three sensory systems, namely tactile, infrared, and ultrasonic.

5.1.4 The visual sensing system

The visual sensing subsystem has been developed and tested at the Universita di Genova[9] based on a set of indoor images acquired within a computer vision laboratory. The perceived scenes included various objects, such as doors, computing and disk units, windows, etc. The lighting conditions have been maintained quite uniform. The training sets (for both the 'whole-image' and the 'window' mode) have been built by associating each image to the set of regulation parameters considered optimal by a human expert. The total amount of training couples is larger than 900. The camera is equipped with a motorized lens system and it is arranged for an external control of both electronic gain and black level. The motorized lens system allows the regulation of the focal length (zoom), the focusing distance, and the aperture diameter (iris).

The control of the objective parameters is accomplished by means of an RS 232 interface. There are three engines for the activation of the parameters, three position receptors for the control of the exact positioning, and a controller for communications with the host computer. Regulation is entirely numeric and programmable.

5.2 Data preprocessing

5.2.1 Example preprocessing and training data generation[10]

Figure 2 shows the forces F_x, F_y, F_z that have been recorded during an insertion operation performed by a human operator using a 6D joystick and on a KUKA-IR 361,

[7]Contact: **Michael Kaiser** (kaiser@ira.uka.de)
[8]Contact: **José del R. Millán** (jose.millan@cen.jrc.it)
[9]Contact: **Francesco de Natale** (dede@dibe.unige.it)
[10]Original references: [66], [68], [69]

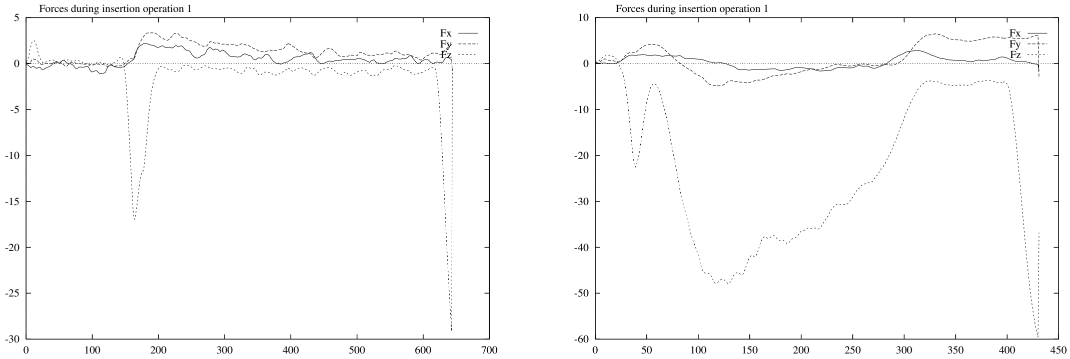

Figure 2: *Forces recorded during an insertion operation. Left: recorded from a human operator. Right: recorded from a working controller.*

using a taskframe controller ([126]), respectively. In general, an example $E = (\vec{x}, \vec{y})^T$ consists of a sequence of measured inputs $\vec{x}_1, \ldots, \vec{x}_T, T > 0$ and the corresponding measured outputs $\vec{y}_1, \ldots, \vec{y}_T$ that are representing the commands applied to the robot. $s := dim(\vec{x})$ is the number of active sensors during recording time, and $a := dim(\vec{y})$ the number of measurable outputs. However, in general it cannot be assumed that the measured inputs at time t contain sufficient information to generate the desired output. Sometimes the history of the inputs must be taken into account. The analysis that aims at finding these complex dependencies takes place during the segmentation of the example. The meaning of the example, i.e. the knowledge about what the robot has actually been doing while the example has been recorded, is known to the symbolic system components, since they are communicating with the user, who acts as the supervising entity.

Segmentation

Generally, an operation performed on the level of control might not only require continuous control but also the consideration of the context as given by the history of the situations that have already been encountered by the robot. This context can be accounted for in several ways. One possibility is to explicitely handle the history of situations by designing skills that do not only realize a mapping $Sit \mapsto Act$, but a mapping $Sit^T \mapsto Act$, thus making the action to be taken dependent on the last T situations. This solution results in high dimensional $(s \times T)$ input spaces, complex mappings, and great difficulties in finding appropriate training sets ([67]).

Another solution is the segmentation of the operation into different **phases**. Based on an example $E = (\vec{x}, \vec{y})^T$ a phase is defined as an interval $P = [t_s \geq 0, t_e \leq T]$ such that $\forall t_1, t_2 \in P : \vec{x}_{t_1} =_s \vec{x}_{t_2} \rightarrow \vec{y}_{t_1} =_a \vec{y}_{t_2}$, where $=_s$ and $=_a$ define a measure of similarity for the situations and the actions, respectively[11]. Thus, during each of the phases the action can be considered to be dependent on the current situation only, i.e., the control task can considered to be Markovian. The advantage of this approach is that the individual skills as well as the input space are less complex. Thus it becomes easier to generate a sufficient set of examples. Also, this kind of segmentation results in structural information that is important for building a symbolic description of the developed skill. On the other hand, this advantage is reduced by the necessity to detect the phase in order to be able to choose the appropriate skill.

A second possibility to reduce the complexity of the control function c representing the rule set associated to each skill is to realize each mapping $c : Sit \mapsto Act$, corresponding to $\bar{c} : \mathrm{IR}^s \mapsto \mathrm{IR}^a$, by a set of functions $c_i : \mathrm{IR}^s \mapsto \mathrm{IR}$ such that $\forall s \in \mathrm{IR}^s$:

[11] Examples are generated by sampling the inputs and outputs at discrete points $(t_1, \ldots, t_T)$ in time. When relating an example to a phase, only those points are considered.

$\bar{c}(s) = \vec{y} = (y_1, \ldots, y_a) = (c_1(s), \ldots, c_a(s))$. Considering the application of robot control, this means that each degree of freedom is controlled individually.

Training data generation

Apart from scaling the sampled input and output values to a range that is acceptable for the selected learning technique (e.g., to $[0, 1]$), the generation of an appropriate training set requires the use of data recorded from several examples. Since each of the examples consists of a sequence of phases, the task is to find and merge corresponding phases. *Correspondancy* between phases comprises two different aspects. On the one hand, skills generated from corresponding phases should essentially realize the same functionality, i.e., "do the same." On the other hand, the combination of both phases should again be a phase (in the sense defined above) in order not to violate the Markov assumption underlying the individual skills.
Therefore, for each pair $p_1 = [t_{11}, t_{12}]$ and $p_2 = [t_{21}, t_{22}]$ of phases belonging to examples $E_1 = (\vec{x_1}, \vec{y_1})^{T_1}$ and $E_2 = (\vec{x_2}, \vec{y_2})^{T_2}$, respectively, it must be checked whether the local goal states (i.e., the situations at the end of the phase) are the same. If this is the case, and $\forall t_1 \in p_1, t_2 \in p_2 : \vec{x_1}_{t_1} =_s \vec{x_2}_{t_2} \rightarrow \vec{y_1}_{t_1} =_a \vec{y_2}_{t_2}$, the phases can be merged to a single one, i.e., they can serve as training data for the same skill.

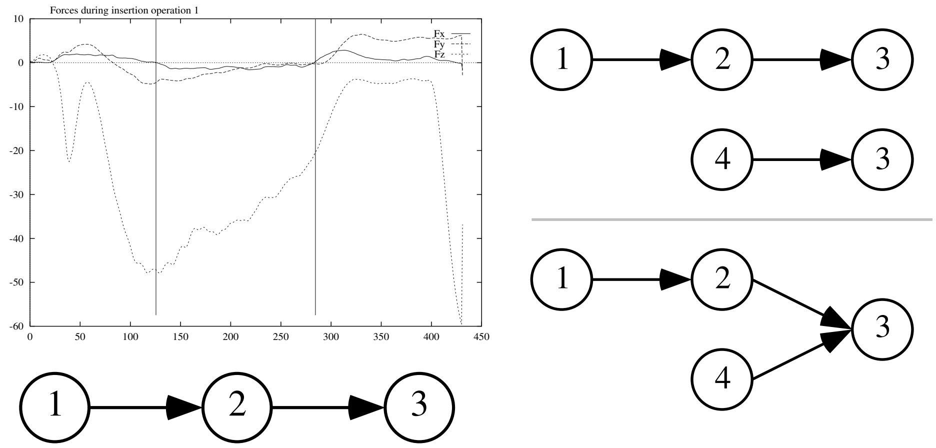

Figure 3: *Mapping of an example to a sequence of states and merging of sequences.*

In order to generate the logical structure of the controller, i.e., the skill model, it is necessary to map the examples to a sequence of states, using the already existing segmentation information. Since it is possible that phases occur several times in a single example (this might indicate a loop that must be executed several times until a certain condition is met), an intra example check similar to the one described above for corresponding phases must be performed (see also [37]).

5.2.2 Feature calculation and extraction[12]

The data used for feature learning are gathered by the vehicle PRIAMOS of the University of Karlsruhe while moving through a room. The vehicle is equipped with 24 sonar sensors, all of which are located at the same height all around the vehicle. The aim is to enable the navigation system to perform high-level tasks such as "pass through the door, turn left, move to the cupboard" in various environments. This requires learning of operational concepts such as "pass through door" from several training traces. Each training trace consists of the following data for each point of time and each sensor:

[12] Original references: [72], [106]

- trace number and point of time; for identification of the measurement,
- robot position in world coordinates,
- sensor number, its position and orientation; for identification of the sensor,
- measured distance,
- position of the sensed point in world coordinates,
- object and edge number of what is sensed.

