24

Intelligent Soft-Computing Techniques in Robotics

Dustic M. Katić
Mihajlo Pupin Institute

Branko Karan
Mihajlo Pupin Institute

24.1 Introduction

Robots and machines that perform various tasks in an intelligent and autonomous manner are required in many contemporary technical systems. Autonomous robots have to perform various anthropomorphic tasks in both unfamiliar or familiar working environments by themselves much like humans. They have to be able to determine all possible actions in unpredictable dynamic environments using information from various sensors. In advance, human operators can transfer to robots the knowledge, experience, and skill to solve complex tasks. In the case of a robot performing tasks in an unknown enviroment, the knowledge may not be sufficient. Hence, robots have to adapt and be capable of acquiring new knowledge through learning. The basic components of robot intelligence are actuation, perception, and control. Significant effort has been attempted to make robots more intelligent by integrating advanced sensor systems as vision, tactile sensing, etc. But, one of the ultimate and primary goals of contemporary robotics is development of intelligent algorithms that can further improve the performance of robotic systems, using the above-mentioned human intelligent functions.

Intelligent control is a new discipline that has emerged from the classical control disciplines with primary research interest in specific kinds of technological systems (systems with recognition

in the loop, systems with elements of learning and self-organization, systems that sometimes do not allow for representation in a conventional form of differential and integral calculus). Intelligent control studies high-level control in which control strategies are generated using human intelligent functions such as perception, simultaneous utilization of a memory, association, reasoning, learning, or multi-level decision making in response to fuzzy or qualitative commands. Also, one of the main objectives of intelligent control is to design a system with acceptable performance characteristics over a very wide range of structured and unstructured uncertainties.

The conditions for development of intelligent control techniques in robotics are different. It is well known that classic model-based control algorithms for manipulation robots cannot provide desirable solutions, because traditional control laws are, in most cases, based on a model with incomplete information and partially known or inaccurately defined parameters. Classic algorithms are extremely sensitive to the lack of sensor information, unplanned events, and unfamiliar situations in robots' working environment. Robot performance is not able to capture and utilize past experience and available human expertise. The previously mentioned facts and examples provide motivation for robotic intelligent control capable of ensuring that manipulation robots can sense the environment, process the information necessary for uncertainty reduction, and plan, generate, and execute high-quality control action. Also, efficient robotic intelligent control systems must be based on the following features:

1. Robustness and great adaptability to system uncertainties and environment changes
2. Learning and self-organizing capabilities with generalization of acquired knowledge
3. Real-time implementation on robot controllers using fast processing architectures

The fundamental aim of intelligent control in robotics represents the problem of uncertainties and their active compensation. Our knowledge of robotic systems is in most cases incomplete, because it is impossible to describe their behavior in a rigorous mathematical manner. Hence, it is very important to include learning capabilities in control algorithms, i.e., the ability to acquire autonomous knowledge about robot systems and their environment. In this way, using learning active compensation of uncertainties is realized, which results in the continous improvement of robotic performances. Another important characteristic that must be included is knowledge generalization, i.e., the application of acquired knowledge to the general domain of problems and work tasks.

Few intelligent paradigms are capable of solving intelligent control problems in robotics. In addition, symbolic knowledge-based systems (expert systems), connectionist theory, fuzzy logic, and evolutionary computation theory (genetic algorithms) are very important in the development of intelligent robot control algorithms. Also, important in the development of efficient algorithms are hybrid techniques based on integration of particular techniques such as neuro-fuzzy networks, neuro-genetic, and fuzzy-genetic algorithms.

Connectionist systems (neural networks) represent massively parallel distributed networks with the ability to serve in advanced robot control loops as learning and compensation elements using nonlinear mapping, learning, parallel processing, self-organizing, and generalization. Usually, learning and control in neurocontrollers are performed simultaneously, and learning continues as long as perturbations are present in the robot under control and/or its environment.

Fuzzy control systems based on mathematical formulation of fuzzy logic have the ability to represent human knowledge or experience as a set of fuzzy rules. Fuzzy robot controllers use human knowhow or heuristic rules in the form of linguistic if–then rules, while a fuzzy inference engine computes efficient control action for a given purpose.

The theory of evolutionary computation with genetic algorithms represents a global optimization search approach that is based on the mechanics of natural selection and natural genetics. It combines survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with expected ever-improving perfomance.

The purpose of this chapter is to present intelligent techniques as new paradigms and tools in robotics. Basic principles and concepts are given, with an outline of a number of algorithms that have been shown to simulate or use a diversity of intelligent concepts for sophisticated robot control systems.

24.2 Connectionist Approach in Robotics

24.2.1 Basic Concepts

Connectionism is the study of massively parallel networks of simple neuron-like computing units.[9.19] The computational capabilities of systems with neural networks are in fact amazing and very promising; they include not only so-called "intelligent functions" like logical reasoning, learning, pattern recognition, formation of associations, or abstraction from examples, but also the ability to acquire the most skillful performance for control of complex dynamic systems. They also evaluate a large number of sensors with different modalities providing noisy and sometimes inconsistent information. Among the useful attributes of neural networks are

- *Learning*. During the training process, input patterns and corresponding desired responses are presented to the network, and an adaptation algorithm is used to automatically adjust the network so that it responds correctly to as many patterns as possible in a training set.
- *Generalization*. Generalization takes place if the trained network responds correctly with a high probability of inputting patterns that were not included in the training set.
- *Massive parallelism*. Neural networks can perform massive parallel processing.
- *Fault tolerance*. In principle, damage to a few links need not significantly impair overall performance. Network behavior gradually decays as the number of errors in cell weights or activations increases.
- *Suitability for system integration*. Networks provide uniform representation of inputs from diverse resources.
- *Suitability for realization in hardware*. Realization of neural networks using VLSI circuit technology is attractive, because identical structures of neurons make fabrication of neural networks cost-effective. However, the massive interconnection may result in some technical difficulties, such as power consumption and circuitry layout design.

Neural networks consist of many interconnected simple nonlinear systems that are typically modeled by appropriate activation functions. These simple nonlinear elements, called nodes or neurons, are interconnected, and the strengths of the interconnections are denoted by parameters called weights. A basic building block of nearly all artificial neural networks, and most other adaptive systems, is the adaptive linear combinier, cascaded by a nonlinearity which provides saturation for decision making. Sometimes, a fixed preprocessing network is applied to the linear combinier to yield nonlinear decision boundaries. In multi-element networks, adaptive elements are combined to yield different network topologies. At input, an adaptive linear combinier receives analog or digital input vector $x = [x_0, x_1, \ldots, x_n]^T$ (input signal, input pattern), and using a set of coefficients, the weight vector, $w = [w_0, w_1, \ldots w_n]^T$, produces the sum s of weighted inputs on its output together with the bias member b:

$$s = x^T w + b \tag{24.1}$$

The weighted inputs to a neuron accumulate and then pass to an activation function that determines the neuron output:

$$o = f(s) \tag{24.2}$$

The activation function of a single unit is commonly a simple nondecreasing function like threshold, identity, sigmoid, or some other complex mathematical function. A neural network is a collection of interconnected neurons. Neural networks may be distinguished according to the type of interconnection between the input and output of network. Basically, there are two types of networks: feedforward and recurrent. In a feedforward network, there are no loops, and the signals propagate in only one direction from an input stage through intermediate neurons to an output stage. With the use of a continuous nonlinear activation function, this network is a static nonlinear map that can be used efficiently as a parallel computational model of a continuous mapping. If the network possesses some cycle or loop, i.e., signals may propagate from the output of any neuron to the input of any neuron, then it is a feedback or recurrent neural network. In a recurrent network the system has an internal state, and thereby the output will also depend on the internal state of the system. Hence, the study of recurrent neural networks is connected to analysis of dynamic systems.

Neural networks are able to store experiential knowledge through learning from examples. They can also be classified in terms of the amount of guidance that the learning process receives from an outside agent. An *unsupervised learning* network learns to classify input into sets without being told anything. A *supervised learning* network adjusts weights on the basis of the difference between the values of the output units and the desired values given by the teacher using an input pattern. Neural networks can be further characterized by their network topology, i.e., by the number of interconnections, the node characteristics that are classified by the type of nonlinear elements used (activation rule), and the kind of learning rules implemented.

The application of neural networks in technical problems consists of two phases:

1. "Phase of learning/adaptation/design" is the special phase of learning, modifying, and designing the internal structure of the network when it acquires knowledge about the real system as a result of interaction with system and real environment using a trial-error method, as well as the result of the appropriate meta rules inherent to global network context.
2. "Pattern associator phase or associative memory mode" is a special phase when, using the stored associations, the network converges toward the stable attractor or a desired solution.

24.2.2 Connectionist Models with Applications in Robotics

In contemporary neural network research, more than 20 neural network models have been developed. Because our attention is focused on the application of neural networks in robotics, we briefly introduce some important types of network models that are commonly used in robotics applications. There are multilayer perceptrons (MP), radial basis function networks (RBF), recurrent version of multilayer perceptron (RMP), Hopfield networks (HN), CMAC networks, and ART networks.

For the study and application of feedforward networks it is convenient to use in addition to single-layer neural networks, more structured ones known as multilayer networks or *multilayer perceptrons*. These networks with an appropriate number of hidden levels have received considerable attention because of better representation capabilities and the possibility of learning highly nonlinear mappings. The typical network topology that represents a multilayer perceptron (Figure 24.1) consists of an input layer, a sufficient number of hidden layers, and the output layer. The following recursive relations define the network with $k + 1$ layers:

$$y_0 = u \tag{24.3}$$

$$y_l = f_l(W_l \bar{y}_{l-1}), \quad l = 1, \ldots, k \tag{24.4}$$

where y_l is vector of neuron inputs in the l-layer ($y_k = y$ – output of $k + 1$ is the network layer, u is network input, f_l is the activation function for the l layer, W_l is the weighting matrix between layers $l-1$ i $l, \bar{y}_j = [y_j, 1]$ is the adjoint vector y_j. In the previous equation, bias vector is absorbed by the weighting matrix.

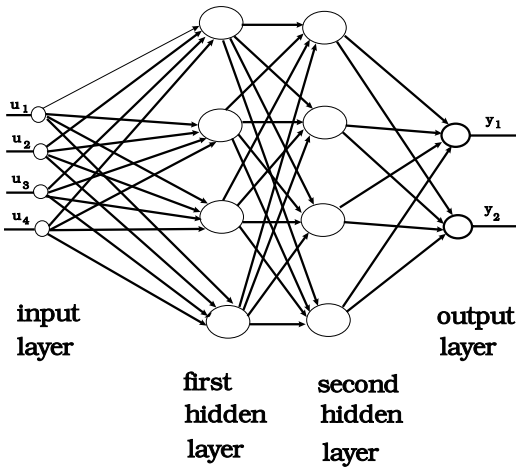

FIGURE 24.1 Multilayer perceptron.

Each layer has an appropriate number of neural units, where each neural unit has some specific activation function (usually a logistic sigmoid function). The weights of the networks are incrementally adjusted according to appropriate learning rules, depending on the task, to improve the system performance. They can be assigned new values in two ways: either via some prescribed offline algorithm that remains fixed during the operation, or adjusted by a learning process. Several powerful learning algorithms exist for feedforward networks, but the most commonly used algorithm is the *backpropagation algorithm*.[9] The backpropagation algorithm as a typical supervised learning procedure that adjusts weights in the local direction of greatest error reduction (steepest descent gradient algorithm) using the square criterion between the real network output and desired network output.

An RBF network approximates an input–output mapping by employing a linear combination of radially symmetric functions. The $k - th$ output y_k is given by:

$$y_k(u) = \sum_{i=1}^{m} w_{ki}\phi_i(u) \tag{24.5}$$

where:

$$\phi(u) = \phi\left(\left\|u - c_i\right\|\right) = \phi(r_i) = \exp\left(\frac{-\eta^2}{2\sigma_i^2}\right),\ r_i \geq 0, \sigma_i \geq 0 \tag{24.6}$$

The RBF network always has one hidden layer of computational modes with a nonmonotonic activation function $\phi(.)$. Theoretical studies have shown that the choice of activation function $\phi(.)$ is not very crucial to the effectiveness of the network. In most cases, the Gaussian RBF given by (24.6) is used, where c_i and σ_i are selected centers and widths, respectively.

One of the earliest sensory connectionist methods capable of serving as an alternative to the well-known backpropagation algorithm is the CMAC (cerebellar model arithmetic computer)[20] (Figure 24.2). The CMAC topology consists of a three-layer network, one layer being the sensory or command input, the second the association layer, and the third the output layer. The association layer is conceptual memory with high dimensionality. On the other hand, the output layer is the actual memory with low dimensionality. The connections between these two layers are chosen in a random way. The adjustable weights exist only between the association layer and the output layer. Using supervised learning, the training set of patterns is presented and, accordingly, the weights are adjusted. CMAC uses the Widrow-Hoff LMS algorithm[6] as a learning rule.

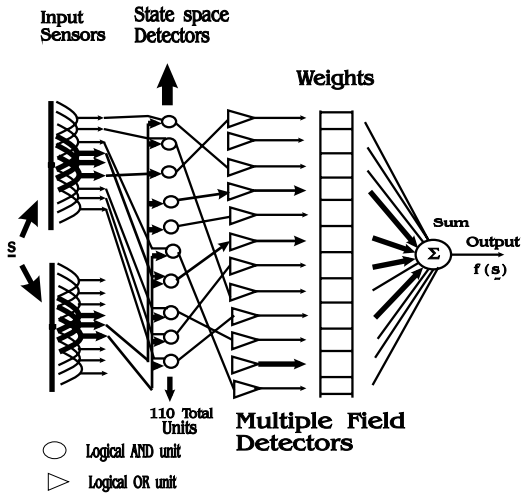

FIGURE 24.2 Structure of CMAC network.

CMAC is an associative neural network using the feature that only a small part of the network influences any instantaneous output. The associative property built into CMAC enables local generalization; similar inputs produce similar outputs while distant inputs produce nearly independent outputs. As a result, we have fast convergence properties. It is very important that practical hardware realization using logical cell arrays exists today.