Currently, 28 traces, each with at least 21 and up to 54 samples, are used. Thus, there are 17472 measurements as examples for the learning system. Most paths are movements through a door with different distances from the door frames, different directions relative to the door frames, and with or without a cupboard at the wall close to the door. Particular constellations of edges, such as two walls being linked by a right angle, are gathered. These constellations are called "line", "jump", "concave", and "convex", respectively.

Building a map of the environment[13]

Principally, there exist two approaches for modeling a-priori unknown objects. On the one hand, parameterized models are used where the modeling task is reduced to an estimation of the object parameters. In a certain sense, this is similar to the object localization done in PRIAMOS. However, since no a-priori object knowledge is available, the full uncertainty of the sensor measurement and the robot's position error has to be taken into account. The second group of approaches is based on two- or threedimensional grids. Cells in a grid are directly related to parts of the environment, and sensor measurements can directly be put in the map. However, the mapping measurement $\mapsto$ cell depends on the sensor charateristics. Since both approaches have their advantages, a combination is used in PRIAMOS. On a low level, grids are used to collect sensor measurements. Feature extraction is performed on the information stored in the grids, the results can serve as a starting point for object parameter estimation. In order to be able to deal with dynamic objects and to reduce the complexity of the grid-based representation, we introduce two enhancements with respect to the traditional approaches. First, we treat the grids locally, i.e. the grids are assigned to a detected object and can be moved according to the object's movements. Second, positive (holding evidence for the non-emptyness of a cell) and negative (holding evidence for the emptyness of a cell) grids are stored separately, helping to detect the motion of objects. Additionally, it is necessary to be able to combine grids that have cells in common. This **fusion of grids** is done on the base of a standard OR operator. The fusion of positive and negative grids is done separately. Afterwards, a new grid is computed by adding the cell values of the corresponding positive and negative grid.

Feature extraction in Machining[14]

From the CNC laboratory facilities available at the Universidade Nova de Lisboa, the lathe STARTURN 4 was selected. The experiments undertaken are aiming at assessing the validity of the adopted approach of correlating the sensorial data patterns to the NC command under execution and, at evaluating the methods for processing the sensorial inputs and for the extraction of features. Three sensors where installed in the lathe, in order to record sound, axis vibrations, and machine main current consumption. They were developed inhouse using standard transducers available on the market.

[13] Original references: [81], [70]
[14] Original references: [117], [11]

Three CNC programs were written in order to activate the machine in the following situations: spindle running at different speeds (100 to 2000 rpm, increments of 50 rpm) without cutting, axis movements (10 to 10 mm/min, increments of 10 mm/min) without cutting and finally a machining program (machining of a brass screw). During the execution of these CNC programs, data from each sensor were collected and stored in a database. All sensorial data were labeled with the corresponding NC instruction under execution.

The FFT transform of each sampled data block was computed off line. The resulting power spectrum was stored in a database. The feature extraction methods were based on a toolbox for analysis and visualization of sensor data ([117]), which was especially developed for this kind of application. Basically, the toolbox permits the analysis of the processed sensorial data in order to identify the most relevant characteristics for discrimination. Three different algorithms are available for feature evaluation: UniVar, M-Covar and MinErr. The best performing algorithm for this kind of data is the MinErr (it is a variant of the Sequential Forward Search (SFS) algorithm described in [33]). The set of selected features is used as a base for the construction of any kind of identifier. As a result the relevant frequency harmonics that present the best correlation to the set of classes (a class corresponds to an NC instruction) are determined. This way the number of features as well as the harmonics that should be used to build the feature vector are found. Feature vectors of only 10 to 20 harmonics give surprisingly good identification values. That means that the generation of identifiers and the respective training can be speeded up.

Similar procedures were done for the data related to the remaining sensors: machining sound, tool carriage vibrations in the X and Z directions, and the main electrical current. After examining the results corresponding to the data collected when the machine was exercised (not cutting), it was found that for this specific machine

- The main electrical current sensor is the one that presents the best identification figures for spindle speed identification.

- As it was expected, the X and Z accelerometers present the best result figures for the axis moving speed identification. For the case of the data collected when machining the piece, the sensors still present the same previously described behaviour, but different harmonics must be used. That means that the sensor data have changed their characteristics. These results meet the expectations. In fact, the cutting process brings new information to the sensors. The cutting forces influence all sensors: the sound, the vibrations of axis and also the power consumption.

Those results lead to the conclusion that the approach based on context sensor data interpretation is plausible and can be used for real time monitoring.

5.3 Learning of control functions

In order to verify the possibility of synthesizing a controller from examples of correct behaviour, an experiment has been performed using both fuzzy controllers and TDNNs. The test case has been taken from a real robotic application where a KUKA-IR 361 was requested to perform the classical peg-into-hole task. The robot was already operating controlled by a PD-controller using force and torque as inputs and producing cartesian velocities as output (([126], [128]).

Hence, each example consisted of six inputs (F_x, F_y, F_z, T_x, T_y and T_z) and six outputs (V_x, V_y, V_z, ω_x, ω_y and ω_z). The learning task itself was posed as in the following:

1. The traces of the signals in input and output corresponding to seven insertion events have been recorded.

2. Three insertion events have been selected for supplying the learning set to induce a control procedure.

3. The obtained control procedures have been tested on the other four events in order to assess the performance achieved.

5.3.1 Learning a Control Procedure using TDNNs[15]

The difficulty usually encountered in applying a neural network to a learning task is the determination of a proper network structure. In the case of a TDNN, this means the proper number of hidden units as well as the number of delays associated to each neuron. There is no theory supporting this choice and only the empirical experimentation with different configurations can lead to discover a proper setting. Moreover, a second dilemma to be resolved is whether to use a unique network with many outputs or to use an independent network for each of the output signals. The first solution could lead to more compact solutions and in principle is more attractive. Nevertheless, the experiments showed that it was more easy to achieve good results adopting the second strategy.

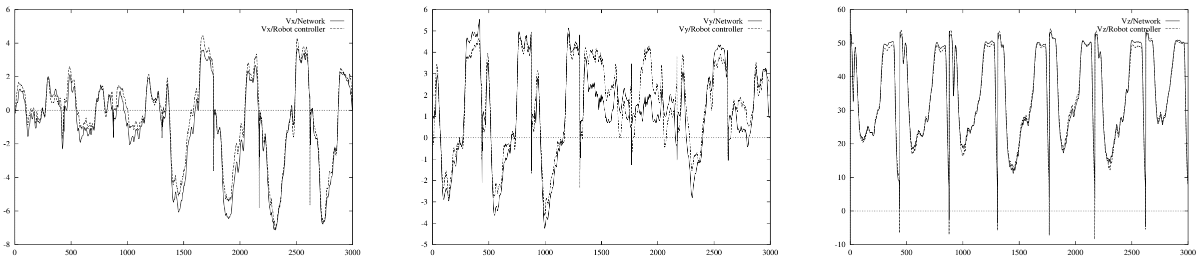

Figure 4: *X,Y, and Z-velocity (test set) of network vs. desired output*

The experimentation itself has been performed starting with simple topologies (i.e., with few hidden units and few delay units), followed by increasing the complexity as long as a significant increase in the accuracy was obtained. The same approach has been used trying to synthesize a multi-output network and also a set of six independent networks (one for each output). The results are reported in table 1, see also figure 4.

5.3.2 Learning a Fuzzy Controller using SMART+[16]

The synthesis of a Fuzzy Controller from examples has been posed as a problem of learning concept descriptions from examples, which can be solved using a symbolic inductive algorithm. In the specific case, SMART+ ([21]) has been used. Looking more closely at the structure of a fuzzy controller, it can be found that the problem is to learn the fuzzy sets of the first layer, the fuzzy sets of the third layer and the set of rules representing the mapping from the first layer to the third one.

In the experiments described here, the number and the size of the fuzzy sets in the third layer were chosen a priori. Afterwards, SMART+ had to learn both the rules and the fuzzy sets in the first layer. However, the method for defining the third layer fuzzy sets consisted of simply subdividing the range of the output signal into n equally spaced intervals each one represented by a triangular fuzzy set, thus the number n of subdivision was actually the only critical choice a human expert was asked to do. Afterwards, SMART+ was able to determine the remaining pieces of knowledge. Also in this case it was necessary to decide whether to have a unique controller with many outputs

[15] Original reference: [67]

[16] Original references: [43], [46], [109], [110]

Method	Signal	Complexity	E_{max}	E_{avg} [%]	$Signal_{max}$	$Signal_{min}$
TDNNs	V_x	28	0.882	2.25	4.4	-7.1
	V_y	28	0.539	2.02	4.9	-3.6
	V_z	28	2.469	1.8	54.2	-8.4
	ω_x	28	0.008	6.6	0.036	-0.06
	ω_y	28	0.0039	9.7	0.054	-0.015
SMART+	V_x	46	1.20	1.49	4.4	-7.1
	V_y	85	2.12	1.27	4.9	-3.6
	V_z	58	54.77	2.12	54.2	-8.4
	ω_x	135	0.0566	4.47	0.036	-0.06
	ω_y	121	0.0437	3.45	0.054	-0.015

Table 1: *Comparison of the performance of the controller synthesized using 5 independent TDNNs and 5 independent FCs induced by SMART+. The component of the angular velocity along z, ω_z, doesn't appear because a round peg has been used. The complexity is evaluated as the total number of neurons for TDNNs and the number of rules for FCs.*

or six independent controllers. The experimentation has been done only for this last case, and the results obtained are described in table 1 as well.

5.3.3 The Comparison[17]

By comparing the best results obtained using TDNNs and Fuzzy Controller synthesized by means of SMART+ (see table 1) we can see a substantial similarity in the performance with respect to the accuracy in approximation the control function. Therefore, according to this criterion, no specific reason for preferring one approach over the other can be indicated. On the other hand, more substantial differences can be distinguished by considering the cost and the difficulty of applying the training process, and by considering the peculiarities of the two approximation techniques. Basically, the learning machinery necessary for applying the back propagation algorithm to the TDNN is uncomparably simpler than the one required by symbolic algorithms such as SMART+. Moreover, the computational cost for 2,000,000 of learning steps for TDNNs is lower than the one for applying SMART+. In fact, the time required by SMART+ was in average a couple of hours on a Sparc 10 against half an hour for training a TDNN. Currenty, Radial Basis Function networks ([103]) are under investigation. The results achieved so far show that, given a sufficient number of cluster nodes, this type of network is able to meet the approximation accuracy of both the automatically generated Fuzzy Controller and the TDNN, while requiring less computational resources.

5.3.4 Control of the automatic vision system[18]

The automatic vision system (AVS) works as a chain of four processing modules ranging from acquisition to symbolic description. Since learning capabilities are currently exploited only in the camera module, the work of the AVS consists of a loop of actions performed in this first module (image acquisition, image quality evaluation, adjustment of the acquisition parameters). If the quality of the acquired image is satisfactory, the loop is exited and the data is passed to the upper modules, which generate the symbolic description. The training set has been built associating quality factors and optimal acquisition parameters for more than 950 images acquired inside the Robot Vision Laboratory

[17] Original reference: [45]
[18] Original references: [102], [101]

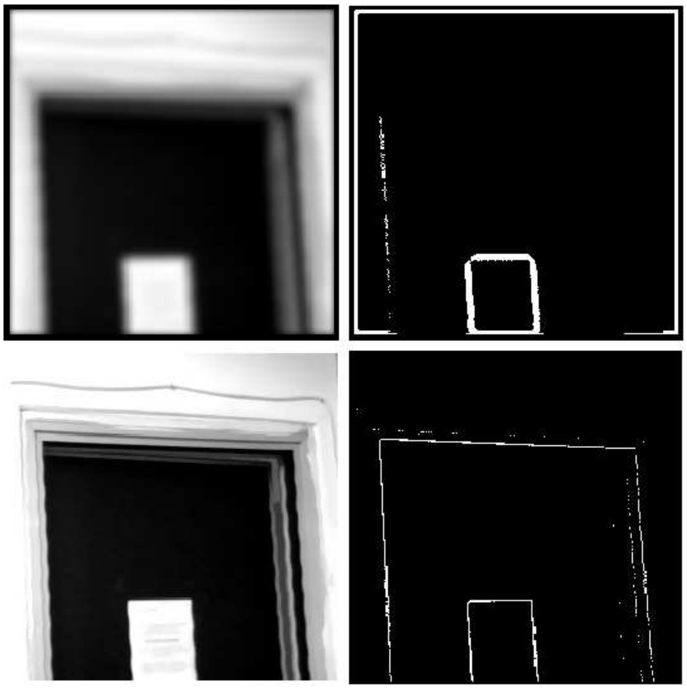

Figure 5: *Upper part of a closed door: a) image acquired with random regulation; b) resulting edges after processing; c) image acquired after regulation loop (one-shot); d) final processed data.*

	number of tests	number of cycles	% of total
success	84	1	89.4
	3	2	3.2
fail	7	4	7.4

Table 2: *Success rate of the automatic vision system.*

of UGE (see figure 5). The ANN has been fed with these few features instead of pixel level information which implies a prohibitive implementation. When the AVS is requested an image, an evaluation-regulation loop is again performed, using the trained network. Experimental results show that the optimal acquisition parameters are achieved in a few iterations of the loop: in the 89.4% of the cases one iteration only is needed (one-shot), two cycles are necessary in the 3.2%, whereas only in 7.4% of cases the ANN could not give good results after 4 cycles. These last cases correspond to situations where there was nearly no light in the scene, so a satisfactory acquisition is very difficult.

Concerning the processing time, which must be carefully taken into consideration particularly in the context of the activation of the system under direct request, a processing sequence beginning with the scene acquisition until the edge detection, requires about 2.5 minutes. Even if such a result seems modest, a few observation should be made: 1. No specific hardware has been used, 2. The current platform (SUN SparcStation 2) performance is not leading-edge. 3. The acquisition system (camera and frame-grabber) is connected to a remote machine, which implies a communication overhead for all image transfers to the host computer.

	acquisition & tuning	preprocessing	edge extraction	edge detection	total
min	70s	20s	6s	32s	2m08s
max	83s	38s	12s	46s	2m59s
average	77s	31s	8s	35s	2m31s

Table 3: *Time required for invididual processing modules of the AVS.*

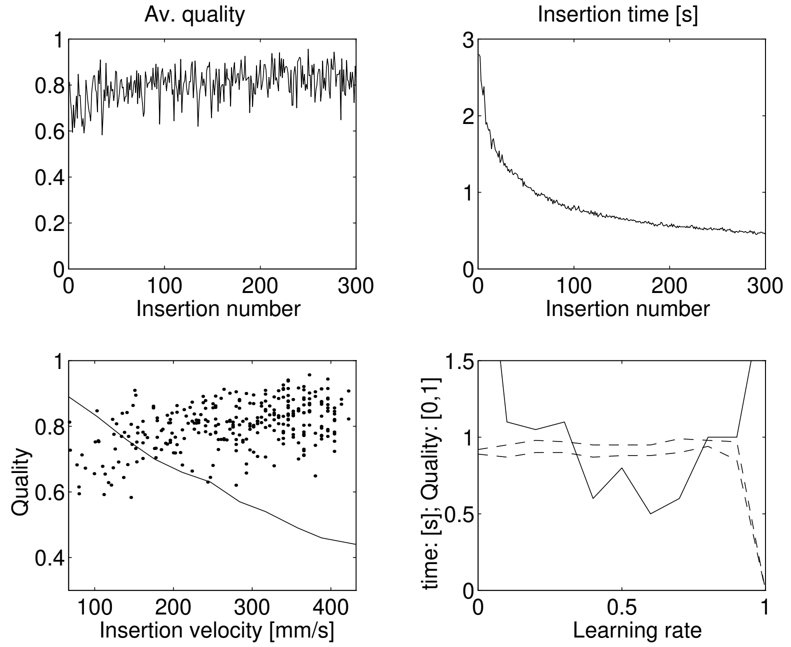

Figure 6: *Simulation results. (a) Top left: evolution of Q_1 over consecutive insertion operations. Q_1 is a dimensionless quality measure for the contact forces occuring during an insertion. The contact forces decrease. (b) Top right: evolution of insertion time (in seconds). After 300 insertion operations, the insertion time has decreased with a factor 3. (c) Bottom left: Q_1 as a function of insertion velocity for (full line) the task frame controller and for (dots) the RL controller. (d) Bottom right: (full line) insertion time of the 300^{th} insertion and (dashed line) quality range of Q_1 after convergence; both as a function of the learning rate of the action network.*

5.4 Learning and refinement of reactive behaviours[19]

Reinforcement Learning as described in section 4.3 is used in different contexts in B-Learn II. In the area of **compliant motion**, the peg-into-hole task is selected as the case to be studied. The optimal controller for this task has been proven to be non-linear ([8]).

In addition to the off-line generation of controllers described in section 5.3, an on-line learning connectionist controller has been developed to work with a contact force simulator. The applied learning algorithms are based on the REINFORCE family of algorithms described in [160]. In the simulation (see [129]), which includes sensor noise and initial position uncertainty, the manipulator works in 3D with 6 degrees of freedom. The learning controller must find a good control relationship between the measured contact forces and the controlled cartesian velocities. In a first phase, the learning controller is trained supervised by a suboptimal taskframe controller, developed at the PMA laboratory,

[19] Original references: [113], [95], [99]

K. U. Leuven ([126]). Afterwards, the reinforcement phase follows. The learning controller is able to increase the insertion speed without increasing the contact forces. The results are shown in figure 6. Also, experimental studies were undertaken to understand the effect of the learning parameters. For example, the figure shows the effect of changing the learning rate of the action network on the resulting performance, too.

To **autonomous robots**, reinforcement connectionist learning brings four benefits. First, this kind of learning robot can improve its performance continuously and can adapt itself to new environments. Second, the connectionist network does not need to represent explicitly all possible situation-action rules as it shows good generalization capabilities. Third, connectionist networks have been shown to deal well with noisy input data, a capability which is essential for any robot working upon information close to the raw sensory data. Fourth, connectionist learning rules are well suited to on-line and real-time learning.

ICB's robot TESEO exhibits all these advantages. In addition, it also overcomes three critical limitations of basic reinforcement connectionist learning that prevent its application to autonomous robots operating in the real world. The first and most important limitation is that reinforcement learning might require an extremely long time. The main reason is that it is hard to determine rapidly promising parts of the action space where to search for suitable reactions. The second limitation regards the robot's behaviour during learning. Learning robots should be operational at any moment and, most critically, they should avoid catastrophic failures such as collisions. Finally, the third limitation concerns the inability of "monolithic" connectionist networks —i.e., networks where the knowledge is distributively codified over all the weights— to support incremental learning. In this kind of standard networks, learning a new rule (or tuning an existing one) could degrade the knowledge already acquired for other situations.

TESEO uses three main ideas to overcome the above-mentioned limitations. First, instead of learning from scratch, TESEO utilizes a fixed set of *basic reflexes* every time its neural network fails to generalize correctly its previous experience to the current situation. The neural network associates the selected reflex with the perceived situation in one step. This new reaction rule will be tuned subsequently through reinforcement learning. Second, TESEO automatically builds *modular network* with a suitable structure and size. Each module codifies a *consistent set of reaction rules*. That is, each module's rules map similar sensory inputs into similar actions and, in addition, they have similar long-term consequences. This procedure guarantees that improvements on a module will not negatively alter other unrelated modules. Finally, TESEO explores the action space by concentrating the search around the best actions currently known. This exploration technique allows TESEO to avoid experiencing irrelevant actions and to minimize the risk of collisions.