If the network possesses some cycle or loop, then it is a feedback or recurrent neural network. In a recurrent network the system has an internal state, and the output will also depend on the internal state of the system. These networks are essentially nonlinear dynamic systems with stability problems. There are many different versions of inner and outer recurrent neural networks (recurrent versions of multilayer perceptrons) for which efficient learning and stabilization algorithms must be synthesized. One of the most commonly used recurrent networks is the Hopfield[23] type neural network that is very suitable for optimization problems. Hopfield introduced a network that employed a continuous nonlinear function to describe the output behavior of the neurons. The neurons are an approximation to biological neurons in which a simplified set of important computational properties is retained. This neural network model, which consists of nonlinear graded-response model neurons organized into networks with effectively symmetric synaptic connections, can be easily implemented with electronic devices. The dynamics of this network is defined by the following equation:

$$\dot{y}_i = -\alpha y_i + \beta f_i\left(\sum_j w_{ij} y_j\right) + I_i \qquad i = 1,..,n, \tag{24.7}$$

where α, β are positive constants and I_i is the array of desired network inputs.

A Hopfield network can be characterized by its energy function:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1} w_{ij} y_i y_j - \sum_{i=1}^{n} I_i y_i \tag{24.8}$$

The network will seek to minimize the energy function as it evolves into an equilibrium state. Therefore, one may design a neural network for function minimization by associating variables in an optimization problem with variables in the energy function.

ART networks are neural networks based on the Adaptive Resonance Theory of Carpenter and Grossberg.[17] An ART network selects its first input as the exemplar for the first cluster. The next input is compared to the first cluster exemplar. It is clustered with the first if the distance to the first cluster is less than a threshold. Otherwise it is the exemplar for a new cluster. This procedure is repeated for all the following inputs. If an input is clustered with the *j*th cluster, the weights of the network are updated according to the following formulae

$$w_{ij}(t+1) = \frac{v_{ij}(t)u_i}{0.5 + \sigma_{i=1}^{n} v_{ij}(t)u_i} \tag{24.9}$$

$$\nu_{ij}(t+1) = u_i v_{ij}(t) \tag{24.10}$$

where $i = 1, 2, \ldots, M$. ART networks belong to the class of unsupervised learning networks. They are stable because new input patterns do not erase previously learned information. They are also adaptive because new information can be incorporated until full capacity of the architecture is utilized.

Proposed neural networks can be classified according to their ability to generalize. CMAC is a local generalizing neural network, while MLPs and recurrent MLPs are suitable for global generalization. RBF networks are placed between them. The choice for either one of the networks depends on the requirement for local generalization. When a strong local generalization is needed, a CMAC is most suitable. For global generalization, MLPs and recurrent MLPs provide a good alternative, combined with an improved weight adjustment algorithm.

24.2.3 Learning Principles and Rules

Adaptation (or machine learning) deals with finding weights (and sometimes a network topology) that will produce the desired behavior. Usually, the learning algorithm works from training examples, where each example incorporates correct input–output pairs (*supervised learning*). This learning form is based on the acquisition of mapping by the presentation of training exemplars (input–output data). Different than supervised learning, *reinforcement learning* considers the improvement of system performances by evaluating some realized control action that is included in the learning rules. Unsupervised learning in connectionist learning is when processing units respond only to interesting patterns on their inputs that are based on internal learning function.

The topology of the network during the training process can be fixed or variable based on evolution and regeneration principles.

The different iterative adaptation algorithms proposed so far are essentially designed in accordance with the *minimal disturbance principle:* Adapt to reduce output error for the current training pattern, with minimal disturbance to responses already learned. Two principal classes of algorithms can be distinguished:

Error-correction rules, alter the weights of a network to correct the error in the output response to the present input pattern.

Gradient-based rules, alter the weights of a network during each pattern presentation by a gradient descent with the objective of reducing mean-square error, averaged over training patterns.

The error-correction rules for networks often tend to be ad hoc. They are most often used when training objectives are not easily quantified, or when a problem does not lend itself to tractable analysis (for instance, networks that contain discontinuous functions, e.g., signum networks).

Gradient adaptation techniques are intended for minimization of the mean-square error associated with an entire network of adaptive elements:

$$e^2 = \sum_{t=1}^{T} \sum_{i=1}^{N_y} [e_i(t)]^2 \tag{24.11}$$

where $e_i^2(t)$ is the square error for particulary patterns.

The most practical and efficient algorithms typically work with one pattern presentation at a time. This approach is referred to as *pattern learning*, as opposite to *batch learning*, in which weights are adapted after presentation of all the training patterns (true *real-time learning* is similar to pattern learning, but it is performed with only one pass through the data). Similar, to the single-element case, in place of the true MSE function, the instantaneous sum squared error $e^2(t)$ is considered, which is the sum of the square errors at each of the N_y outputs of the network:

$$e^2(t) = \sum_{i=1}^{N_y} [e_i(t)]^2 \tag{24.12}$$

The corresponding instantaneous gradient is

$$E = \hat{\nabla}(t) = \frac{\partial e^2(t)}{\partial w(t)} \tag{24.13}$$

where $w(t)$ denotes a vector of all weights in the network. The steepest descent with the instantaneous gradient is a process presented by

$$w(t + 1) = w(t) + \Delta w(t)$$

$$\Delta w(t) = \mu(-\hat{\nabla}(t)) \tag{24.14}$$

The most popular method for estimating the gradient $\hat{\nabla}(t)$ is the backpropagation algorithm.

The backpropagation algorithm or generalized delta rule is the basic training algorithm for multilayer perceptrons. The basic analysis of an algorithm application will be shown using a three-layer perceptron (one hidden layer with a sigmoid function in the hidden and output layers). The main relations in the training process for one input–output pair $p = p(t)$ are given by the following relations:

$$s_2^p = W_{12}^{pT} u_1^p \quad s_2^p \varepsilon R^{L_1} \tag{24.15}$$

$$o_{2a}^p = 1/(1 + \exp(-s_{2a}^p)) \quad a = 1, \ldots, L_1 \quad o_{20}^p = 1 \tag{24.16}$$

$$s_3^p = W_{23}^{pT} o_2^p \quad s_3^p \varepsilon R^{N_y} \tag{24.17}$$

$$o_{3b}^p = 1/(1 + \exp(-s_{3b}^p)) \qquad b = 1, \ldots, N_y \tag{24.18}$$

$$y_c^p = o_{3c}^p \qquad c = 1, \ldots, N_y \tag{24.19}$$

where s_2^p, s_3^p are input vectors of the hidden and output layers of the network; o_2^p, o_3^p are output vectors of the hidden and output layers; $W_{12}^p = [w_{12ij\,Nu+1\times L1}(t)]$, $W_{23}^p = [w_{23ij_{L1+1\times N_y}}(t)]$ are weighting factors; w_{tuij} is the weighting factor that connects neuron j in layer t with neuron i in output layer

u; u_1^p is the input vector ($u_{10}^p = 1$; N_u-number of inputs; y^p is the output vector (N_y_ number of outputs; L_1 = number of neurons in a hidden layer).

The square error criterion can be defined as:

$$E = \sum_{p \in P} E^p = 0.5 \sum_{p \in P} \left| \hat{y}^p - y^p \right|^2 \tag{24.20}$$

where $\hat{y}^p$ is the desired value of the network output; y^p je output value of the networks; E^p is the value of the square criterion for one pair of input–output data; P is the set of input–output pairs.

The corresponding gradient component for the output layer is

$$\frac{\partial E}{\partial w_{23ij}} = \sum_{p \in P} \frac{\partial E^p}{\partial w_{23ij}} = \sum_{p \in P} \frac{\partial E_p}{s_{3i}^p} \frac{\partial s_{3i}^p}{\partial w_{23ij}} = -\sum_{p \in P} \delta_{3i}^p o_{2j}^p \tag{24.21}$$

$$\delta_{3i}^p = (\hat{y}_i^p - y_i^p) df_{3i}^p / ds_{3i}^p = (\hat{y}_i^p - y_i^p) f'_{3i}(s_{3i}^p) \tag{24.22}$$

where f_{gi} is the activation function for neuron i in layer g.

For the hidden layer, the gradient component is defined by:

$$\begin{aligned} \frac{\partial E}{\partial w_{12ij}} &= \sum_{p \in P} \frac{\partial E^p}{\partial w_{12ij}} = \sum_{p \in P} \frac{\partial E_p}{s_{2i}^p} \frac{\partial s_{2i}^p}{\partial w_{12ij}} \\ &= \sum_{p \in P} \sum_r \frac{\partial E^p}{\partial s_{3r}^p} \frac{\partial s_{3r}^p}{\partial o_{2i}^p} \frac{\partial o_{2j}^p}{\partial s_{2i}^p} \frac{\partial s_{2i}^p}{\partial w_{12ij}} \\ &= -\sum_{p \in P} \sum_r \delta_{3r}^p w_{23ri} f'_{2i}(s_{2i}^p) u_{1j}^p \\ &= -\sum_{p \in P} \delta_{2i}^p u_{1j}^p \end{aligned} \tag{24.23}$$

$$\delta_{2i}^p = \sum_r \delta_{3\tau}^p w_{23ri} f'_{2i}(s_{2i}^p) \tag{24.24}$$

Based on previous equations, starting from the output layer and going back, the error backpropagation algorithm is synthesized. The final version of the algorithm modified by weighting factors is defined by the following relations:

$$\delta_{3i}(t) = (\hat{y}_i(t) - y_i(t)) f'_{3i}(s_{3i}(t)) \tag{24.25}$$

$$\Delta w_{23ij}(t) = -\eta \frac{\partial E}{\partial w_{23ij}} = \eta \delta_{3i}(t) o_{2j}(t) \tag{24.26}$$

$$\delta_{2i}(t) = \sum_r \delta_{3r}(t) w_{23ri}(t) f'_{2i}(s_{2i}(t)) \tag{24.27}$$

$$\Delta w_{12ij}(t) = -\eta \frac{\partial E}{\partial w_{12ij}} = \eta \delta_{2i}(t) u_{1j}(t) \tag{24.28}$$

$$w_{23ij}(t+1) = w_{23ij}(t) + \Delta w_{23ij}(t) \tag{24.29}$$

$$w_{12ij}(t+1) = w_{12ij}(t) + \Delta w_{12ij}(t) \tag{24.30}$$

where η is the learning rate.

Also, numerous variants are used to speed up the learning process in the backpropagation algorithm. The one important extension is the *momentum technique* which involves a term proportional to the weight change from the previous iteration:

$$w(t+1) = w(t) + \Delta w(t)$$

$$\Delta w(t) = (1-\eta) \cdot \mu(-\hat{\nabla}(t)) + \eta \cdot \Delta w(t-1) \tag{24.31}$$

The momentum technique serves as a low-pass filter for gradient noise and is useful in situations when a clean gradient estimate is required, for example, when a relatively flat local region in the mean square error surface is encountered. All gradient-based methods are subject to convergence on local optima. The most common remedy for this is the sporadic addition of noise to the weights or gradients, as in simulated annealing methods. Another technique is to retrain the network several times using different random initial weights until a satisfactory solution is found. Backpropagation adapts the weights to seek the extremum of the objective function whose domain of attraction contains the initial weights. Therefore, both choice of the initial weights and the form of the objective function are critical to the network performance. The initial weights are normally set to small random values. Experimental evidence suggests choosing the initial weights in each hidden layer in a quasi-random manner, which ensures that at each position in a layer's input space the outputs of all but a few of its elements will be saturated, while ensuring that each element in the layer is unsaturated in some region of its input space.

There are more different learning rules for speeding up the convergence process of the backpropagation algorithm. One interesting method is using recursive least square algorithms and the extended Kalman approach instead of gradient techniques.[12]

The training procedure for the RBF networks involves a few important steps:

Step 1: Group the training patterns in M subsets using some clustering algorithm (k-means clustering algorithm) and select their centers c_i.
Step 2: Compute the widths, σ_i, ($i = 1, \ldots, m$), using some heuristic method (p-nearest neighbor algorithm).
Step 3: Compute the RBF activation functions $\phi_i(u)$, for the training inputs.
Step 4: Compute the weight vectors by least squares.

24.3 Neural Network Issues in Robotics

Possible applications of neural networks in robotics include various purposes suh as vision systems, appendage controllers for manufacturing, tactile sensing, tactile feedback gripper control, motion control systems, situation analysis, navigation of mobile robots, solution to the inverse kinematic problem, sensory-motor coordination, generation of limb trajectories, learning visuomotor coordination of a robot arm in 3D, etc.[5,11,16,38,39,43] All these robotic tasks can be categorized according to the type of hierarchical control level of the robotic system, i.e., neural networks can be applied at a strategic control level (task planning), at a tactic control level (path planning), and at an executive

control level (path control). All these control problems at different hierarchical levels can be formulated in terms of optimization or pattern association problems. For example, autonomous robot path planning and stereovision for task planning can be formulated as optimization problems, while on the other hand, sensor/motor control, voluntary movement control, and cerebellar model articulation control can be formulated as pattern association tasks. For pattern association tasks, neural networks in robotics can have the role of function approximation (modeling of input/output kinematic and dynamic relations) or the role of pattern classification necessary for control purposes.

24.3.1 Kinematic Robot Learning by Neural Networks

It is well known in robotics that control is applied at the level of the robot joints, while the desired trajectory is specified through the movement of the end-effector. Hence, a control algorithm requires the solution of the inverse kinematic problem for a complex nonlinear system (connection between internal and external coordinates) in real time. However, in general, the path in Cartesian space is often very complex and the end-effector location of the arm cannot be efficiently determined before the movement is actually made. Also, the solution of the inverse kinematic problem is not unique, because in the case of redundant robots there may be an infinite number of solutions. The conventional methods of solution in this case consist of closed-form and iterative methods. These are either limited only to a class of simple non-redundant robots or are time-consuming and the solution may diverge because of a bad initial guess. We refer to this method as the *position-based inverse kinematic control*. The *velocity-based inverse kinematic control* directly controls the joint velocity (determined by the external and internal velocities of the Jacobian matrix). Velocity-based inverse kinematic control is also called inverse Jacobian control.

The goal of kinematic learning methods is to find or approximate two previously defined mappings: one between the external coordinate target specified by the user and internal values of robot coordinates (position-based inverse kinematic control) and a second mapping connected to the inverse Jacobian of the robotic system (velocity-based inverse kinematic control).