The environment where TESEO has to perform its missions consists of a corridor with offices at both sides. In one of the experiments TESEO is asked to go from inside an office to a point in the corridor. The first time it tries to reach the goal it relies almost all the time on the basic reflexes which make TESEO follow walls and move around obstacles. In the first trial, TESEO enters into a dead-end section of the office (but it does not get trapped into it) and even it collides against the door frame because its sensors were not able to detect it. Collisions happened because the frame of the door is relatively thin and the incident angles of the rays drawn from the sensors were too large resulting in specular reflections.

Thus this task offers three learning opportunities to TESEO. The first and simplest one is to tune slightly certain sections of the trajectory generated by the basic reflexes. The second opportunity consists of avoiding dead-ends or, in general, of not following "wrong" walls. The third and most critical opportunity arises in very particular occasions where the robot collides because its sensors cannot detect obstacles.

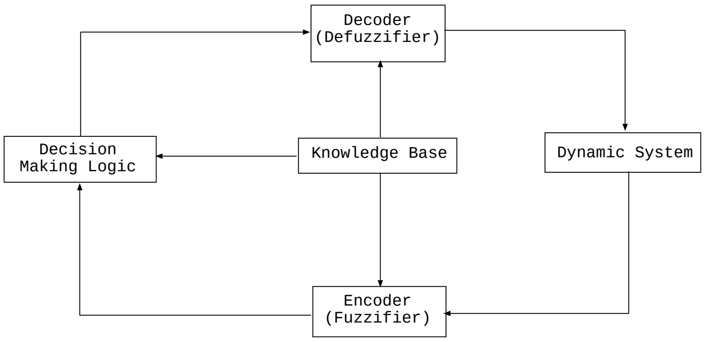

Figure 7: Basic architecture of a fuzzy controller.

TESEO solves all these three learning subtasks *very rapidly*. It reaches the goal efficiently and without colliding after travelling 10 times from the starting location to the desired goal. The total length of the first trajectory is approximately 13 meters while the length of the trajectory generated after TESEO has learned the suitable sequence of reactions is about 10 meters. This experiment was run several times, and TESEO learned the suitable motor skills after, at most, 13 trials.

To conclude, in this project we have shown how a goal-directed autonomous mobile robot can rapidly learn efficient navigation strategies in an unknown indoor environment. Our robot TESEO self-adapts permanently as it interacts with the environment through reinforcement connectionist learning. TESEO is not only operational from the very beginning and improves its performance with experience, but also learns to avoid collisions even when its sensors cannot detect the obstacles. This is a definite advantage over non-learning reactive robots. Finally, TESEO also exhibits incremental learning, high tolerance to noisy sensory data, and good generalization abilities. All these features make our robot learning architecture very well suited to real-world applications.

5.5 Manual fuzzy controller design[20]

Unlike neural network learning techniques, fuzzy logic can also be used as a mean to describe control knowledge in a user-friendly way. The strength of a fuzzy controller (FC) is its capability to handle both linguistic knowledge and numerical sensor data. For the application considered in B-Learn II, fuzzy control is also a means to achieve non-linear control. The basic fuzzy control architecture is shown in figure 7. The fuzzifier maps continuous numerical sensor data into discretized linguistic descriptions. The rule matcher maps these discrete values to a set of discrete control values by means of heuristic rules. The defuzzifier maps these into continuous numerical control signals. A FC is thus characterized by its linguistic variables, its rule base and its fuzzification and defuzzification procedure. Trapezoidal membership functions, that are symmetrical for positive and negative linguistic variables, were chosen for the experiments.

Rules similar to the Sugeno type ([13]) were used because of real-time constraints. These rules were observed to be as effective despite the fact that their fuzziness is limited to the antecedent part and their consequent being a numerical value $v_i = f(X, Y \ldots)$ with $X, Y, \ldots$ being input variables.

The experiments were performed on the KUKA-IR 361, equipped with a force sensor made at KULeuven/PMA (see also section 5.1.1). The robot task consists of inserting an eccentrical round peg (peg diameter measures 35 mm) into a hole (hole length is 70 mm.). The clearance between peg and hole is 0.15 mm. This task is prone to two-point contact situations because of the eccentricity combined with a very flexible end-

[20]Original reference: [110]

Linguistic variable	Force (N)				Torque (Nmm)			
	m_1	m_2	m_3	m_4	m_1	m_2	m_3	m_4
zero	-2	-1	1	2	-12	-3	3	12
low	1	2	4	10	3	12	18	120
medium	4	10	20	40	18	120	180	1200
high	20	40	60	200	180	1200	1800	3000
to_infinity	60	200	350	5000	1800	3000	5000	19000

Table 4: Trapezoidal membership function definitions of the fuzzy controller.

point compliance. The FC was implemented in ANSI-C, and was compiled with a Logical Systems C compiler. A real-time sampling frequency of 150 Hz was attained. To achieve this frequency, the rules and fuzzification and defuzzification procedures were implemented as macros. This avoids the time wasted with parameter passing and stack operations due to function calls.

5.6 Concept learning[21]

In B-Learn II, concept learning refers especially to the learning of operational descriptions of objects existing in the environment of a mobile robot ([106], [73], [35]). For instance, the task might be to learn what it means to pass through a doorway. Consider a trace where the robot moves through the doorway parallel to the door frames. When approaching the door, the sensors on the right side perceive the jump caused by the door frame. Correspondingly, the sensors on the front left side perceive the jump caused by the left door frame. The sensors at the front constantly measure the back wall of the room. This is what human inspectors of the sensory data realize. The issue now is to have the system detect these relations by machine learning. In this application, we learned the following three types of rules:

- Rules for patterns for single sensors
- Rules for patterns for classes of sensors
- Rules for going through a door in terms of patterns for sensor groups

For learning horn clauses characterizing these perceptions, we used algorithms with the ability to restrict the hypothesis space, i.e., the space of learnable rules, syntactically, because other learning algorithms could learn too many unusable rules [72]. The most applicable learning algorithms were **RDT** ([71]) and a modification of it, **GRDT** ([72]). Both algorithms are model-based learners, learning instantiations of rule schemata. They learn from most general rules to more specific rules. In contrast to learning algorithms that learn most specific generalizations, they learn most general rules that fulfill the user given evaluation criterion. In contrast to systems that stop after having found a good rule, they learn as many most general rules as possible.

The algorithms learn from classified examples. Their Input are instances of the goal predicate to be characterized and the events used to describe the goal concept. Additionally, background knowledge like information about the sensor classes, etc., was given.

In a nutshell, the results are as follows. Given 1004 examples for the four sensor features, 129 rules were learned, covering 87% of the given examples. An example rule characterizing possible perceptions the robot gets from a "jump" is:

[21] Original references: [106], [73]

stable(Trace, Orientation, Sensor, $Time_1$, $Time_2$, $Grad_1$) &
incr_peak(Trace, Orientation, Sensor, $Time_2$, $Time_3$, $Grad_2$) &
stable(Trace, Orientation, Sensor, $Time_3$, $Time_4$, $Grad_3$)
$\rightarrow$ s_jump(Trace, Sensor, $Time_1$, $Time_4$, parallel).

The rule describes that, first, the measured distance of a single sensor must be more or less stable, then the distance must become much greater and then it must be stable again.

For learning sensor group features, we had 956 examples, from which we learned 136 rules. Using the learned rules for sensor features and sensor group features, we got a coverage of 64%. We present an example rule for sensor group features, too:

s_jump(Trace, $Sensor_1$, $Start_1$, End_1, parallel) & sclass(Trace, $Sensor_1$, right_side)
&
s_jump(Trace, $Sensor_2$, $Start_2$, End_2, parallel) & sclass(Trace, $Sensor_2$, right_side)
&
succ($Start_1$, $Start_2$) & succ(End_1, End_2)
$\rightarrow$ sg_jump(Trace, right_side, $Start_1$, End_2, parallel).

According to this rule, at least two sensors of the right-side sensors must perceive a "jump". Additionally, start and end time of the perception of the second sensor must follow immediately the perception of the first sensor.

For the goal concept through_door, we had ten examples. We learned three rules, two of them covering exactly the ten examples, one of them describing that on both sides of the robot, the robot must perceive a jump:

sg_jump(Trace, right_side, T_1, T_2, parallel) &
sg_jump(Trace, left_side, T_1, T_2, parallel)
$\rightarrow$ through_door(Trace, T_1, T_2, both_sides, parallel).

The third learned rule was a poor one deriving examples as well as additional facts. But this rule was not necessary to cover the examples. All in all, the results promise good further results.

5.7 Learning of diagnostic knowledge[22]

With respect to diagnosis, the experimental work realized concerns the identification of the execution failure. Some failures can easily be identified by simple discrete sensors. For instance, if the wrong tool is attached to the robot, this can be detected by one sensor. A part missing in the feeder may as well be detected with little effort. Such kind of knowledge can be easily encoded by hand as rules. In fact, as a result of a first attempt, it seems that it can be more cost effective to hand-code rules for monitoring binary sensors than to devise a costly training strategy suitable for the acquisition of enough examples to be used by a learning algorithm.

However, how to characterize the situation in which the force profile in the robot wrist is different from normal? Different external exceptions can occur causing execution failures that manifest through abnormal force and torque profiles. These profiles, although sometimes recognizable by a human, are difficult to model analytically. Therefore, what would be desirable is that the system learned to "look" at the force profiles in order to identify different situations.

The performed experiments are a first step in this direction. In general, when a failure is detected during the execution of an action, a trace of sensor data, for the period of time in which that failure occurred, is obtained. The failure should then be identified using diagnosis knowledge generated manually or automatically. In the case of

[22]Original references: [11], [88], [26]

the force/torque sensor data, it is not easy to generate that knowledge manually, and so machine learning algorithms, mainly inductive learning algorithms, are used.

The chosen situation for these first experiments was an operation in which the robot holds a part to be assembled. During the training phase, the operation was executed many times and several external exceptions were simulated. In most cases an object was placed in the path the robot had to move along.

The force traces in an interval surrounding each failure were collected, and the failure classification was associated to it. The length of the trace was 15 samples. After this phase, the data were analyzed using two very simple statistical variables: the average and the standard deviation. Diagrams of typical behaviour for each failure were produced. These diagrams include the lines of average force profile, average force plus standard deviation profile and average force minus standard deviation profile.

The second step was to prepare the data to be the input to the learning algorithm. If all numerical values of the forces in the trace, in total 45 values, are given to the learning algorithm, probably it will run less efficiently and the knowledge produced will be less readable and less efficient to use. On the other hand, when humans look at force profiles, they can easily recognize trends and high level features that the learning algorithm will ignore if the training set is not given to it in terms of the *good* features. For these first experiments, the raw data for each force were transformed to average force, slope, and monotonicity, in order to reduce the total number of features, with some of them (slope and monotonicity) being clearly of a higher level of abstraction.

In our application, 120 classified examples were used. The generated tree has 53 leaf nodes, which means that 53 diagnosis rules are contained in it. About two thirds of these rules can uniquely identify the class of the object (i.e. the error classification). Most of the other rules assign a set of two classes to the given error. Only 9 % of the generated rules assign a set of three or four classes to the error. The algorithm may be parameterized in order to obtain larger trees, thereby minimizing the number of leaf nodes labeled with more than one class.

One particularly interesting result is that the attributes selected by automatically by the algorithm were related to the force in Z-axis. This is knowledge that could not be anticipated, but which makes sense, since the operation that is being considered is the approach operation in Z direction.

Complementary to the learning experiments described above, a joint activity with the Universita di Torino is being carried out. Torino's SMART+ (a multi-strategy learning system, see [21]) is being investigated as an alternative learning system in the assembly supervision task. In general, the approach is to evaluate various learning techniques, compare their results and define an architecture where possibly different learning algorithms can be applied for the various functional blocks and various levels of abstraction in the monitoring architecture.

5.8 Learning of planning knowledge

5.8.1 Fine motion planning[23]

The assembly planner must generate a sequence of commands for the robot control system in order to successfully perform the assembly task. During assembly, the plan must provide a command for each given arm configuration in order to correctly continue the task. This is equivalent to define the robot commands as a function over the Configuration Space. Since there are infinitely many possible configurations, two approaches are possible. One consists of defining an algebraic function like, for instance, a potential field.

[23] Original references: [138],[139]

However, in the presence of uncertainty and due to friction forces such a function may be quite difficult to determine.

The second way consists of subdividing the function domain into a finite number of sets, such that all the configurations in each set can be considered as equivalent for the planning purpose, having the same robot command assigned to them. These sets must be well defined in the presence of uncertainty, in order to clearly determine to which one a sensed configuration belongs. In this way it is possible to build a plan to move from an initial configuration within an "initial set" to a configuration within a "goal set" (unless the goal set is composed of only one configuration a "goal configuration" can not be guaranteed). The general procedure to build a plan is quite simple: look for a sequence of contiguous sets between the initial and goal ones and determine a command to move the arm from one set to the next in the sequence. The plan is executed by identifying to which set the current arm configuration belongs and executing the corresponding command afterwards.

Our work follows this approach, but since the direction of the reaction force is used to reduce uncertainty in the Configuration Space, the command function depends not only on the sensed configuration but also on the sensed reaction force.

We consider a finite number of *task states* defined according to the contacts between the objects to be assembled. Each task state can be reached in a set of sensed configurations, called *configuration observation domain*, that can be determined a priori from the nominal object models and the geometric uncertainty domains CU. The sets of two different states may have a non-empty intersection, and therefore information from the sensed reaction forces and torques is also processed in order to better identify the current state during task execution. So, the set of possible observed reaction forces, called *force observation domain*, is determined off-line for each task state by using the domains FU. The commands to perform transitions between states are characterized by *state transition operators* which represent movement directions determined taking into account the uncertainty in the robot control. The task states, obtained from the nominal model of the objects, are represented as nodes in a graph, *DG-Nom*. The plan itself is generated as follows:

1. *Search for a basic solution.* First, a path from the initial to the final state is selected in *DG-Nom*. Then, for each state in the path, a set of state transition operators that may allow transition to the next state in the path is determined. Finally, all the states that are not included in the path and may be reached applying the selected operators are considered. The result is a subgraph, *DG-Plan*, of *DG-Nom*, with each node of *DG-Plan* having an associated set of state transition operators.

2. *Search for a complete solution.* This is equivalent to the search for a basic solution, but considering as initial states the terminal states of *DG-Plan* different from the final state. This step is repeated until the unique terminal state of *DG-Plan* is the final state.

3. *Final adjustment of the plan.* In this step only one state transition operator from each previously selected set is chosen to be applied during task execution.

Different criteria can be used to guide the search for the sequences of states (for instance minimizing the number of operator changes or the number of states). The plan is composed of two main modules. One of them has the information to decide what to do when the task is in a given state, and the other contains the information to identify the current state on-line, i.e. the configuration and the force observation domains of the states in *DG-Plan*.

The task execution following the plan consists, basically, of identifying the current state by matching the sensed configuration and generalized force with the configuration and force observation domains of the possible states, and applying the corresponding state transition operator afterwards. A more detailed description of the planner as well as an example describing how it works can be found in [112] and [138].

Inclusion of Learning in the Planner

Some of the described planner basic parts have been implemented for planar movements using procedural techniques (including friction and different sources of uncertainty). This is not an easy task even for a 3-dof problem. The next step in the work consist of trying to build the modules of the plan by applying learning techniques to examples generated from several task executions (guided by a human operator or following predefined strategies) in order to avoid the hard task of analytically determining them. From another point of view, including learning techniques in the generation of the plan structure may lead to an important result, too: the improvement of the plan performance while it is been executed according to a first version.

Following these ideas, the module of the plan that performs the identification of the current state during task execution by using the precomputed configuration and force observation domains (CU and FU) is going to be built by learning from examples.

The task will be performed in simulation considering, each time, a random set of deviations within the corresponding uncertainty range for each uncertainty source. During contact situations, configurations and reaction forces are recorded simultaneously with the current state. This information is intended to be used to learn some simple rules allowing current state identification from sensed data during task execution.

5.8.2 Mobile robot path planning[24]

For the mobile robot path planning, an hierarchical approach has been employed. Based on the world model, on each level a navigation module decomposes the task which is given by the next higher level into a set of more simple ones. Using four levels, a location the robot should reach is transformed into a set of incremental position commands that are passed to the vehicle position controller. The four planners resp. controllers are a topological planner, a geometric planner, a reflexive control module and finally the robot motion controller.

Geometrical planning is restricted to a local area which is slightly larger than the perceivable range of the robot sensors, as planning is performed in the local geometric model. The central part of the geometric model is obtained by a fusion process of current sensor data and a-priori knowledge. Planning in areas further away is mainly a static procedure and can be performed on layers of higher abstraction. However, due to the local nature of the local model, the next graph node might not be located within the area of geometric planning. In such cases special care has to be taken in order to plan a path which finally leads to the correct topological node. Path planning itself is based on the idea of potential distance fields ([156]). The algorithm described below guarantees to find the shortest collision free path. Its main advantage is its computational efficiency. The holonomic driving principle of PRIAMOS offers the advantage that no additional constraints have to be taken into account. Summarizing, planning is done in five steps.

- If a topological path node is located outside the local model, a goal at the border of the local model from where the next topological path node can be reached has to be computed.
- Obstacles are enlarged by half of the robot width so that further planning can consider the robot as a point.

[24] Original references: [156], [155]

- In the third step, a square grid that covers the planning region and which is centered around the current vehicles position is generated. Each cell of the grid contains binary information whether this cell is free or covers an obstacle.

- The next step is path planning itself. The cell of the goal position is initialized with a start-value ('1'). The 4 neighbours of a cell are recursively (beginning with the goal-cell) given a value: *cell_value + 1* if they do not cover an obstacle. This procedure is repeated until the cell which covers the robot position is reached. Starting at the robot position, a simple gradient algorithm finds a path by successively moving to the lowest valued neighbour cell (8 neighbours are considered in this step).

- Finally, a polygon is defined by the list of cells. A path is represented by the corner-points of this polygon.

The complete obstacle transformation and planning procedure is illustrated in figure 8.

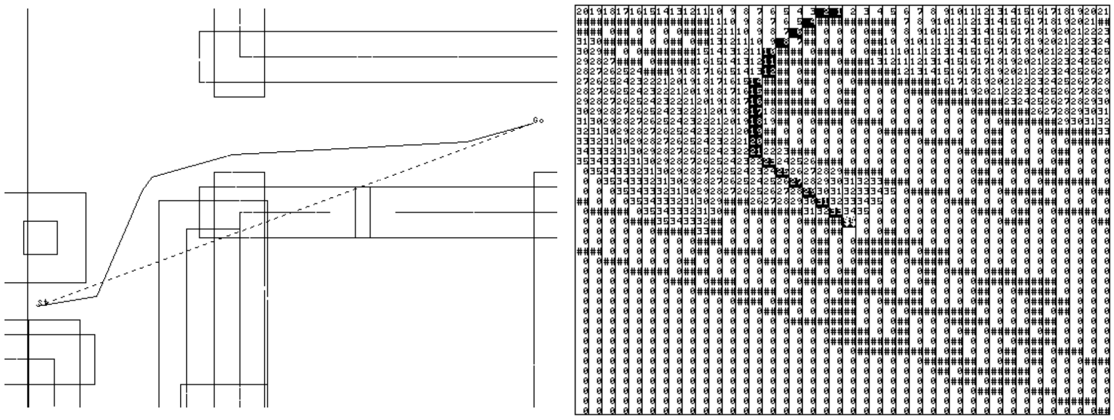

Figure 8: *A geometric planning example.* Shown is a ground plan of an office environment as well as the result of a planning cycle. (#: obstacles; $$: robot position; filled squares indicate the path)

The complete planning module provides two different modes of operation. In MOTION mode a trajectory towards a given goal is planned within the extension of the current local model. In NO_MOTION mode, two positions are specified. In this mode the planner reads the obstacles known to exist in this area from the database and executes a planning step in this static model. The NO_MOTION mode is necessary to support the topological planner in finding the optimal nodes for entering and leaving the topological graph. This is done by planning paths from the vehicles current position to the preselected entry nodes, respectively the preselected leaving nodes to the goal. In case of successful planning the length of a generated path is returned to the topological planner.