In the area of position-based inverse kinematic control problems various methods have been proposed to solve them. The basic idea common to all these algorithms is the use of the same topology of the neural network (multilayer perceptron) and the same learning rule: the backpropagation algorithm. Although the backpropagation algorithms work for robots with a small number of degrees of freedom, they may not perform in the same way for robots with six degrees of freedom. In fact, the problem is that these methods are naive, i.e., in the design of neural network topology some knowledge about kinematic robot model has not been incorporated. One solution is to use a hybrid approach, i.e., a combination of the neural network approach with the classic iterative procedure. The iterative method gives the final solution in joint coordinates within the specified tolerance.

In the velocity-based kinematic approaches, the neural network has to map the external velocity into joint velocity. A very interesting approach has been proposed using the context-sensitive networks. It is an alternative approach to the reduction of complexity, as it proposes partition of the network input variables into two sets. One set (context input) acts as the input to a context network. The output of the context network is used to set up the weights of the function network. The function network maps the second set of input variables (function input) to the output. The original function to be learned is decomposed into a parameterized family of functions, each of which is simpler than the original one and is thus easier to learn.

Generally, the main problem in all kinematic approaches is accurately tracking a predetermined robot trajectory. As is known, in most kinematic connectionist approaches, the kinematic input/output mapping is learned offline and then control is attempted. However, it is necessary to examine the proposed solutions by learning control of manipulation robots in real-time, because the robots are complex dynamic systems.

24.3.2 Dynamic Robot Learning at the Executive Control Level

As a solution in the context of robot dynamic learning, neural network approaches provide the implementation tools for complex input/output relations of robot dynamics without analytic modeling. Perhaps the most powerful property of neural networks in robotics is their ability to model the whole controlled system itself. In this way the connectionist controller can compensate for a wide range of robot uncertainties. It is important to note that the application of the connectionist solution for robot dynamic learning is not limited only to noncontact tasks. It is also applicable to essential contact tasks, where inverse dynamic mapping is more complex, because dependence on contact forces is included.

The application of the connectionist approach in robot control can be divided according to the type of learning into two main classes: neurocontrol by supervised and neurocontrol by unsupervised learning.

For the first class of neurocontrol a teacher is assumed to be available, capable of teaching the required control. This is a good approach in the case of a human-trained controller, because it can be used to automate a previously human-controlled system. However, in the case of automated linear and nonlinear teachers, the teacher's design requires *a priori* knowledge of the dynamics of the robot under control. The structure of the supervised neurocontrol involves three main components, namely, a teacher, the trainable controller, and the robot under control.[1] The teacher can be either a human controller or another automated controller (algorithm, knowledge-based process, etc.). The trainable controller is a neural network appropriate for supervised learning prior to training. Robot states are measured by specialized sensors and are sent to both the teacher and the trainable controller. During control of the robot by the teacher, the control signals and the state variables of the robot are sampled and stored for neural controller training. At the end of successful training the neural network has learned the right control action and replaces the teacher in controlling the robot.

In unsupervised neural learning control, no external teacher is available and the dynamics of the robot under control is unknown and/or involves severe uncertainties. There are different principal architectures for unsupervised robot learning.

In the *specialized learning architecture* (Figure 24.3), the neural network is tuned by the error between the desired response and actual response of the system. Another solution, *generalized learning architecture* (Figure 24.4), is proposed in which the network is first trained offline based on control error, until good convergence properties are achieved, and then put in a real-time feedforward controller where the network continues its adaptation to system changes according to specialized learning procedures.

The most appropriate learning architectures for robot control are *feedback-error learning architecture* and *adaptive learning architecture*. The feedback-error learning architecture (Figure 24.5) is an exclusively online achitecture for robot control that enables simultaneous processing of learning and control. The primary interest is learning an inverse dynamic model of robot mechanism for the tasks with holonomic constraints, where exact robot dynamics is generally unknown. The neural network as part of feedforward control generates necessary driving torques in robot joints as a nonlinear mapping of robot desired internal coordinates, velocities, and accelerations:

$$P_i = g(w_{jk}^{ab}, q_d, \dot{q}_d, \ddot{q}_d) \quad i = 1, \ldots, n. \tag{24.32}$$

where $P_i \varepsilon R^n$ is a joint-driving torque generated by a neural network; w_{jk}^{ab} are adaptive weighting factors between neuron j in $a - th$ layer and neuron k in $b - th$ layer; g is nonlinear mapping.

According to the integral model of robotic systems, the decentralized control algorithm with learning has the form

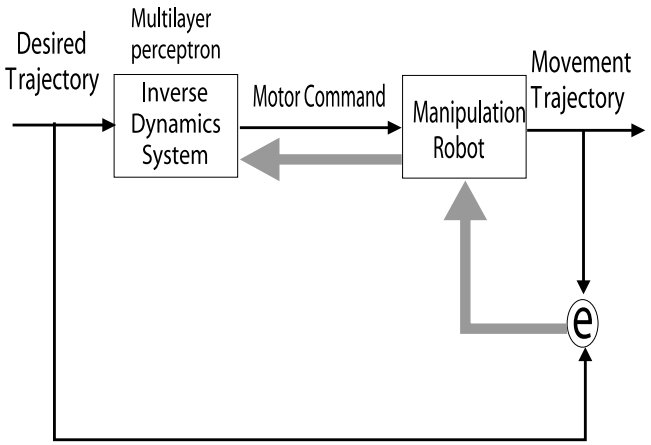

FIGURE 24.3 Specialized learning architecture.

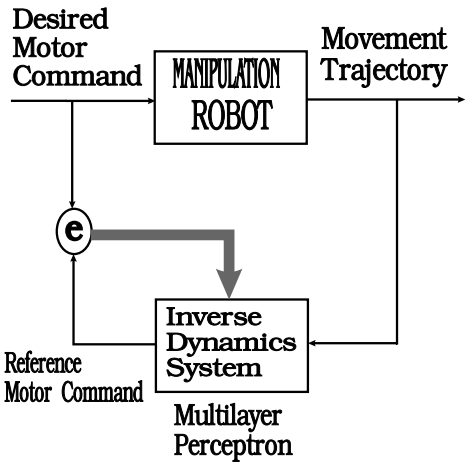

FIGURE 24.4 Generalized learning architecture.

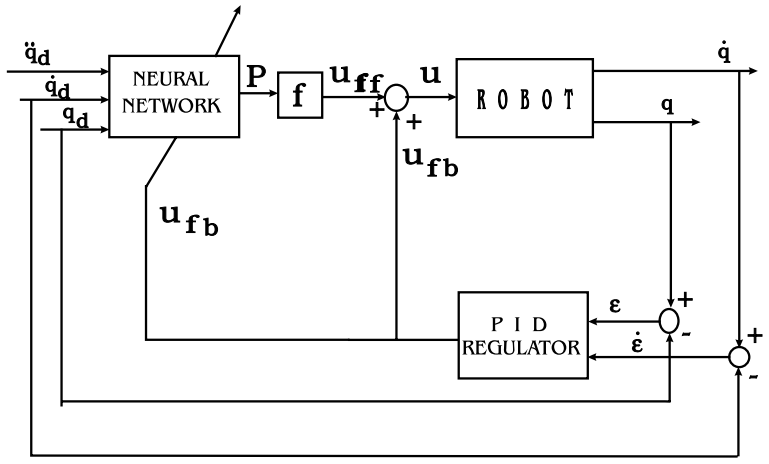

FIGURE 24.5 Feedback-error learning architecture.

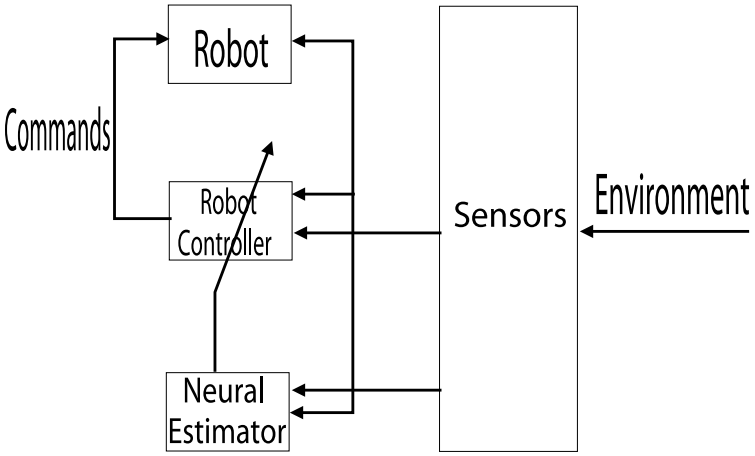

FIGURE 24.6 Sensor-based learning architecture.

$$u_i = u_{ii}^{ff} + u_{ii}^{fb} \qquad i = 1, \ldots, n. \tag{24.33}$$

$$u_i = f_i(q_d, \dot{q}_d, \ddot{q}_d, P) - KP_{ii}\varepsilon_i - KD_{ii}\dot{\varepsilon}_i - KI_{ii}\int \varepsilon_i dt \quad i = 1, \ldots, n. \tag{24.34}$$

where f_i is the nonlinear mapping which describes the nature of the robot actuator model; $KP, KF, KI \varepsilon R^{n \times n}$ are position, velocity, and integral local feedback gains, respectively; $\varepsilon \varepsilon R^n$ is the feedback error. Training and learning the proposed connectionist structure can be accomplished using the well-known backpropagation algorithm.[9] In the process of training we can use the feedback control signal:

$$e_i^{bp} = u_i^{fb} \quad i = 1, \ldots, n \tag{24.35}$$

where $e_i^{bp} \varepsilon R^n$ is the output error for the backpropagation algorithm.

A more recent and sophisticated learning architecture (adaptive learning architecture) involves the neural estimator that identifies some robot parameters using available information from robot sensors (Figure 24.6). Based on information from the neural estimator, the robot controller modifies its parameters and then generates a control signal for robot actuators. The robot sensors observe the status of the system and make available information and parameters to the estimator and robot controller. Based on this input, the neural estimator changes its state, moving in the state space of its variables. The state variables of the neural estimator correspond exactly to the parameters of robot controller. Hence, the stable-state topology of this space can be designed so that the local minima correspond to an optimal law.

The special reactive control strategy applied to robotic dynamic control[51] can be characterized as reinforcement learning architecture. In contrast to the supervised learning paradigm, the role of the teacher in reinforcement learning is more evaluative than instructional. The teacher provides the learning system with an evaluation of the system performance of the robot task according to a certain criterion. The aim of this learning system is to improve its performance by generating appropriate outputs. In Gullapalli[51] a stochastic reinforcement learning approach with application in robotics for learning functions with continuous outputs is presented. The learning system computes real-valued output as some function of a random activation generated using normal distribution. The parameters of normal distribution are the mean and the standard deviation that

depend on current input patterns. The environment evaluates the unit output in the context of input patterns and sends a reinforcement signal to the learning system. The aim of learning is to adjust the mean and the standard deviation to increase the probability of producing the optimal real value for each input pattern.

A special group of dynamic connectionist approaches is the methods that use the "black-box" approach in the design of neural network algorithms for robot dynamic control. The "black box" approach does not use any *a priori* experience or knowledge about the inverse dynamic robot model. In this case it is a multilayer neural network with a sufficient number of hidden layers. All we need to do is feed the multilayer neural network the necessary information (desired positions, velocities, and accelerations at the network input and desired driving torque at the network output) and let it learn by test trajectory. In Ozaki et al.[48] a nonlinear neural compensator that incorporates the idea of computed torque method is presented. Although the pure neural network approach without knowledge about robot dynamics may be promising, it is important to note that this approach will not be very practical because of the high dimensionality of input–output spaces. Bassi and Bekey[10] use the principle of functional decomposition to simplify robot dynamics learning. This method includes *a priori* knowledge about robot dynamics which, instead of being specific knowledge corresponding to a certain type of robot models, incorporates common invormation about robot dynamics. In this way, the unknown input–output mapping is decomposed into simpler functions that are easier to learn because of smaller domains. In Katić and Vukobratović,[12] similar ideas in the development of the fast learning algorithm were used with decomposition at the level of internal robot coordinates, velocities, and accelerations.

The connectionist approach is very efficient in the case of robots with flexible links or for a flexible materials handling system by a robotic manipulators where the parameters are not exactly known and the learning capability is important to deal with such problems. Because of the complex nonlinear dynamical model, the recurrent neural network is very suitable for compensating flexible effects.

With recent extensive research in the area of robot position/force control, a few connectionist learning algorithms for constrained manipulation have been proposed. We can distinguish two essential different approaches: one, whose aim is the transfer of human manipulation skills to robot controllers, and the other, in which the manipulation robot is examined as an independent dynamic system in which learning is achieved through repetition of the work task.

The principle of transferring human manipulation skill (Figure 24.7) has been developed in the papers of Asada and co-workers.[18] The approach is based on the acquisition of manipulation skills and strategies from human experts and subsequent transfer of these skills to robot controllers. It is essentially a playback approach, where the robot tries to accomplish the working task in the same way as an experienced worker. Various methods and techniques have been evaluated for acquisition and transfer of human skills to robot controllers.

This approach is very interesting and important, although there are some critical issues related to the explicit mathematical description of human manipulation skill because of the presence of subconscious knowledge and inconsistent, contradictory, and insufficient data. These data may cause system instability and wrong behavior by the robotic system. As is known, dynamics of the human arm and a robot arm are essentially different, and therefore it is not possible to apply human skill to robot controllers in the same way. The sensor system for data acquisition of human skill can be insufficient for extracting a complete set of information necessary for transfer to robot controllers. Also, this method is inherently an offline learning method, whereas for robot contact tasks online learning is a very important process because of the high level of robot interaction with the environment and unpredictable situations that were not captured in the skill acquisition process.