6 CONCLUSIONS

The main objective of B-Learn II is to enhance robot programming and robot autonomy by introducing learning capabilities in order to close the loop between sensing and action. To reach this aim, it is necessary to build up a thorough knowledge about both the application area and the machine learning techniques under consideration. The work described in this paper has served exactly for this purpose. The individual results that have been achieved are very good, but the real spinoff of B-Learn II will become evident after the pieces are combined.

So far, B-Learn II has proven that given proper expertise both from robotics and machine learning, the two fields can cross–fertilize each other very well. Robotics is in fact a challenging area for machine learning, and machine learning can help in development and deployment of real robotic systems.

Acknowledgement

This work has been funded by the ESPRIT Basic Research Action No. 7274, "B-Learn II". It has been performed at the Division of Production Engineering, Machine Design, and Automation, Prof. Dr. ir. Hendrik Van Brussel, Katholieke Universiteit Leuven, Belgium, at the Instituto de Cibernètica, Prof. Luis Basañez and Prof. Carme Torras, Universitat Politecnica de Catalunya, Spain, at the Lehrstuhl Informatik VIII, Prof. Dr. Katharina Morik, Department of Computer Science, University of Dortmund, Germany, at the Department of Biophysical and Electronic Engineering, Prof. G. Vernazza, Universita di Genova, Italy, at the Institute for Real-Time Computer Systems & Robotics, Prof. Dr.-Ing. U. Rembold and Prof. Dr.-Ing. R. Dillmann, Department of Computer Science, University of Karlsruhe, Germany, at the Departamento de Informatica, Prof. A. Steiger-Garcao, Universidade Nova de Lisboa, Monte Caparica, Portugal, and at the Dipartimento di Informatica, Prof. Dr. Lorenza Saitta and Prof. Dr. Attilio Giordana, Universita di Torino, Italy.

REFERENCES

(1) J.E. Albus, A.J. Barbera, and R.N. Nagel. Theory and practice of hierarchical control. In *Proceedings of the 23rd IEEE Computer Society International Conference*, 1981.

(2) J.S. Albus. A new approach to manipulator control: The cerebellar model articulation controller. *Dynamic Systems, Measurement and Control*, 1975.

(3) C.W. Anderson. *Learning and Problem Solving with Multilayer Connectionist Systems*. PhD thesis, University of Massachusetts, 1986.

(4) C.W. Anderson. Strategy learning with multilayer connectionist representation. Technical Report TR87-509.3, GTE Laboratory Inc., May 1988.

(5) P. H. Anderson, S. J. Torvinen, and L. Vasek. A concept for maintaining quality in flexible production. *Computers in Industry*, 17, 1991.

(6) M. Anthony and N.L. Biggs. *Computational Learning Theory: An introduction*. Cambridge University Press, 1992.

(7) C. Archibald and E. Petriu. Computational paradigm for creating and executing sensorbased robot skills. In *Proceedings of the 24th International Symposium on Industrial Robots (ISIR '93)*, 1993.

(8) H. Asada. Teaching and learning of compliance using neural nets: Representation and generation of nonlinear compliance. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pages 1237 – 1244, 1990.

(9) H. Asada and B.-H. Yang. Skill acquisition from human experts through pattern processing of teaching data. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, pages 1302 – 1307, 1989.

(10) M. M. Barata, T. W. Rauber, and A. Steiger-Garcao. Sensor integration for expert cnc machines supervision. In *Proceedings of the ETFA '92*, 1992.

(11) M. M. Barata, T. W. Rauber, and A. Steiger-Garcao. Prognostic and monitoring system for cnc machines. *Revue Europ éenne Diagnostic et Sûreté de Fonctionnement*, 1994.

(12) D. Barschdorff and L. Monostori. Neural networks- their applications and perspectives in intelligent machining. *Computers in Industry*, 17, 1991.

(13) A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 835–846, 1983.

(14) A.G. Barto and R.S. Sutton. Landmark learning: an illustration of associative search. *Biological Cybernetics*, 42:1–8, 1981.

(15) L. Basañez and R. Suárez. Uncertainty modelling in configuration space for robotic motion planning. In *Proceedings of the SYROCO'91*, 1991.

(16) R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

(17) H. R. Berenji. A reinforcement learning based architecture for fuzzy logic control. *Int. Journal of Approximate Reasoning*, 6:267 – 292, 1992.

(18) F. Bergadano, A. Giordana, and L. Saitta. *Machine learning: an integrated framework andits applications.* Ellis Horwood, 1991.

(19) K. Berns, R. Dillmann, and U. Zachmann. Reinforcement learning for the control of an autonomous mobile robot. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, Raleigh, NC, 1992.

(20) L.B. Booker. *Intelligent Behavior as an Adaptation to the Task Environment.* PhD thesis, University of Michigan, 1982.

(21) M. Botta and A. Giordana. SMART+: A multi-strategy learning tool. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '93)*, Chamberry, France, 1993.

(22) R. C. Brost and M. T. Mason. Graphical analysis of planar rigid-body dynamics with multiple frictional contacts. In *Fifth Int. Symp. of Robotics Research*, 1989.

(23) B. Brunner, G. Hirzinger, K. Landzettel, and J. Heindl. Multi-sensory shared autonomy and telesensor-programming – key issues in the space robot technology experiment ROTEX. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '92)*, Yokohama, Japan, 1992.

(24) H. Van Brussel and J. Simons. The adaptable compliance concept and its use for automatic assembly by active force feedback accommodation. In *Proceedings of the 9th International Symposium on Industrial Robots (ISIR '79)*, 1979.

(25) S. J. Buckley. Teaching compliant motion strategies. *IEEE Transactions on Robotics and Automation*, 5(1), 1989.

(26) L. M. Camarinha-Matos, L. Seabra Lopes, and J. Barata. Execution monitoring in assembly with learning capabilities. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1994.

(27) W. W. Cooley and P. R. Lohnes. *Multivariate Data Analysis.* Wiley & Sons, New York, 1971.

(28) I.J. Cox and G.T. Wilfong. *Autonomous Robot Vehicles.* Springer, Berlin, Heidelberg, New York, 1990.

(29) J.L. Crowley. Dynamic modelling of free space for a mobile robot. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 1989.

(30) A. Curran and K.J. Kyriakopoulos. Sensor-based self-localization for wheeled mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Atlanta, Georgia, 1993.

(31) A. I. Cypher. *Watch what I do – Programming by Demonstration.* MIT Press, Cambridge, Massachusetts, 1993.

(32) N. Delson and H. West. Robot programming by human demonstration: subtask compliance controller identification. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, Yokohama, Japan, 1993.

(33) P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice Hall, 1982.

(34) R. Dillmann. *Lernende Roboter; Aspekte maschinellen Lernens.* Springer Verlag Berlin, Heidelberg, 1988.

(35) R. Dillmann, M. Kaiser, and V. Klingspor. Incorporating learning capabilities in mobile robots. To appear.

(36) R. Dillmann, J. Kreuziger, and F. Wallner. PRIAMOS: An experimental platform for reflexive navigation. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS '93)*, 1993.

(37) B. Dufay and J.-C. Latombe. An approach to automatic robot programming based on inductive learning. In *Proceedings of the 1st International Symposium on Robotics Research*, 1984.

(38) A. Elfes. A sensor-based mapping and navigation system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1986.

(39) M. Erdmann. On motion planning with uncertainty. Master's thesis, MIT Artificial Intelligence Laboratory, 1984.

(**40**) R. Forsty and R. Rada. *Machine learning applications in expert systems and information retrieval.* Ellis Horwood, 1986.

(**41**) A. Gelperin, J.J. Hopfield, and D.W. Tank. The logic of Limax learning. In *Model neural networks and behavior.* A. Selverson, New York: Plenum Press, 1985.

(**42**) J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40:11 – 61, 1989.

(**43**) A. Giordana and C. Baraglio. Automatic synthesis of a fuzzy controller. In *Proceedings of the Second European Workshop on Learning Robots*, Torino, Italy, 1993.

(**44**) A. Giordana, L. Camarinha-Marcos, M. Kaiser, J. del R. Millan, C. Moneta, K. Morik, and M. Nuttin. *B-Learn II - D 101.* B-Learn II - ESPRIT BRA Project No. 7274, 1993.

(**45**) A. Giordana, M. Kaiser, and M. Nuttin. On the reduction of costs for robot controller synthesis. In *International Symposium on Intelligent Robotic Systems (IRS '94)*, Grenoble, France, 1994.

(**46**) A. Giordana, L. Saitta, and C. Baroglio. Learning simple recursive theories. In *Methodologies for Intelligent Systems, Proc. of the 7th International Symposium, ISMIS-93*, Trondheim, Norway, June 1993. Springer-Verlag.