The second group of learning methods, based on autonomous online learning procedures with working task repetition, have also been evaluated through several algorithms. The primary aim is to build internal robot models with compensation of the system uncertainties or direct adjustment of control signals or parameters (reinforcement learning). Using a combination of different intelligent paradigms (fuzzy + neuro) Kiguchi and Fukuda[25] proposed a special algorithm for approach,

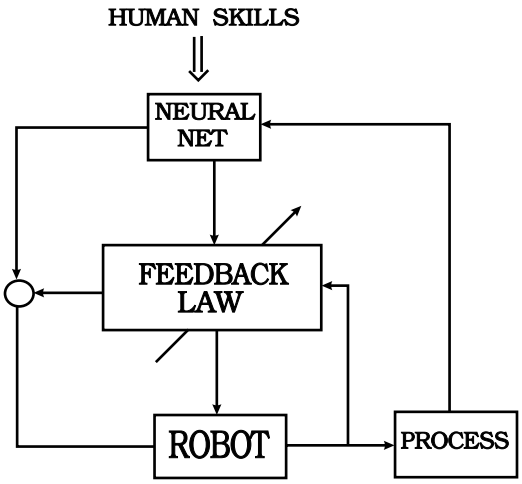

FIGURE 24.7 Transfer of human skills to robot controllers by the neural network approach.

contact, and force control of robot manipulators in an unknown environment. In this case, the robot manipulator controller, which approaches, contacts, and applies force to the environment, is designed using fuzzy logic to realize human-like control and then modeled as a neural network to adjust membership functions and rules to achieve the desired contact force control.

As another exposed problem in control robotic contact tasks, the connectionist approach is used for dynamic environment identification. A new learning control concept based on neural network classification of unknown dynamic environment models and neural network learning of robot dynamic model has been proposed.[13] The method classifies characteristics of environments by using multilayer perceptrons based on the first neural network, and then determines the control parameters for compliance control using the estimated characteristics. Simultaneously, using the second neural network, compensation of robot dynamic model uncertainties is accomplished. The classification capability of the neural classifier is realized by an efficient offline training process. It is important that the pattern classification process can work in an online manner as a part of selected compliance control algorithm.

The first objective is the application of connectionist structures to fast online learning of robotic system uncertainties as a part of the stabilizing control algorithm mentioned previously. The role of the connectionist structure has a broader sense, because its aim is to compensate possible uncertainties and differences between real robot dynamics and assumed dynamics defined by the user in the process of control synthesis. Hence, to achieve good tracking performance in the presence of model uncertainties, a fixed non-recurrent multilayer perceptron is integrated into the non-learning control law with the desired quality of transient processing for interaction force.

In this case, compensation by neural network is connected to the uncertainties of robot dynamic model. But, the proposed learning control algorithm does not work in a satisfactory way if there is no sufficiently accurate information about the type and parameters of the robot environment model. Hence, to enhance connectionist learning of the general robot-environment model, a new method is proposed whose main idea is using a neural network approach through an offline learning process and online sufficiently exact classification of robot dynamic environment. The neural network classifier based on a four-layer perceptron is chosen due to good generalization properties. Its objective is to classify the model profile and parameters of environment in an online manner. In the acquisition process, based on real-time realization of proposed contact control algorithms and using previously chosen sets of different working environments and model profiles of working environments, some force data from force sensors are measured, calculated, and stored as special input patterns for training the neural network. On the other side, the acquisition process must be

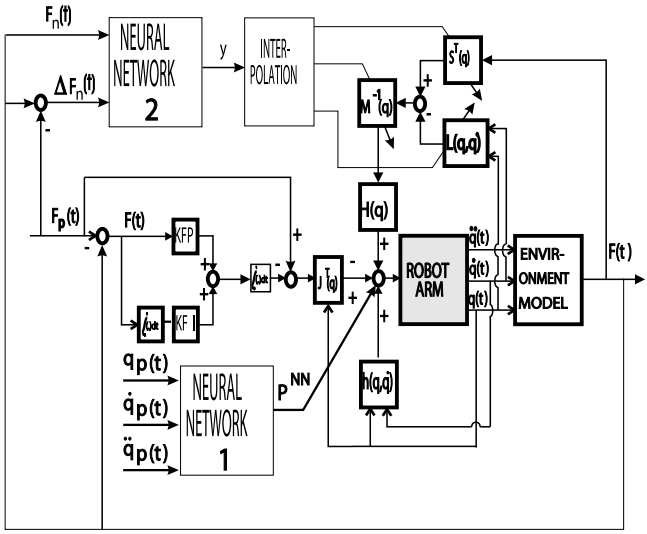

FIGURE 24.8 Scheme of the connectionist control law stabilizing interaction force.

accomplished using various robot environments, starting with the environment with a low level of system characteristics (for example, with a low level of environment stiffness) and ending with an environment with a high level of system characteristics (with high level of environment stiffness). As another important characteristic in the acquisition process, different model profiles of the environment are used based on additional damping and stiffness members that are added to the basic general impedance model.

After that, during the extensive offline training process, the neural network receives a set of input–output patterns, where the input variables form a previously collected set of force data. As a desired output, the neural network has a value between 0 and a value defined by the environment profile model (the whole range between 0 and 1) that exactly defines the type of training robot environment and environment model. The aim of connectionist training is for the real output of the neural network for given inputs to be exact or very close to the desired output value determined for an appropriate training robot environment model.

After the offline training process with different working environments and different environment model profiles, the neural classifier is included in the online version of the control algorithm to produce some value at the network's output between 0 and 1. In the case of an unknown environment, information from the neural classifier output can be utilized efficiently for calculating the necessary environment parameters by linear interpolation procedures. Figure 24.8 shows the overall structure of the proposed algorithm.

24.3.3 Sensor-Based Robot Learning

A completely different approach of connectionist learning uses sensory information for robot neural control. Sensor-based control is a very efficient method in overcoming problems with robot model and environment uncertainties, because sensor capabilities help in the adaptation proces without explicit control intervention. It is adaptive sensor-motor coordination that uses various mappings given by the robot sensor system. Particular attention has been paid to the problem of visuo-motor coordination, in particular for eye–head and arm–eye systems. In general, in visuo-motor coordination by neural networks, visual images of the mechanical parts of the systems can be directly related to posture signals. However, tactile-motor coordination differs significantly from visuo-motor because the intrinsic dependency on the contacted surface. The direct association of tactile sensations with positioning of the robot end-effector is not feasible in many cases, hence it is very

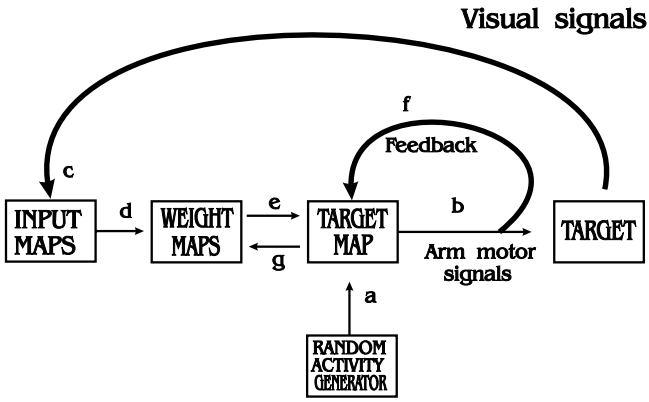

FIGURE 24.9 Sensory-motor circular reaction.

important to understand how a given contact condition will be modified by motor actions. The task of the neural network in these cases is to estimate the direction of a feature-enhancing motor action on the basis of modifications in the sensed tactile perception.

After many years of being thought impractical in robot control, it was demonstrated that CMAC could be very useful in learning state-space dependent control responses.[56] A typical demonstration of CMAC application in robot control involves controlling an industrial robot using a video camera. The robot's task is to grasp an arbitrary object lying on a conveyor belt with a fixed orientation or to avoid various obstacles in the workspace. In the learning phase, visual input signals about the objects are processed and combined into a target map through modifiable weights that generate the control signals for the robot's motors. The errors between the actual motor signals and the motor signals computed from the camera input are used to incrementally change the weights. Kuperstain[33] has presented a similar approach using the principle of sensory-motor circular reaction (Figure 24.9). This method relies on consistency between sensory and motor signals to achieve unsupervised learning. This learning scheme requires only availability of the manipulator, but no formal knowledge of robotic kinematics. Opposite to previously mentioned approaches for visuo-motor coordination, Rucci and Dario[34] experimentally verified autonomous learning of tactile-motor coordination by a Gaussian network for a simple robotic system composed of a single finger mounted on a robotic arm.

24.4 Fuzzy Logic Approach

24.4.1 Introduction

The basic idea of fuzzy control was conceived by L. Zadeh in his papers from 1968, 1972, and 1973.[59,61,62] The heart of his idea is describing control strategy in linguistic terms. For instance, one possible control strategy of a single-input, single-output system can be described by a set of control rules:

If (error is positive and error change is positive), then
  control change = negative
Else if (error is positive and error change is negative), then
  control change = zero
Else if (error is negative and error change is positive), then
  control change = zero
Else if (error is negative and error change is negative), then
  control change = positive

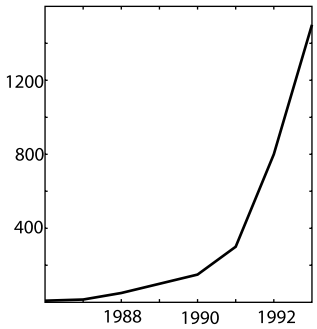

FIGURE 24.10 Estimated number of commercial applications of fuzzy systems.

Further refining of the strategy might take into account cases when, e.g., the error and error change are small or big. Such a procedure could make it possible to describe the control strategy used, e.g., by trained operators when controlling a system manually.

Statements in natural language are intrinsically imprecise due to the imprecise manner of human reasoning. Development of techniques for modeling imprecise statements is one of the main issues in implementation of automatic control systems based on using linguistic control rules. With fuzzy controllers, modeling of linguistic control rules (as well as derivation of control action on the basis of given set of rules and known state of the controlled system) is based on the theory of fuzzy sets introduced by Zadeh in 1965.[58]

In 1974, Mamdani described the first application of fuzzy set theory to automatic control.[30] However, almost 10 years passed before broader interest was reestablished for fuzzy logic and its applications in automatic control. The number of reported fuzzy applications has been increasing exponentially (Figure 24.10). Current applications based on fuzzy control appear in such diverse areas as the automatic control of trains, road cars, cranes, lifts, nuclear plants, home appliances, etc. Commercial applications in robotics still do not exist; however, numerous research efforts promise that fuzzy robot control systems will be developed, notably in the fields of robotized part processing, assembly, mobile robots, and robot vision systems.

Thanks to its ability to manipulate imprecise and incomplete data, fuzzy logic offers the possibility of incorporating expertise into automatic control systems. Fuzzy logic already has proven itself useful in cases where the process is too complex to be analyzed by conventional quantitative techniques, or where the available information is qualitative, imprecise, or unreliable. Considering that it is based on precise mathematical theory, fuzzy logic additionally offers the possibility of integrating heuristic methods with conventional techniques for analysis and synthesis of automatic control systems, thus facilitating further refinement of fuzzy control-based systems.

24.4.2 Mathematical Foundations

24.4.2.1 Fuzzy Sets

At the heart of fuzzy set theory is the notion of fuzzy sets that are used to model statements in natural (or artificial) language. Fuzzy set is a generalization of classical (crisp) sets. The classical set concept assumes that it is possible to divide particles of some universe into two parts: those that are members of the given set, and those that are not. This partitioning process can be described by means of a characteristic *membership function*. For a given universe of discourse X and a given set A, membership function $\mu_A(\cdot)$ assigns a value to each particle $x \in X$ so that

$$\mu_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}$$

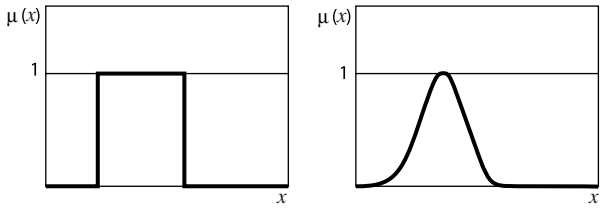

FIGURE 24.11 Membership functions of conventional and fuzzy sets.

With fuzzy sets, the set's boundary is not strict between the members and nonmembers. This softening of the boundary is defined mathematically using the *membership degree function,* which assigns each particle a value that indicates the degree of membership in the given set (see Figure 24.11). Accordingly, fuzzy set $\tilde{A}$ in the universe of discourse X is defined by its degree of membership function $\mu_{\tilde{A}}(\cdot)$ of the form:

$$\mu_{\tilde{A}}: X \mapsto [0, 1].$$

For each fuzzy set, its support can be defined. The support of fuzzy set $\tilde{A}$ is an ordinary set A that contains all elements from the universe X with nonzero membership degrees in $\tilde{A}$:

$$\text{supp}(\tilde{A}) = \{x \in X: \mu_{\tilde{A}}(x) > 0\}.$$

The notion of support allows a formal definition of empty fuzzy sets. An *empty fuzzy set* is a fuzzy set with empty support.

It is customary to represent fuzzy sets by fuzzy singletons. A *fuzzy singleton* is a fuzzy set for which its support is a single particle x from the universe X. If fuzzy set $\tilde{A}$ has a finite support $\text{supp}(\tilde{A}) = \{x_1, x_2, \ldots, x_n\}$ with degrees of membership $\mu_{\tilde{A}}(x_i)$, $i = 1, 2, \ldots, n$, such a fuzzy set is conveniently written as:

$$\tilde{A} = \mu_{\tilde{A}}(x_1)/x_1 + \mu_{\tilde{A}}(x_2)/x_2 + \cdots + \mu_{\tilde{A}}(x_n)/x_n = \sum_{i=1}^{n} \mu_{\tilde{A}}(x_i)/x_i$$

Here, the plus sign indicates that pairs $\mu_{\tilde{A}}(x_i)/x_i$ collectively form the definition of fuzzy set $\tilde{A}$. If universe X is an interval of real numbers, then the following notation for fuzzy set $\tilde{A}$ in X is customary:

$$\tilde{A} = \int_X \mu_{\tilde{A}}(x)/x$$

The notions of fuzzy subsets and equality between fuzzy sets are also defined in terms of membership degree functions. Fuzzy set $\tilde{A}$ is said to be a *subset* of $\tilde{B}$ if all particles $x \in X$ have degrees of membership to $\tilde{A}$ lower or equal to their degrees of membership to $\tilde{B}$:

$$\tilde{A} = \tilde{B} \text{ iff } \mu_{\tilde{A}}(x) \le \mu_{\tilde{B}}(x) \text{ for all } x \in X$$

Fuzzy sets are *equal* if their membership functions are equal for all elements in the universe of discourse:

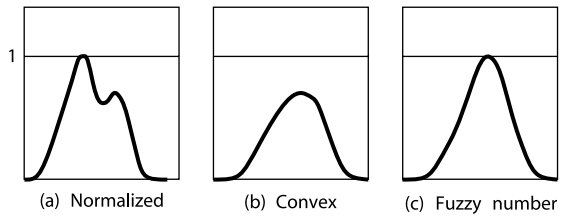

FIGURE 24.12 Examples of fuzzy sets.

$$\tilde{A} = \tilde{B} \text{ iff } \mu_{\tilde{A}}(x) = \mu_{\tilde{B}}(x) \text{ for all } x \in X$$

An important class of fuzzy sets is normalized fuzzy sets. A fuzzy set $\tilde{A}$ is said to be normalized if its height $h(\tilde{A})$, defined as the largest degree of membership attained by elements in its support, is equal to 1:

$$h(\tilde{A}) \equiv \max_X \mu_{\tilde{A}}(x) = 1$$

The value $m \in X$ for which $\mu_{\tilde{A}}(m) = h(\tilde{A})$ is called the *modal value* of the fuzzy set.

Fuzzy set $\tilde{A}$ in Euclidean space R^n is *convex* if, for any vectors $\mathbf{x}, \mathbf{y} \in R^n$, the following is valid:

$$\mu_{\tilde{A}}(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \geq \min[\mu_{\tilde{A}}(\mathbf{x}), \mu_{\tilde{A}}(\mathbf{y})]$$

Fuzzy sets that are normalized, convex, and, additionally, have a piecewise continuous membership degree function, are denoted as *fuzzy intervals*. A special class of fuzzy intervals is fuzzy numbers. A *fuzzy number* is a fuzzy interval with an unique modal value. The concept of fuzzy numbers is based on fuzzy artihmetic that may be considered a generalization of classical arithmetic. Examples of membership functions of normalized, convex fuzzy sets, and fuzzy numbers are shown in Figure 24.12.

24.4.2.2 Operations on Fuzzy Sets

The basic principle for generalization of classical mathematical concepts to the field of fuzzy sets is known as the principle of extension.[63] Formally, given a function $f: X \to Y$, mapping elements of ordinal set X into elements of set Y, and an arbitrary fuzzy set $\tilde{A} \in \tilde{P}(X)$, e.g.,

$$\tilde{A} = \mu_1/x_1 + \mu_2/x_2 + \cdots + \mu_n/x_n$$

the principle of extension states that the following relation has to be preserved:

$$\begin{aligned} f(\tilde{A}) &\triangleq f(\mu_1/x_1 + \mu_2/x_2 + \cdots + \mu_n/x_n) \\ &= \mu_1/f(x_1) + \mu_2/f(x_2) + \cdots + \mu_n/f(x_n) \end{aligned}$$

In other words, operations on fuzzy sets should preserve important properties of operations on classical sets. Unfortunately, it turns out that it is not possible to define of basic fuzzy set operations that would preserve all the important properties of the corresponding operations on classical sets. For example, it is shown that arbitrary fuzzy complement, union, and intersection operations satisfying the law of contradiction and law of excluded middle are not distributive. Therefore, the

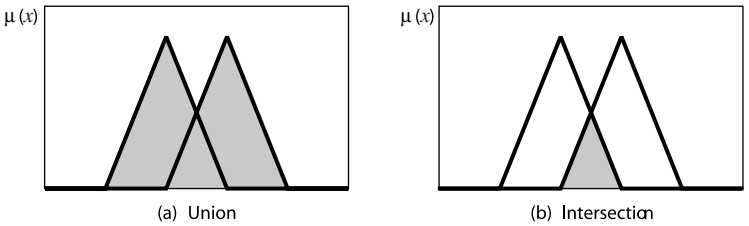

FIGURE 24.13 Standard operations on fuzzy sets.

choice of basic fuzzy set operations has to be made by considering the context in which these operations will be carried out. The most often used set of basic standard operations of fuzzy set theory is (see Figure 24.13):

$$\textit{Complement:} \quad \mu_{\bar{\tilde{A}}}(x) = 1 - \mu_{\tilde{A}}(x)$$

$$\textit{Union:} \quad \mu_{\tilde{A} \cup \tilde{B}}(x) = \max[\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)]$$

$$\textit{Intersection:} \quad \mu_{\tilde{A} \cap \tilde{B}}(x) = \min[\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)]$$

Fuzzy set theory based on such defined operators is usually referred to as *possibility theory*. However, in some situations, different definitions of basic fuzzy set operators are preferable. For example, a union $\tilde{A} \cup \tilde{B}$ intuitively is a disjunction of the concepts represented by $\tilde{A}$ and $\tilde{B}$. Additionally, the notion of union normally implies a certain level of interchangeability between the concepts represented by its arguments. On the other hand, a standard union max operator is rigid in the sense that it does not assume such an interchangeability. If the union were specified by the function

$$f_u: [0, 1] \times [0, 1] \rightarrow [0, 1]$$

that assigns a value $f_u[\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)]$ to given pair of membership degrees $\mu_{\tilde{A}}(x)$ and $\mu_{\tilde{B}}(x)$, then the intuitive meaning of the union implies the following relation:

$$f_u[\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)] \geq \max[\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)]$$

It is evident that standard union operation, defined as $\max[\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)]$, yields the lowest possible degree of membership. For this reason, in some cases, alternative formulations are used in place of the max operator. All potential formulations $f_u(\cdot)$ are required to satisfy the minimum axiomatic conditions:

U1. *Boundary conditions:* $f_u(0, 0) = 0$ and $f_u(0, 1) = f_u(1, 0) = f_u(1, 1) = 1$
U2. *Commutativity:* $f_u(x, y) = f_u(y, x)$
U3. *Monotony:* if $x \leq x'$ and $y \leq y'$, then $f_u(x, y) \leq f_u(x', y')$
U4. *Associativity:* $f_u(f_u(x, y), z) = f_u(x, f_u(y, z))$

The functions satisfying these axioms are called *triangular conorms* (*t-conorms*). Evidently, the standard union operation is a t-conorm. Other t-conorms are proposed as well, such as algebraic sum, bounded sum, etc.

Fuzzy intersection $\tilde{A} \cap \tilde{B}$ intuitively denotes a conjunction of concepts represented by $\tilde{A}$ and $\tilde{B}$. As in the case of union, the intersection operation can be specified using the function:

$$f_i: [0, 1] \times [0, 1] \rightarrow [0, 1]$$

The minimum axiomatic skeleton that functions $f_i(\cdot)$ have to satisfy to qualify as candidates for defining fuzzy intersection consists of conditions:

I1. *Boundary conditions:* $f_i(1, 1) = 1$ and $f_i(0, 0) = f_i(0, 1) = f_i(1, 0) = 0$
I2. *Commutativity:* $f_i(x, y) = f_i(y, x)$
I3. *Monotony:* if $x \le x'$ and $y \le y'$, then $f_i(x, y) \le f_i(x', y')$
I4. *Associativity:* $f_i(f_i(x, y), z) = f_i(x, f_i(y, z))$

The functions satisfying axioms I1–I3 are called *triangular norms* (*t-norms*). Obviously, the standard min operation is a t-norm.

Analogous to the case of the union, intersection of fuzzy sets normally implies a certain requirement level for the simultaneous satisfaction of concepts represented by its arguments. On the other hand, the standard min operation is rigid in the sense that it does not account for the benefits of simultaneous memberships. Hence, alternative t-norms are proposed in which different intensities of intersections are achieved: algebraic product, bounded product, etc. Standard min operation is the upper bound of the possible intersection operations (the weakest intersection).

24.4.2.3 Fuzzy Relations

Fuzzy relations are generalizations of the classical concept of relations among elements of two or more sets. Additionally, fuzzy relations allow the specification of different levels of strength of association among individual elements. The levels of association are represented by degrees of membership to the fuzzy relations, in the same manner as the degree of membership to a fuzzy set is represented.

Formally, a fuzzy relation among elements of ordinary sets $X_1, X_2, \ldots, X_n$ is a fuzzy subset $\tilde{R} = \tilde{R}(X_1, X_2, \ldots, X_n)$ of Cartesian product $X_1 \times X_2 \times \ldots \times X_n$ and it is defined by the membership degree function:

$$\mu_{\tilde{R}}: X_1 \times X_2 \times \ldots \times X_n \to [0,1]$$

Thus, tuples $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in X_1 \times X_2 \times \ldots \times X_n$ may have different degrees of membership $\mu_{\tilde{R}}(x_1, x_2, \ldots, x_n) \in [0,1]$ to the fuzzy relation.

When the sets $X_1, X_2, \ldots, X_n$ are finite, fuzzy relation $\tilde{R}(X_1, X_2, \ldots, X_n)$ is suitably represented by an *n*-dimensional *membership matrix,* whose elements show the degree to which the individual tuples belong to a given fuzzy relation. For instance, binary fuzzy relation $\tilde{R}(X, Y)$ between sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ is conveniently represented by the matrix:

$$\tilde{\mathbf{R}} = \begin{bmatrix} \mu_{x_1,y_1} & \cdots & \mu_{x_1,y_m} \\ \vdots & \ddots & \vdots \\ \mu_{x_n,y_1} & \cdots & \mu_{x_n,y_m} \end{bmatrix}$$

For a given family of sets $\tilde{A}_1, \tilde{A}_2, \ldots, \tilde{A}_n$, defined in the universes $X_1, X_2, \cdots, X_n$, the Cartesian product of fuzzy sets:

$$\tilde{A}_1 \times \tilde{A}_2 \times \cdots \times \tilde{A}_n$$

is a fuzzy set in the universe of discourse $X_1 \times X_2 \times \ldots \times X_n$. Consequently, the Cartesian product is an *n*-ary fuzzy relation with the degree of membership function defined by:

$$\mu_{\tilde{A}_1 \times \tilde{A}_2 \times \cdots \times \tilde{A}_n}(x_1, x_2, \ldots, x_n) = \mu_{\tilde{A}_1}(x_1) * \mu_{\tilde{A}_2}(x_2) * \cdots * \mu_{\tilde{A}_n}(x_n)$$

for all $x_1 \in X_1$, $x_2 \in X_2$, …, $x_n \in X_n$, where the sign $*$ denotes one of the triangular norms (i.e., the intersection operation).

Among the operations over fuzzy relations, compositions of binary relations are of special significance. For ordinary binary relations $P(X, Y)$ and $Q(Y, Z)$, defined in the common set Y, the *composition* of P and Q:

$$R(X, Z) = P(X, Y) \circ Q(Y, Z)$$

is defined as a subset $R \subseteq X \times Z$ such that

$$(x, z) \in R \text{ iff there exists } y \in Y \text{ for which } (x, y) \in P \text{ and } (x, z) \in Q$$

The concept of composition is extended to fuzzy relations in a number of ways aimed at preserving important properties of corresponding compositions of classical relations. The most important types of compositions of binary fuzzy relations are

- *Max–min composition.* Denoted by $\tilde{P}(X,Y) \circ \tilde{Q}(Y,Z)$, this operation is defined by

$$\mu_{\tilde{P}\circ\tilde{Q}}(x,z) = \max_{y\in Y} \min[\mu_{\tilde{P}}(x,y), \quad \mu_{\tilde{Q}}(y,z)]$$

  Thus, the strength of the relation between elements x and z is equal to the strength of the strongest chain between these elements, whereas the strength of each chain x–y–z is equal to the strength of its weakest link.
- *Max–product composition.* The composition is denoted by $\tilde{P}(X,Y) \circ \tilde{Q}(Y,Z)$ and defined by:

$$\mu_{\tilde{P}\circ\tilde{Q}}(x,z) = \max_{y\in Y}[\mu_{\tilde{P}}(x,y) \cdot \mu_{\tilde{Q}}(y,z)]$$

The max–min and max–product compositions may be regarded as specializations of the more general *sup-star composition,* denoted by $\tilde{P}(X,Y) \circ \tilde{Q}(Y,Z)$ and defined by

$$\mu_{\tilde{P}\circ\tilde{Q}}(x,z) = \sup_{y\in Y}[\mu_{\tilde{P}}(x,y) \ * \ \mu_{\tilde{Q}}(y,z)]$$

where the sign $*$ represents any triangular norm, and the sup operator denotes supremum (the lowest upper bound).

Compositions of binary relations in finite sets may be efficiently realized using membership matrices. For example, the composition $\tilde{P} \circ \tilde{Q}$ can be calculated as a matrix product:

$$\tilde{\mathbf{P}} \cdot \tilde{\mathbf{Q}}$$

where multiplication is replaced by the min, and addition by the max operator.

24.4.2.4 Fuzzy Logic

Fuzzy logic is a discipline comprising formal principles of approximate reasoning.[64] Its main issue is modeling of imprecise modes of human reasoning in conditions characterized by unreliability and imprecision, whereby the theory of fuzzy sets is used as a basic methodology.

Fuzzy logic is an extension to classical logic, in which the basic objects are logical propositions that may take one of the possible values of truth: true or false, i.e., 1 or 0. Contrary to classical formal systems, fuzzy logic allows evaluation of the truth of a proposition as, e.g., a real number in interval [0,1]. The basis of fuzzy logic is the theory of fuzzy sets. For example, the characterization

of fuzzy set $\tilde{A}$ with membership function $\mu_{\tilde{A}}(x)$, $x \in X$, can be interpreted as the truth value of the proposition:

$$x \text{ is element of } \tilde{A}$$

To enable work with imprecise propositions, fuzzy logic permits use of:

- *Fuzzy predicates.* Truth values of an imprecise predicate $P(x)$ (e.g., x is *small, big,* etc.) can be described for any $x \in X$ by the fuzzy set with membership function $\mu_P(x)$, determined by the predicate $P(\cdot)$.
- *Fuzzy truth values.* Fuzzy sets, defined on the interval [0, 1], can be used to describe different levels of truth (e.g., *fairly true, completely false,* etc.).
- *Fuzzy quantifiers.* In addition to the usual quantifiers from classical logic ($\forall$, $\exists$), imprecise statements may contain imprecise quantifiers (e.g., *sometimes, almost always*) represented by fuzzy numbers.
- *Fuzzy modifiers.* Different forms of fuzzy modifiers (*probably, fairly,* etc.) can be described by utilizing special operations on fuzzy sets representing the modified propositions.