(**47**) D. Guinea. Multi-sensor integration - an automatic feature selection and state identification methodology for tool wear estimation. *Computers in Industry*, 17, 1991.

(**48**) V. Gullapalli. *Reinforcement Learning and its application to control.* PhD thesis, University of Massachusetts, Department of Computer and Information Science, 1992.

(**49**) V. Gullapalli, R.A. Grupen, and A.G. Barto. Learning reactive admittance control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Nice, France, 1992.

(**50**) M.M. Gupta and D.H. Rao. On the principles of fuzzy neural networks. *Fuzzy Sets and Systems*, (61):1 – 18, 1994.

(**51**) S.E. Hampson. *A neural model of adaptative behavior.* PhD thesis, Department of Information and Computer Science, University of California, Irvine, 1983.

(**52**) T. Hasegawa, T. Suehiro, and K. Takase. A model-based manipulation system with skill-based execution. *IEEE Transactions on Robotics and Automation*, 8(5), 1992.

(**53**) Y. Hayashi. A neural expert system with automated extraction of fuzzy if-then rules and its application to medical diagnosis. In *Advances in Neural Information Processing Systems 3 (NIPS-3)*, Denver, Colorado, 1990.

(**54**) R. Hecht-Nielsen. *Neurocomputing*. Addison Wesley, Reading, MA, 1990.

(**55**) R. Heise. *Demonstration instead of programming: Focussing attention in robot task acquisition.* Research report no. 89/360/22, Department of Computer Science, University of Calgary, 1989.

(**56**) G. E. Hinton. Special issue on connectionist symbolic processing. *Artificial Intelligence*, November 1990.

(**57**) K. Hirota, Y. Arai, and S. Hachisu. Moving mark recognition and moving object manipulation in a fuzzy controlled robot. *Control-Theory and Advanced Technology*, 2, 1986.

(**58**) G. Hirzinger. ROTEX – the first robot in space. In *Proceedings of the International Conference on Advanced Robotics (ICAR '93)*, Tokyo, Japan, 1993.

(**59**) J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard. *Induction processes of inference, learning, and discovery.* The MIT Press, 1987.

(**60**) J.H. Holland. Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule based systems. In R.S. Michalski, J.C. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An artificial intelligence approach*, volume 2. Morgan Kaufmann, 1986. Los Altos, Ca.

(**61**) T. C. Hsia. Adaptive control for robot manipulators - a review. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1986.

(**62**) Y. K. Hwang and N. Ahuja. Gross motion planning – a survey. *ACM Computing Surveys*, 24(3):219 – 293, 1992.

(**63**) J.-S. R. Jang. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3), 1993.

(**64**) M. Junkar, B. Filipic, and I. Bratko. Identifying the grinding process by means of inductive learning. *Computers in Industry*, 17, 1991.

(**65**) L.P. Kaelbling. *Learning in embedded systems.* PhD thesis, Stanford University, 1990. Technical report TR-90-04.

(66) M. Kaiser. A framework for the generation of robot controllers from examples. In *Proceedings of the 10th ISPE/IFAC Symposium on CAD/CAM, Robotics, and Factories of the Future*, Ottawa, Canada, 1994.

(67) M. Kaiser. Time-delay neural networks for robot control. Submitted to *Symposium on Robot Control '94 (SYROCO '94)*, Capri, Italy, 1994.

(68) M. Kaiser, A. Giordana, and M. Nuttin. Integrated acquisition, execution, evaluation and tuning of elementary skills for intelligent robots. In *Proceedings of the IFAC Symposium on Artificial Intelligence in Real Time Control (AIRTC '94)*, Valencia, Spain, 1994.

(69) M. Kaiser and J. Kreuziger. Integration of symbolic and connectionist processing to ease robot programming and control. In *ECAI'94 Workshop on Combining Symbolic and Connectionist Processing*, 1994.

(70) M. Kaiser and F. Wallner. Using machine learning for enhancing mobile robots' skills. In *Proceedings of the IRTICS'93 workshop on intelligent real-time control systems*, Madrid, Spain, 1993.

(71) J.-U. Kietz and S. Wrobel. Controlling the complexity of learning in logic through syntactic and task-oriented models. In Stephen Muggleton, editor, *Inductive Logic Programming*, chapter 16, pages 335 – 360. Academic Press, London, 1992. Also available as Arbeitspapiere der GMD No. 503, 1991.

(72) V. Klingspor. On the application of ILP techniques to robot navigation tasks. In A. Giordana, editor, *Proceedings of the II European Workshop on Learning Robots*, Torino, Italy, 1993.

(73) V. Klingspor. Representation of operational concepts. In *Proc. of the Workshop on Planning and Configuration*, 1994. (in german).

(74) A.H. Klopf. A neuronal model of classical conditioning. Technical Report 87-1139, Wright Aeronautical Laboratories, OH: Wright-Patterson Air Force Base, 1987.

(75) Y. Kodratoff and R.S. Michalski. *Machine Learning: An artificial intelligence approach*, volume III. Morgan Kaufmann Publishers, 1991.

(76) T. Kohonen. The self–organizing map. *Proceedings of the IEEE*, 78, 1990.

(77) J. Kreuziger. Application of machine learning to robotics: An analysis. In *Proceedings of the Second International Conference on Automation, Robotics, and Computer Vision (ICARCV '92)*, Singapore, 1992.

(78) J. Kreuziger and S. Cord. *Anwendungen symbolischer Lernverfahren in der Robotik*, volume 23/92 of *Internal research report*. Universität Karlsruhe, 1992. (in german).

(79) S. Y. Kung and J. N. Hwang. Neural network architectures for robotic applications. *IEEE Transactions on Robotics and Automation*, 5(5), 1989.

(80) R.A. Olsen L. Breiman, J.H. Friedman and C.J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.

(81) T. Längle. Modellierung der lokalen Umgebung unter Berücksichtigung dynamischer Objekte. Master's thesis, University of Karlsruhe, 1993. (in german).

(82) J. J. Leonard and H. F. Durrant-Whyte. *Directed sonar sensing for mobile robot navigation*. Kluwer Academic Publishers, Boston, London, Dordrecht, 1992.

(83) C.-S. Lin and H. Kim. Use of CMAC neural networks in reinforcement self-learning control. In *Artificial Neural Networks (ANN '91)*, 1991.

(84) L. J. Lin. Programming robots using reinforcement learning and teaching. In *Proceedings of the AAAI '91*, 1991.

(85) L. J. Lin. *Reinforcement learning for robots using neural networks*. PhD thesis, Carnegie Mellon University, School of Computer Science, 1993.

(86) L. J. Lin. Scaling up reinforcement learning for robot control. In *Machine Learning: Proceedings of the Tenth International Conference*, pages 182–189, 1993.

(87) S. Liu and H. Asada. Teaching and learning of deburring robots using neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Atlanta, Georgia, 1993.

(88) L. Sebra Lopes and L. M. Camarinha-Matos. Learning in assembly task execution. In *Proceedings of the Second European Workshop on Learning Robots*, 1993.

(89) S. Mahadevan. Enhancing transfer in reinforcement learning by building stochastic models of robot actions. In *Proceedings of the International Workshop on Machine Learning*, 1992.

(90) S. Mahadevan and J. Connel. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55, 1992.

(91) M.T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man and Cybernetics*, 11, 1981.

(92) L. Matthies and A. Elfes. Integration of sonar and stereo range data using a grid-based representation. In *Proceedings of the 26th IEEE Conference on Decision and Control*, Los Angeles, CA, 1987.

(93) R. S. Michalski. Special issue on multistrategy learning. *Machine Learning*, 11, 1993.

(94) J. del R. Millán. *A reinforcement connectionist learning approach to robot path finding*. Ph.D. thesis, Dept. de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, 1992.

(95) J. del R. Millán. Learning efficient reactive behavioral sequences from basic reflexes in a goal-directed autonomous robot. In *Proceedings of the third International Conference on Simulation of Adaptive Behavior*, 1994.

(96) J. del R. Millán. Reinforcement learning for robot navigation. In P. van der Smagt, editor, *Neural Systems for Robotics*. Norwood, NJ, forthcoming.

(97) J. del R. Millán. Reinforcement learning of goal-directed obstacle avoidance reaction strategies in an autonomous mobile robot. *Robotics and Autonomous Systems*, to appear.

(98) J. del R. Millán and C. Torras. A reinforcement connectionist approach to robot path finding in non-maze-like environments. *Machine Learning*, 8:363 – 395, 1992.

(99) J. del R. Millán and C. Torras. Efficient reinforcement learning of navigation strategies in an autonomous robot. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 1994.

(100) T. M. Mitchell and S. B. Thrun. Explanation-based neural network learning for robot control. In *Advances in Neural Information Processing Systems 5 (NIPS-5)*, Denver, Colorado, 1992.

(101) C. Moneta and F.G.B. De Natale. Adaptive control in visual sensing. In *Proceedings of the IMACS International Symposium on Signal Processing, Robotics, and Neural Networks*, 1994.

(102) C. Moneta, G. Vernazza, and R. Zunino. On the need for integrated approaches to image understanding. *European Transactions on Telecommunications*, 3(4):465 – 478, 1992.

(103) J. Moody and C. Darken. Learning with localized receptive fields. In T. Sejnowski D. Touretzky, G. Hinton, editor, *Proceedings of the Connectionist Models Summer School*. Carnegie Mellon University, 1988.

(104) A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 1993.

(105) J.W. Moore, J.E. Desmond, N.E. Berthier, D.E. Blazis, R.S. Sutton, and A.G. Barto. Simulation of the classically conditioned nictitating membrane response by a neuron-like adaptive element: Response topography, neuronal firing and interstimulus intervals. *Behavioral Brain Research*, pages 143–154, 1986.