The central problem of *quantitative fuzzy semantics* is calculating the meaning of *linguistic variables,* i.e., the variables whose values are sentences in a specific (natural or artificial) language.[60] The linguistic variable can be regarded as a variable whose value is a fuzzy number (the *meaning* of the variable) or as a variable whose values are linguistically defined (the *label* of the variable).[63] Generally, the label of a linguistic variable is obtained by concatenating the terms of the language according to some rules. In simple cases, these terms can be divided into four categories:

1. *Primary terms* that represent labels of specific fuzzy sets
2. Negation *not* and connectives *or* and *and*
3. *Modifiers* that modify the basic concept to which they are applied (e.g., *very, extremely,* etc.)
4. *Markers,* such as parentheses

Negation *not* and connectives *or* and *and* may be considered labels of the corresponding operations on fuzzy sets:

- *Complement* $\overline{\tilde{A}}$ that represents the fuzzy concept "*not* $\tilde{A}$"
- *Union* $\tilde{A} \cup \tilde{B}$ that represents the fuzzy concept "$\tilde{A}$ *or* $\tilde{B}$"
- *Intersection* $\tilde{A} \cap \tilde{B}$ that represents the fuzzy concept "$\tilde{A}$ *and* $\tilde{B}$"

Linguistic modifiers can be expressed by specific operations on the fuzzy set $\tilde{A}$ describing the basic concept, e.g.,

- *Exponent* $\tilde{A}^{\alpha}$, defined as

$$\mu_{\tilde{A}^{\alpha}}(x) = [\mu_{\tilde{A}}(x)]^{\alpha}$$

- *Concentration,* defined as

$$\text{con}\,(\tilde{A}) = \tilde{A}^2$$

The operation of concentration may be interpreted as "*very* $\tilde{A}$" and its effect is a large reduction of the degrees of membership of those values of x that already have a small degree of membership $\mu_{\tilde{A}}(x)$ to the basic concepts $\tilde{A}$, with an additional small reduction for those x with high membership $\mu_{\tilde{A}}(x)$.

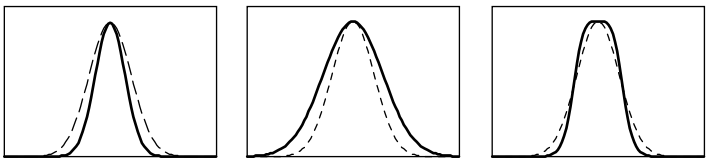

FIGURE 24.14 Examples of linguistic modifiers.

- *Dilatation,* defined by

$$\mathrm{dil}(\tilde{A}) = \tilde{A}^{0.5}$$

The operation of dilatation can be described as "*more or less* $\tilde{A}$" and its effect is opposite to that of concentration.

- *Contrast intensification* int($\tilde{A}$), defined as

$$\mu_{\mathrm{int}(\tilde{A})}(x) = \begin{cases} 2[\mu_{\tilde{A}}(x)]^2 & \text{for } \mu_{\tilde{A}}(x) \le 0.5 \\ 1 - 2[1 - \mu_{\tilde{A}}(x)]^2 & \text{for } \mu_{\tilde{A}}(x) > 0.5 \end{cases}$$

This operation has the consequence of increasing the values $\mu_{\tilde{A}}(x)$ that are above the *crossover point* 0.5 and the reduction of values below this point.

Examples of linguistic modifiers are illustrated in Figure 24.14.

Among various forms of fuzzy propositions, fuzzy implications are of special importance. *Fuzzy implication* is a statement of the form:

$$\tilde{A}(x) \Rightarrow \tilde{B}(y)$$

or, equivalently,

$$\text{if } \tilde{A}(x) \text{ than } \tilde{B}(y)$$

where $x \in X$, $y \in Y$ are linguistic variables and $\tilde{A}(\cdot)$ $\tilde{B}(\cdot)$ are fuzzy predicates in universes of discourse *X, Y,* respectively. Essentially, such a statement describes the fuzzy relation:

$$\tilde{R}_{\Rightarrow}(X,Y) \subseteq X \times Y$$

between the two fuzzy sets, i.e., between the equivalent fuzzy propositions $\tilde{A}(x)$ and $\tilde{B}(y)$.

Fuzzy implication is important because of its role in automatic inferencing. The two basic *fuzzy inference rules* that are based on fuzzy implication are

- *Generalized modus ponens*: $(\tilde{A}'(x) \wedge (\tilde{A}(x) \Rightarrow \tilde{B}(y))) \Rightarrow \tilde{B}'(y)$
- *Generalized modus tollens*: $(\tilde{B}'(y) \wedge (\tilde{A}(x) \Rightarrow \tilde{B}(y))) \Rightarrow \tilde{A}'(x)$

The generalized modus ponens is closely related to the mechanism of forward inferencing (data-driven inference) and it reduces to the classical modus ponens when $\tilde{A}' = \tilde{A}$ and $\tilde{B}' = \tilde{B}$. Analogously, the generalized modus tollens is closely related to the mechanism of backward inferencing (goal-driven inference) and it reduces to the classical modus tollens when $\tilde{A}' = \overline{\tilde{A}}$ and $\tilde{B}' = \overline{\tilde{B}}$.

A basic technique that lies at the heart of most implementations of automatic fuzzy inference is the *compositional rule of inference* proposed by Zadeh.[62] According to this rule, binary fuzzy relation $\tilde{R}$ from X to Y and fuzzy set $\tilde{x} \subseteq X$ induce the fuzzy set $\tilde{y} \subseteq Y$ determined by the sup-star composition

$$\tilde{y} = \tilde{x} \circ R$$

in which $\tilde{x}$ plays the role of unary fuzzy relation. When setting $\tilde{R} = \left(\tilde{A} \Rightarrow \tilde{B}\right) \tilde{x} = \tilde{A}'$, and $\tilde{y} = \tilde{B}'$ in the compositional rule, the rule becomes an implementation of generalized modus ponens:

$$\tilde{B}' = \tilde{A}' \circ (\tilde{A} \Rightarrow \tilde{B}).$$

If $\tilde{A}, \tilde{A}', \tilde{B}, \tilde{B}'$ are nonfuzzy and $\tilde{A}' = \tilde{A}$, the compositional rule of inference becomes

$$\tilde{B}' = \tilde{A} \circ (\tilde{A} \Rightarrow \tilde{B}) = \tilde{B}.$$

Thus, the compositional rule can be regarded as an approximate extension, i.e., a fuzzy generalization of modus ponens: The more different $\tilde{A}'$ is from $\tilde{A}$, the less sharply defined is $\tilde{B}'$.

Because of the significance of fuzzy implication, a number of distinct fuzzy implication functions have been proposed for its implementation. The proposed functions can be divided into five families:[14]

1. Material implication: $\tilde{A} \Rightarrow \tilde{B} \stackrel{\Delta}{=} \overline{\tilde{A}} \cup \tilde{B}$
2. Implication in propositional calculus: $\tilde{A} \Rightarrow \tilde{B} \stackrel{\Delta}{=} \overline{\tilde{A}} \cup (\tilde{A} \cap \tilde{B})$
3. Extended implication in propositional calculus: $\tilde{A} \Rightarrow \tilde{B} \stackrel{\Delta}{=} (\overline{\tilde{A}} \cap \overline{\tilde{B}}) \cup \tilde{B}$
4. Generalization of modus ponens: $\tilde{A} \Rightarrow \tilde{B} \stackrel{\Delta}{=} \sup_{\tilde{C}} \{\tilde{C} : \tilde{C} \cap \tilde{A} \subseteq \tilde{B}\}$
5. Generalization of modus tollens: $\tilde{A} \Rightarrow \tilde{B} \stackrel{\Delta}{=} \inf_{\tilde{C}} \{\tilde{C} : \tilde{C} \cup \tilde{B} \subseteq \tilde{A}\}$

Several authors have analyzed axiomatic requirements and criteria for selection of appropriate functions for implementation of fuzzy implication.[4,15,28] One of the widely accepted definitions is the *standard fuzzy implication*, an implementation of generalized modus ponens in which the standard union and intersection operators are used:

$$\mu_{\tilde{A} \Rightarrow \tilde{B}}(x, y) = \begin{cases} 1 & \text{for } \mu_{\tilde{A}}(x) \le \mu_{\tilde{B}}(y) \\ \mu_{\tilde{B}}(y) & \text{for } \mu_{\tilde{A}}(x) > \mu_{\tilde{B}}(y) \end{cases}$$

24.4.3 Fuzzy Controller

Fuzzy control approaches the control problem in a radically different way compared to the traditional model-based techniques. Instead of precise mathematical models, fuzzy control uses an imprecise and incomplete description of the process and/or the way the system is controlled by human operators, where the theory of fuzzy sets is used as a principle tool.

A fuzzy controller consists of four basic components (see Figure 24.15): condition (fuzzification) interface, knowledge base, inference mechanism, and action interface.

The block denoted as the condition interface performs measurement of input (state) variables:

$$x = [x_1, x_2, \ldots, x_n]^T$$

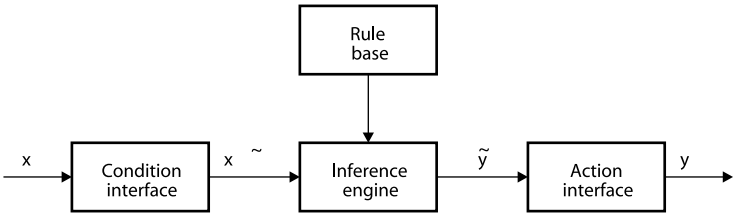

FIGURE 24.15 Components of fuzzy controller.

of the controlled process and translates them into fuzzy linguistic terms X_1, X_2, …, X_n that are represented by fuzzy sets $\tilde{x}_1, \tilde{x}_2,\ldots,\tilde{x}_n$ in appropriate universes of discourse U_1,U_2, …, U_n, respectively. The obtained fuzzy values constitute the *fuzzy state* of the process:

$$\tilde{\mathbf{x}} = [\tilde{x}_1, \tilde{x}_2,\ldots,\tilde{x}_n]^T$$

in the state space $U = U_1 \times U_2 \times \ldots \times U_n$. The fuzzy state variables are further used in evaluation of fuzzy control rules.

The knowledge base consists of control rules and a fuzzy set definition base. The definition base provides the definitions necessary to characterize fuzzy control rules and manipulation of fuzzy data. The rule base consists of heuristic fuzzy control rules that describe control goals and policy. A *fuzzy control rule* is a fuzzy conditional statement (fuzzy implication) in which the antecedent is a condition and the consequent is a control action. Thus, the rule base can be represented as:

R_1: if X_1 is A_{11} and … and X_n is A_{1n}, then $Y_1 = B_{11}$ and … and $Y_m = B_{1m}$

R_2: if X_1 is A_{21} and … and X_n is A_{2n}, then $Y_1 = B_{21}$ and … $Y_m = B_{2m}$

⋮

R_r: if X_1 is A_{r1} and … and X_n is A_{rn}, then $Y_1 = B_{r1}$ and … and $Y_m = B_{rm}$

where:

$X_1, X_2, \ldots, X_n$ = labels of fuzzy state variables $\tilde{x}_1,\tilde{x}_2,\ldots,\tilde{x}_n$ in universes $U_1, U_2, \ldots,U_n$

$Y_1, Y_2, \ldots, Y_m$ = labels of fuzzy actions $\tilde{y}_1, \tilde{y}_2, \ldots \tilde{y}_n$ in universes $V_1, V_2, \ldots, V_m$

A_{ki} and B_{kj} = labels of fixed linguistic values represented by fuzzy sets $\tilde{a}_{ki} \subseteq U_i, \tilde{b}_{kj} \subseteq V_j$

Rules R_k, $k = 1, 2, \ldots, r$ are also mutually interconnected via implicit connectives. Each control rule is implemented by the fuzzy relation $\tilde{\mathbf{R}}_k$ in $\mathbf{U}\times\mathbf{V}$, where $\mathbf{U} = U_1 \times U_2 \times \ldots \times U_m$ and $\mathbf{V} = V_1 \times V_2 \times \ldots \times \boldsymbol{V_m}$. The rule base is an aggregate of individual rules. By integration of particular relations $\tilde{\mathbf{R}}_k, k = 1, 2, \ldots, r$, the aggregate relation of the whole rule base is obtained as:

$$\tilde{\mathbf{R}} \subseteq \mathbf{U}\times\mathbf{V}$$

The block designated as the inference mechanism is responsible for evaluation of control rules. Evaluation is commonly carried out using the sup-star compositional rule:[62]

$$\tilde{\mathbf{y}} = \tilde{\mathbf{x}} \circ \tilde{\mathbf{R}}$$

The result is the *fuzzy control action* $\tilde{\boldsymbol{y}} = [\tilde{y}_1\ \tilde{y}_2 \ldots \tilde{y}_m]^T$ in universe $\mathbf{V}$ of possible control actions. Within the action interface, the fuzzy action $\tilde{\mathbf{y}}$ is converted into defuzzified action $\mathbf{y} = (y_1, y_2, \ldots, y_m)$.

24.4.3.1 Condition Interface

The task of the condition interface is (1) to perform *scale mapping,* which transfers the range of values of input variables into corresponding universes of discourse, and (2) to perform *fuzzification,* which converts crisp inputs into fuzzy sets.

The most frequent fuzzification strategy consists of transforming the measured value x into a fuzzy singleton $\tilde{x}$. Thus, input x is interpreted as a fuzzy set $\tilde{x}$ with the membership function equals zero in all points $u \in U$ except for the point u_0, where $\mu_{\tilde{x}}(u_0) = 1$.

24.4.3.2 Fuzzy Set Definition Base

The fuzzy set definition base contains definitions of fuzzy sets $\tilde{a}_{ki}$ and $\tilde{b}_{kj}$ ($i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$. $k = 1, 2, \ldots, r$) that correspond to linguistic labels A_{ki} and B_{kj} appearing in the control rules. These fuzzy sets are frequently designated as *primary fuzzy sets.*

The universes of discourse for input and output control signals can be discrete or continuous. To attain a more efficient manipulation with fuzzy sets, two basic transformations are commonly applied to the input/output spaces:

- *Normalization*, by which the universe of discourse U is transformed into the normalized closed interval $U_N = [-1, +1]$. The transformation function $f_N(\cdot)$ may be linear or nonlinear and its synthesis assumes *a priori* knowledge on the possible range $U = [u_{\min}, u_{\max}]$ of the signal. For the case of linear mapping,

$$f_N(u) = [(u - u_{\max}) + (u - u_{\min})]/(u_{\max} - u_{\min})$$

  By choosing appropriate nonlinear transformation, a uniform distribution of symmetric and mutually equal primary sets may be achieved.
- *Discretization* (*quantization*), by which the continuous universe U or U_N is partitioned into a finite number of segments:

$$\overline{u}_1 = [\hat{u}_0, \hat{u}_1], \overline{u}_2 = (\hat{u}_1, \hat{u}_2], \ldots, \overline{u}_q = (\hat{u}_{q-1}, \hat{u}_q],$$

- specified by quantization levels $u_{\min} = \hat{u}_0 < \hat{u}_1 < \cdots < \hat{u}_q = \hat{u}_{\max}$. Each segment $\overline{u}_i$, $i = 1, 2, \ldots, q$ is treated as a generic element representing all elements $u \in \overline{u}_i$. In this manner, fuzzy sets can now be defined by assigning degrees of membership to each generic element of the universe $\overline{U} = \{\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_q\}$.

  Quantization may also be linear or nonlinear. The number of quantization levels should be sufficiently large to ensure adequate approximation and yet be small enough to save memory space. In the majority of applications, the number of quantization levels is 16 to 32.

Primary fuzzy sets are usually represented by linquistic labels such as: NB, negative big; NM, negative medium; NS, negative small; ZE, zero; PS, positive small; PM, positive medium; and PB, positive big. The set of different labels:

$$A_i = \bigcup_{k=1}^{r} \{A_{ki}\}$$

is called *fuzzy input space* of the i-th input variable, $i = 1, 2, \ldots, n$. Analogously, the set of different labels:

$$B_j = \bigcup_{k=1}^{r} \{B_{kj}\}$$

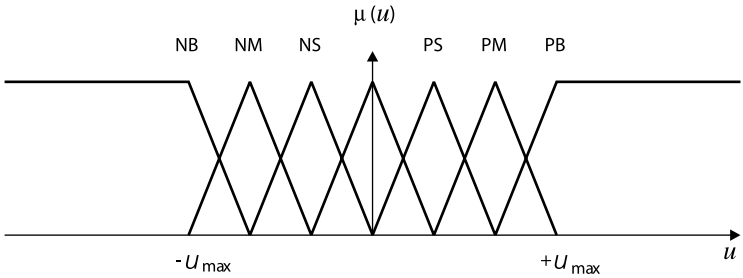

FIGURE 24.16 Primary fuzzy sets.

is called *fuzzy control space* of *j*-th control variable, $j = 1, 2, \ldots, m$. The number of different labels in fuzzy input space determines the number of possible control rules. Finding the optimal fuzzy partition of the input space is a difficult task and is usually performed in a heuristic way.

Depending on whether the underlying universe of discourse is continuous or discrete, primary fuzzy sets are specified using a functional or numerical definition. For the case of continuous universe, commonly applied functional forms of membership functions are

- *Triangular functions*: $\mu_f(x) = \max\left(1 - \left|1 + \frac{x - x_f}{\sigma_f}\right|, 0\right)$
- *Bell-shaped functions*: $\mu_f(x) = e^{-\frac{1}{2}\left(\frac{x - x_f}{\sigma_f}\right)^2}$

An example of triangular primary fuzzy sets is given in Figure 24.16. If the universe of discourse is discrete, a fuzzy set is represented as a vector whose elements are values of the membership degree.

24.4.3.3 Control Rules

A rule R_k in the rule base typically takes the form of a *state evaluation fuzzy control rule*:

R_k: if (X_1 is A_{k1} and … and X_n is A_{kn}), then $Y = B_k$

Linguistic variables appearing on the left side of the implication are typically the process state error (i.e., deviation from desired state), and error change (i.e., time derivative of the error). The variable on the right side is usually the control output or a change of the control output.

A more general form is the *functional control rule,* where premises and consequences are specified as (logical) functions:

$$R_k\text{: if } f_k\ (X_1 \text{ is } A_{k1}, \ldots, X_n \text{ is } A_{kn}), \text{ then } Y = g_k\ (X_1, X_2, \ldots X_n)$$

A popular form of functional rules is the *Sugeno-type rules*,[44] where the antecendent propositions are connected by fuzzy conjunction, and the consequent is a linear function of input variables:

R_k: if X_1 is A_{k1} and … and X_n is A_{kn}, then $y_k = g_{k0} + g_{k1} \times_1 + \cdots + g_{kn}\, x_n$

The principal questions about implementation of fuzzy controllers are connected to the derivation and validation of control rules. Keeping in mind that fuzzy control is primarily efficient in cases when only qualitative and incomplete information is available, the rules are often derived in a heuristic way. Use of expert knowledge and imitation of procedures employed by trained operators are commonly used. Adjustments of control parameters to improve system performances are often made using *ad hoc* procedures that usually reduce to trial-and-error.

Intensive investigations were conducted on development of systematic methods of deriving fuzzy control rules. Most of them use notion of the *fuzzy process model,* i.e., the linguistic description of dynamic characteristics of controlled process.[45,57] The fuzzy model approaches identification of

linguistic structure and model parameters in a systematic manner. Based on the known fuzzy model, control rules can be generated for attaining optimal system behavior.

An alternative solution lies in adding fuzzy controller learning capabilities, i.e., facilities to adapt general control rules to an actual situation. In principle, a fuzzy controller with learning capabilities has a hierarchical structure consisting of two rule bases: a general rule base and base of meta-rules. The meta-rules exhibit human-like learning ability to create and modify the general rules based on the observed and desired performance characteristics of the system. The first such system with learning capabilities, a *self-organizing controller,* was described by Procyk and Mamdani.[37]

24.4.3.4 Inference Mechanism

Consider the rule base:

$R_{1:}$ if X_1 is A_{11} and … and X_n is A_{1n}, then $Y = B_1$

$R_{2:}$ if X_1 is A_{21} and … and X_n is A_{2n}, then $Y = B_2$

⋮

$R_{r:}$ if X_1 is A_{r1} and … and X_n is A_{rn}, then $Y = B_r$

The antecedent of each rule R_k, $k = 1, 2, \ldots, r$ is interpreted as a fuzzy set:

$$\tilde{\boldsymbol{a}}_k = \tilde{a}_{k1} \times \tilde{a}_{k2} \times \cdots \times \tilde{a}_{kn}$$

in the product space $\mathbf{U} = U_1 \times U_2 \times \cdots \times U_n$, with the membership function $\boldsymbol{\mu}_{\tilde{\mathbf{a}}_k(\cdot)}$ given for all $\mathbf{u} \in \mathbf{U}$ by:

$$\mu_{\tilde{\mathbf{a}}_k}(u) = \mu_{\tilde{\mathbf{a}}_k}(u_1, u_2, \ldots, u_n) = f_i(\mu_{\tilde{\mathbf{a}}_{k1}}(u_1), u_{\tilde{\mathbf{a}}_{k2}}(u_2), \ldots, u_{\tilde{\mathbf{a}}_{kn}}(u_n))$$

where $f_i(\cdot)$ denotes any t-norm (intersection) function, such as min or algebraic product. Thus, the rule base can be represented in the form:

R_1: if $\tilde{\mathbf{a}}_1$, then $\tilde{b}_1$

R_2: if $\tilde{\mathbf{a}}_2$, then $\tilde{b}_2$

⋮

R_r: if $\tilde{\mathbf{a}}_r$, then $\tilde{b}_r$

where the antecendents are fuzzy sets in the universe **U**, and the consequents are fuzzy sets in the universe *V.*

If the rule base is *complete* (i.e., it contains all possible fuzzy conditions and, additionally, for every input **u** there exists a *dominant rule* R_k with applicability degree $\mu_{\tilde{\mathbf{a}}_k}(\mathbf{u})$ higher than some level, say, 0.5), then such a base can be interpreted as a sequence of *fuzzy conditional statements:*

if $\tilde{\mathbf{a}}_1$, then $\tilde{b}_1$

else if $\tilde{\mathbf{a}}_2$, then $\tilde{b}_2$

⋮

else if $\tilde{\mathbf{a}}_r$, then $\tilde{b}_r$

It is natural to interpret the fuzzy conditional statements as Cartesian products, and connectives between the conditional statements as union. Thus, the relation represented by the rule base is naturally implemented as:

$$\tilde{\mathbf{R}} = \bigcup_{k=1}^{r} \tilde{\mathbf{R}}_k = \bigcup_{k=1}^{r} \tilde{\mathbf{a}}_k \times \tilde{b}_k$$

Taking into account computational aspects, Cartesian products $\tilde{\mathbf{R}}_k, k = 1, 2, \ldots, r$ are frequently implemented by using min or algebraic product functions, yielding two commonly used *operation rules:*

- *Mamdani's mini-operation rule*: $\mu_{\tilde{\mathbf{R}}_k}(\mathbf{u}, v) = \min(\mu_{\tilde{\mathbf{a}}_k}(\mathbf{u}), \mu_{\tilde{b}_k}(v))$;
- *Larsen's product operation rule*: $\mu_{\tilde{\mathbf{R}}_k}(\mathbf{u}, v) = \mu_{\tilde{\mathbf{a}}_k}(\mathbf{u}).\mu_{\tilde{b}_k}(v)$.

Detailed analysis of the influence of different fuzzy implication functions and union and intersection operators on control quality can be found in, e.g., Lee,[28] Mizumoto,[32] and Stachowicz and Koshanska.[42]

The inference mechanism is based on the sup-star compositional rule of inference:

$$\tilde{y} = \tilde{\mathbf{x}} \circ \tilde{\mathbf{R}}$$

The rule is usually implemented using sup-min or sup-product compositional operator. If this is the case, and if the union is implemented using the max function, fuzzy control action $\tilde{y}$ can be expressed as:

$$\tilde{y} = \bigcup_{k=1}^{r} \tilde{\mathbf{x}} \circ \tilde{\mathbf{R}}_k = \bigcup_{k=1}^{r} \tilde{y}_k$$

or, in terms of the degree of membership function, as:

$$\mu_{\tilde{y}}(v) = \max_{k \in 1,\ldots,r} \tilde{y}_k$$

where $\tilde{y}_k = \tilde{\mathbf{x}} \circ \tilde{\mathbf{R}}_k$ is a local fuzzy control action inferred from the *k*th rule. In terms of the degree of membership function, the local fuzzy control action is determined by:

$$\mu_{\tilde{y}_k}(v) = \begin{cases} \sup_{\mathbf{u}} \min[\mu_{\tilde{\mathbf{x}}}(\mathbf{u}), \mu_{\tilde{\mathbf{R}}_k}(\mathbf{u}, \mathbf{v})] & \text{in case of sup-min composition} \\ \sup_{\mathbf{u}}[\mu_{\tilde{\mathbf{x}}}(\mathbf{u}) \cdot \mu_{\tilde{\mathbf{R}}_k}(\mathbf{u}, v)] & \text{in case of sup-product composition} \end{cases}$$

If input $\tilde{x}$ is a fuzzy singleton with the membership function equaling zero at all points except at the point $\mathbf{u}_0$ at which $\mu_{\tilde{\mathbf{x}}}(\mathbf{u}_0) = 1$, then both versions of the compositional rule of inference reduce to:

$$\mu_{\tilde{y}_k}(v) = \mu_{\tilde{\mathbf{R}}_k}(\mathbf{u}_0, v)$$

In this way, local fuzzy action is determined by the membership function:

$$\mu_{\tilde{y}_k}(v) = \begin{cases} \min[\mu_{\tilde{\mathbf{a}}_k}(\mathbf{u}_0), \mu_{\tilde{b}_k}(v)] & \text{in case of Mamdani's rule} \\ \mu_{\tilde{\mathbf{a}}_k}(\mathbf{u}_0) \cdot \mu_{\tilde{b}_k}(v) & \text{in case of Larsen's rule} \end{cases}$$

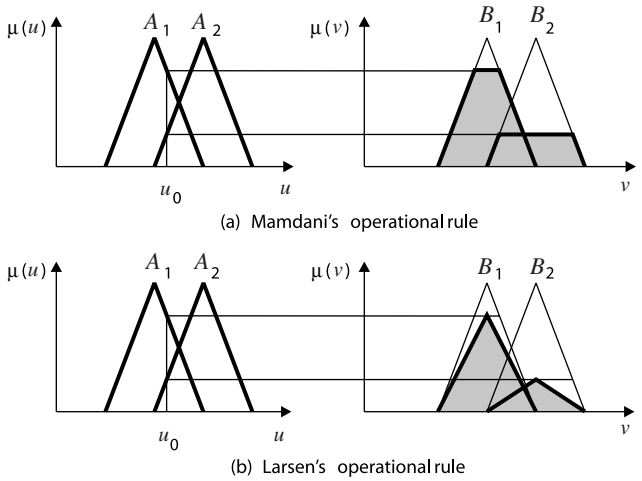

FIGURE 24.17 Operational rules.

The quantity $\mu_{\tilde{\mathbf{a}}_k}(\mathbf{u}_0)$ is referred to as the *firing strength* of the kth rule and represents a measure of the contribution of the kth rule to integral fuzzy control action. It is computed by applying the corresponding operation rule:

$$\mu_{\tilde{\mathbf{a}}_k}(\mathbf{u}_0) = \begin{cases} \min[\mu_{\tilde{\mathbf{a}}_{k1}}(u_{10}), \mu_{\tilde{\mathbf{a}}_{k2}}(u_{20}), \ldots, \mu_{\tilde{a}_{kn}}(u_{n0})] & \text{for Mamdani's rule} \\ \mu_{\tilde{\mathbf{a}}_{k1}}(u_{10}) . \mu_{\tilde{\mathbf{a}}_{k2}}(u_{20}), \ldots, \mu_{\tilde{\mathbf{a}}_{kn}}(u_{n0}) & \text{for Larsen's rule} \end{cases}$$

The mechanism of inference in these two methods is illustrated in Figure 24.17. The first method has the advantage of enhancing the contribution of the dominant rule, so that it is widely used in fuzzy applications. On the other hand, the second method has an advantage of preserving the contribution of all rules to the control action.