(106) K. Morik and A. Rieger. Learning action-oriented perceptual features for robot navigation. In Attilio Giordana, editor, *Proc. of the ECML-93 Workshop on Learning Robots*, Vienna, Austria, 1993. Also available as Research Report 3,University of Dortmund, Dept. Computer Science VIII, D-44221 Dortmund.

(107) S. Münch, J. Kreuziger, M. Kaiser, and R. Dillmann. Robot programming by demonstration - using machine learning and user interaction methods for the development of easy and comfortable robot programming systems. In *Proceedings of the International Symposium on Industrial Robots (ISIR '94)*, Hannover, Germany, 1994.

(108) K. B. Natajaran. *Machine learning: a theoretical approach*. Morgan Kaufmann, 1991.

(109) M. Nuttin and C. Baroglio. Fuzzy controller synthesis in robotic assembly: Procedure and experiments. In *Proceedings of the Second European Workshop on Learning Robots*, Torino, Italy, 1993.

(110) M. Nuttin, H. Van Brussel, C. Baroglio, and R. Piola. Fuzzy controller synthesis in robotic assembly: Procedure and experiments. In *FUZZ-IEEE-94: Third IEEE International Conference on Fuzzy Systems, World Congress on Computational Intelligence*, 1994.

(111) M. Nuttin, A. Giordana, M. Kaiser, and R. Suarez. *B-Learn II - D 201*. B-Learn II - ESPRIT BRA Project No. 7274, 1993.

(112) M. Nuttin, A. Giordana, M. Kaiser, and R. Suarez. *B-Learn II - D 202*. B-Learn II - ESPRIT BRA Project No. 7274, 1993.

(113) M. Nuttin, J. Peirs, A.S. Soembagijo, S. Sonck, and H. Van Brussel. Learning the peg-into-hole assembly with a connectionist reinforcement technique. In *Submitted to: Symposium on Robot Control '94 (SYROCO '94)*, 1994.

(114) J. Peng and R. J. Williams. Efficient learning and planning within the DYNA framework. In *Proceedings of the Second International Conference on Simulation of Adaptive Behaviour*, 1992.

(115) E. Plaza, editor. *Proceedings of the Workshop on Integrated Learning Architectures (ILA '93)*, Vienna, Austria, 1993.

(116) J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81 – 106, 1986.

(117) T. W. Rauber, M. M. Barata, and A. Steiger-Garcao. A toolbox for analysis and visualization of sensor data in supervision. In *Proceedings of the TOOLDIAG '93*, 1993.

(118) U. Rembold, R. Dillmann, and P. Levi, editors. *6. Fachgespräch autonome mobile Systeme (AMS 6)*, 1990.

(119) U. Rembold, T. Lueth, and A. Hörmann. Advancement of intelligent machines. In *Proc. of the ICAM JSME International Conference on Advanced Mechatronics*, Tokyo, 1993.

(120) W. D. Rencken. Concurrent localization and map building for mobile robots using ultrasonic sensors. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, Yokohama, Japan, 1993.

(121) U. Roy and C. R. Liu. Feature-based representational scheme of a solid modeler for providing dimensioning and tolerancing information. *Robotics and Computer Integrated Manufacturing*, 4, 1988.

(122) D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, pages 533 – 536, 1986.

(123) D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing : Explorations in the Microstructure of Coginition, Parts I & II.* MIT Press, 1986.

(124) A.L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 1959.

(125) J. G. Schneider and C. M. Brown. Robot skill learning, basis functions, and control regimes. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Atlanta, Georgia, 1993.

(126) J. De Schutter and H. Van Brussel. Compliant robot motion, a control approach based on external control loops. *International Journal on Robotics Research*, 7(4), August 1988.

(127) J. De Schutter and H. Van Brussel. Compliant robot motion, a formalism for specifying compliant motion tasks. *International Journal on Robotics Research*, 7(4), August 1988.

(128) J. De Schutter, S. Demey, H. Van Brussel, S. Dutre, W. Persoons, W. Witvrouw, and P. Van De Poel. Model based and sensor based programming of compliant motion tasks. In *Proceedings of the 24th International Symposium on Industrial Robots (ISIR '93)*, 1993.

(129) J. De Schutter, W. Witvrouw, P. Van De Poel, and H. Bruyninckx. Rosi: a task specification and simulation tool for force sensor based robot control. In *Proceedings of the 24th International Symposium on Industrial Robots (ISIR '93)*, 1993.

(130) L. Seabra Lopes. Sistema integrado de inducao. Gr rt-sd-7-91, Universidade Nova de Lisboa, 1991.

(131) A. M. Segre. *Machine Learning of Robot Assembly Plans.* Kluwer Academic Publishers, 1989.

(132) S. P. Singh. Reinforcement learning within a hierarchy of abstract models. In *Proceedings of the AAAI '92*, 1992.

(133) S. P. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8, 1992.

(134) S. P. Singh. *Learning to solve markovian decision processes.* PhD thesis, University of Massachusetts, Department of Computer Sciene, 1994.

(135) E.D. Sontag. Some topics in neural networks and control. In *Proceedings of the European Control Conference*, 1993.

(136) A. Steiger-Garcao, M. M. Barata, and L. F. S. Gomes. Integrated environment for prognosis and monitoring system support. In *Proceedings of the 1st UNIDO workshop on Robotics and Computer Integrated Manufacturing*, 1989.

(137) M. R. Stein and R. P. Paul. Behavior based control in time delayed teleoperation. In *Proceedings of the International Conference on Advanced Robotics (ICAR '93)*, Tokyo, Japan, 1993.

(138) R. Suárez and L. Basañez. Fine motion planning in presence of uncertainty. In *Proceedings of the Second European Workshop on Learning Robots*, 1993.

(139) R. Suárez, L. Basañez, and J. Rosell. Assembly contact force domains in the presence of uncertainty. Technical Report IC-DT-9403, Instituto de Cibernética (UPC-CSIC), Barcelona, Spain, 1994. also submitted to SYROCO'94.

(140) R. S. Sutton. Learning to predict by methods of temporal difference. *Machine Learning*, 3:9 – 44, 1988.

(141) R. S. Sutton. Integrated modeling and control based on reinforcement learning and dynamic programming. In *Advances in Neural Information Processing Systems 3 (NIPS-3)*, Denver, Colorado, 1990.

(142) R.S. Sutton and A.G. Barto. A temporal-difference method of classical conditioning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, pages 355–378, Seattle, WA, 1987. Lawrence Erlbaum.

(143) R.S. Sutton, A.G. Barto, and R.J. Williams. Reinforcement learning is direct adaptive control. *IEEE Control Systems Magazine*, pages 19–22, April 1992.

(144) L. Tollenaere. Supersab: Fast adaptive back propagation with good scaling properties. *Neural Networks*, pages 561–573, 1990.

(145) C. Torras. Symbolic planning versus neural control in robots. In F. Cervantes P. Rudomin, M. Arbib, editor, *Natural and Artificial Intelligence: A Meeting between Neuroscience and AI.* 1992.

(146) C. Torras. Neural learning for robot control. In *Proceedings of the ECAI '94*, 1994.

(147) G. G. Towell and J. W. Shavlik. Interpretation of artificial neural networks: mapping knowledge-based networks into rules. In *Advances in Neural Information Processing Systems 3 (NIPS-3)*, Denver, Colorado, 1990.

(148) G. G. Towell and J. W. Shavlik. Using symbolic learning to improve knowledge-based neural networks. In *Proceedings of the tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 177 – 182, 1992.

(149) G. G. Towell, J. W. Shavlik, and M. O. Noordewier. Refinement of approximate domain theories by knowledge-based neural networks. In *Proceedings of the eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 861 – 866, 1990.

(150) P. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4:161–186, 1989.

(151) L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134 – 1142, 1984.

(152) W. Van De Velde. Special issue: Towards learning robots. *Robotics and Autonomous Systems*, 8(1,2), 1991.

(153) T. P. Vogl. Accelerating the convergence of the back-propagation method. *Biological Cybernetics*, 1988.

(154) A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on acoustics, speech and signal processing*, March 1989.

(155) F. Wallner, M. Kaiser, and H. Friedrich. Integrated topological and geometrical planning in a learning mobile robot. In *IEEE/RSJ Conference on Intelligent Robots and Systems*, Munich, Germany, 1994.

(156) F. Wallner, T.C. Lüth, and F. Langiniuex. Fast local path planning for a local robot. In *Proc. of the Second International Conference on Automation, Robotics, and Computer Vision*, Singapore, 1992.

(157) F. Wallner, P. Weckesser, and R. Dillmann. Calibration of the active stereo vision system kastor with standardized perspective matrices. In *Proceedings of the Second International Conference on Optical 3D Measurement Techniques*, Zürich, Switzerland, 1993.

(158) C. J. C. H. Watkins. *Learning with delayed rewards.* PhD thesis, University of Cambridge, 1989.

(159) B. L. Whitehall and S. C. Lu. Machine learning in engineering automation: the present and the future. *Computers in Industry*, 17, 1991.

(160) R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, pages 229 –256, 1992.

(161) I.H. Witten. An adaptive optimal controller for discrete-time markov environments. *Information and Control*, 34:286–295, 1977.

(162) S. Wrobel. Towards a model of grounded concept formation. In *Proc. 12th International Joint Conference on Artificial Intelligence*, pages 712 – 719, Los Altos, CA, 1991. Morgan Kaufman.

(163) Y. Xu and R. P. Paul. A robot compliant wrist system for automated assembly. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1750 – 1755, 1990.

(164) J. Yang, Y. Xu, and C.S. Chen. Hidden markov model approach to skill learning and its application in telerobotics. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Atlanta, Georgia, 1993.

(165) L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338 – 353, 1965.