The obtained fuzzy control action $\tilde{y}$ (or a set of local control actions y_k, $k = 1, 2, \ldots, r$) is transferred to the action, i.e., defuzzification interface, where the actual crisp control signal is generated.

24.4.3.5 Action Interface

Degree of membership function of fuzzy control action can be interpreted as a distribution of possibility $\mu_{\tilde{y}}(v)$ to achieve a control goal by the signal v. Action interface's purpose is to generate a control signal that will best represent the possibility distribution of inferred fuzzy action. Frequently used strategies employed by the action interface are

- *The mean-of-maximum method.* With this strategy, control action is derived as a mean value of all points v at which the membership function of the fuzzy control reaches the global maximum $M = \max_v[\mu_{\tilde{y}}(v)]$. In case of a discrete universe $V = \{\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_q\}$ the control y is computed as:

$$y = \frac{1}{|L|} \sum_{l \in L} \bar{v}_l$$

where L is a set of all indices for which the grade membership function reaches maximum, i.e., $L = \{l : \mu_{\tilde{y}}(\bar{y}_l) = M\}$.

- *The center-of-mass method.* With this method, control is generated as the center of mass of the possibility distribution of fuzzy control. In case of the discrete universe,

$$y = \frac{\sum_{l=1}^{q} \mu_{\tilde{y}}(\bar{v}_l) \cdot v_l}{\sum_{l=1}^{q} \mu_{\tilde{y}}(\bar{v}_l)}.$$

Analytical indices that could serve as guides for selecting the preferred method are not known. However, experiments conducted by several authors, e.g., Mandič et al.[31] have shown that the mean-of-maximum method yields a better transient performance while the center-of-mass method yields a better steady-state performance.

A slightly different scheme is employed with Sugeno-type rules, where fuzzy logic is employed only to describe conditions for the application of the rule whereas the control actions are fuzzy singletons, i.e., the control signals in classical sense:

$$R_k\text{: if } X_1 \text{ is } A_{k1} \text{ and } \ldots \text{ and } X_n \text{ is } A_{kn} \text{ then } y_k = g_{k0} + \sum_{i=1}^{n} g_{ki} x_i$$

Here, it is natural to employ the firing strengths $\mu_{\tilde{\mathbf{a}}_k}(\mathbf{u}_0)$ of the rules to directly determine the crisp control signal y. A standard technique is to generate the aggregate signal as a weighted average of local controls, where the firing strengths are used as weighting factors:

$$y = \frac{\sum_{k=1}^{r} \mu_{\tilde{\mathbf{a}}_k}(\mathbf{u}_0) \cdot \mathbf{y}_k}{\sum_{k=1}^{r} \mu_{\tilde{\mathbf{a}}_k}(\mathbf{u}_0)}$$

24.4.4 Direct Applications

Frequently used control rules in fuzzy controllers are of the type:

$$R_k\text{: if } (E \text{ is } A_k \text{ and } \Delta E \text{ is } B_k) \text{ then } U = C_k$$

where E represents value of error e, ΔE represents error change Δe between successive operation cycles of the controller, and U represents fuzzy control action that is transferred to the action interface which in turn generates control signal u. For the operation of such a controller of special importance are *normalizing gains* that are effectively applied within normalization that takes place during conversion of actual signal values and their fuzzy representations. Normalized values of the signals can be represented as:

$$e' = e/E_{\max} = G_E \cdot e$$

$$\Delta e' = \Delta e/\Delta E_{\max} = G_{\Delta E} \cdot \Delta e$$

$$u' = u/U_{\max} = u/G_U$$

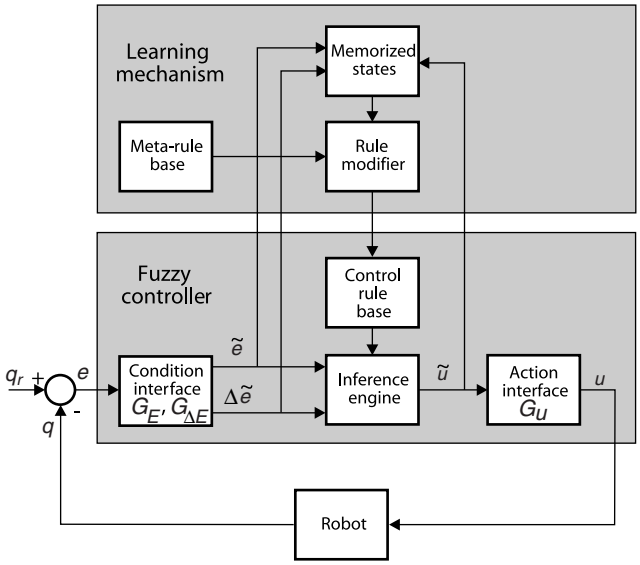

FIGURE 24.18 Self-organizing controller.

where E_{max}, ΔE_{max}, U_{max} are maximum values of the signals, e', $\Delta e'$, u' are corresponding normalized values, and G_E, $G_{\Delta E}$, G_U denote normalizing gains. If the operation of the fuzzy controller is regarded as nonlinear mapping from e and Δe to u, e.g.,

$$u = G_U \cdot f\,(G_E \cdot e,\ G_{\Delta E} \cdot \Delta e)$$

then it is obvious that fuzzy controller is actually a nonlinear PD controller. Conversely, the conventional PD controller can be considered as a special case of the fuzzy PD controller for which the consequent of the rule is

$$u = K_p e + K_d \Delta e$$

Analogously, a fuzzy controller with control rules of the type:

$$R_k\text{: if } (E \text{ is } A_k \text{ and } \Delta E \text{ is } B_k) \text{ then } \Delta U = C_k$$

where ΔU represents change in control output, is actually a nonlinear PI controller. This analogy with conventional controllers is often employed in synthesis and adjustments of fuzzy control.

The first application of fuzzy control to robotic manipulators was described by Mandič, Scharf, and Mamdani.[31] In their 1985 paper, these authors described a series of experiments with a two-DOF robot controlled by two independent self-organizing controllers. Both controllers are of the same structure (see Figure 24.18) that consists of two levels. The lower level is a usual fuzzy controller with control rules, whereas the upper level is a system that realizes the mechanism of automatic learning, i.e., the generation and modification of the rules at the lower level. Control rules employed at the lower level were of the type:

$$R_k\text{: if } (E \text{ is } A_k \text{ and } \Delta E \text{ is } B_k) \text{ then } U = C_k$$

where E represents joint position error $e = q - q_0$, ΔE represents error change, and U represents control action. The upper level is responsible for evaluating controller performance and modifying

the control rule base. Evaluation of performance is achieved using a production system whose structure is identical to the basic fuzzy controller. Performance is evaluated using a local criterion that roughly expresses deviation P between the actual and desired system response. Evaluation criteria are expressed using a set of meta-rules of the form:

$$\Pi_k\text{: if } (E \text{ is } V_{Ek} \text{ and } \Delta E \text{ is } V_{\Delta Ek}), \text{ then } P = V_{Pk}$$

In these rules, parameters of the primary fuzzy set V_{Pk} = zero define a tolerance range for system response, whereas the values different from zero imply the desired degree of correction. Remarkably, such defined rules depend to a very small extent on the controlled process and really express the tolerable errors and degree of acceptability of the errors. If the base of meta-rules is represented by fuzzy relation Π, then the output of the evaluator is a nonlinear function:

$$p = \pi(e, \Delta e)$$

If the precise model of the controlled system were available, then a needed correction in control Δu in principle could be calculated from the known index p. Because the use of the model is always accompanied by inaccuracies, the self-organizing controller instead performs a modification of the control rules that is based on simplified assumptions that (1) the current system performance index $p(t)$ is a consequence of control $u(t - nT)$ generated n operation cycles prior the current time instant t, and (2) the necessary correction in control $\Delta u(t - nT)$ is proportional to $p(t)$:

$$r(t) = \Delta u(t - nT) = \lambda \cdot p(t)$$

In other words, it is accepted that corrections in control are not 100% accurate and that the learning process is slower.

Modifications in the rule base are achieved using fuzzy set operations. Namely, the rule base at the time instant t can be represented as union:

$$\tilde{R}(t) = \bigcup_{k=1}^{r(t)} \tilde{e}_k \times \Delta\tilde{e}_k \times \tilde{u}_k$$

If the function transforming value x into fuzzy singleton $\tilde{x} = F\{x\}$ is denoted by $f\{\cdot\}$, then the control, generated at the time instant $t - nT$ may be regarded as a value that corresponds to the fuzzy implication:

$$\tilde{R}'(t) = f_E\{e(t - nT)\} \times f_{\Delta E}\{\Delta e(t - nT)\} \times f_U\{u(t - nT)\}$$

whereas the desired control at the current time instant is regarded as a value corresponding to the implication:

$$\tilde{R}''(t) = f_E\{e(t - nT)\} \times f_{\Delta E}\{\Delta e(t - nT)\} \times f_U\{u(t - nT) + \lambda \cdot p(t)\}$$

Now, the problem of modifying control rules can be expressed as a problem of substituting implication $\tilde{R}'(t)$ with the implication $\tilde{R}''(t)$. One of the ways to achieve it is to describe the substitution with the expression:

$$\tilde{R}(t + T) = [\tilde{R}(t) \text{ and not } \tilde{R}'(t)] \text{ or } \tilde{R}''(t)$$

or, equivalently:

$$\tilde{R}(t+T) = \left[\tilde{R}(t) \cap \overline{\tilde{R}'(t)}\right] \cup \tilde{R}''(t)$$

Direct application of this formula would lead to exponential growth of the number of rules. Therefore, an approximate method is used where only a single rule is modified at a time. The modified rule is the dominant rule, i.e., the rule $\tilde{R}_k = \tilde{e}_k \times \Delta\tilde{e}_k \times \tilde{u}_k$ for which the heights of intersections $\tilde{e}_k \cap f_E\{e(t-nT)\}$ and $\Delta\tilde{e}_k \cap f_{\Delta E}\{\Delta e(t-nT)\}$ are at least equal to 0.5. Once the dominant rule is identified, its old fuzzy action $\tilde{u}_k$ is replaced by the action $f_U\{u(t - nT) + \lambda \cdot p(t)\}$.

In experiments by Mandič, Scharf, and Mamdani,[31] it was demonstrated on a real robot that a self-organizing controller was able, after only a few adaptation cycles, to attain steady performance that was completely comparable to that of a conventional PID controller. Tanscheit and Scharf,[46] who described several experiments with self-organizing controllers applied to control of a second-order linear system that represented the transfer function of a DC motor with variable load (the variable load was represented by the variable moment of inertia) arrived at similar results.

These and other works, in which direct control of manipulation robots by fuzzy controllers was tried, emphasized two main problems. The first is manifested by the lack of analytical tools for control synthesis, i.e., the selection of parameters of fuzzy controllers (or initial values of the parameters in cases of self-organizing controllers). Second, ordinary fuzzy controllers have attained performances similar to, or slightly better than simple PID schemes. Therefore, it may be expected that direct application of fuzzy controllers will not yield satisfactory performance in more complex robotic tasks, such as tracking fast trajectories. The appearance of these problems can be partially explained by the fact that the early works primarily concentrated on demonstrating the ability of fuzzy logic-based methods to effectively master the nonlinear control problems without need for exact mathematical modeling of the controlled system. For this reason, the role of *a priori* available mathematical knowledge (in situations where the system dynamics is deterministic) as well as the established model-based control techniques was somewhat overshadowed.

24.4.5 Hybridization with Model-Based Control

The problem of merging fuzzy logic-based control with analytic methodologies to exploit advantages of both approaches in real-time robot control was addressed by several authors. Lim and Hiyama[29] have proposed a decentralized control strategy that incorporates a PI controller and a simple fuzzy logic controller. In their approach, the PI controller was used to enhance transient response and steady-state accuracy, whereas fuzzy control was to enhance damping of the overall system. A tighter connection between fuzzy and standard control methods was proposed by Tzafestas and Papanikolopoulos,[50] who suggested employing a two-level hierarchy in which a fuzzy logic-based expert system is used for fine tuning low-level PID control. A similar approach was applied to robot control by Popović and Shekhawat.[36] However, the two-level control hierarchy by itself does not actually solve the problem of weak performance in situations characterized by quickly varying robot dynamics. In such cases, knowledge of readily available mathematical models of robot dynamics cannot be ignored. Therefore, fuzzy logic-based control should not be viewed as a pure alternative to model-based robot control. Instead, a combined approach is preferred, and it may yield superior control schemes over both simple model-based or fuzzy logic-based approaches.

The general idea behind the hybrid approach is utilization of a satisfactory approximation of the model of robot dynamics to decrease dynamic coupling between robot joints and then engage the fuzzy logic-based heuristics as a effective tool for creating a nonlinear performance-driven PID control to handle the effects uncovered by the approximate model. A similar concept was formulated by de Silva and MacFarlane,[8] who proposed a three-level hierarchy for robot control. The proposed hierarchy consists of:

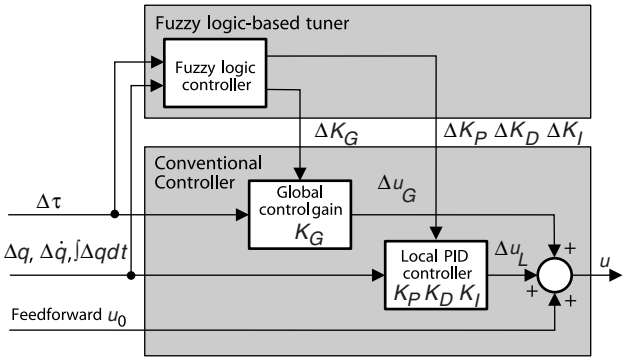

FIGURE 24.19 Hybrid control scheme.

1. Conventional robot controller, i.e., a set of PID controllers closed around a fast decoupling controller. The important role of this controller is to assure the decoupling and linearization needed to efficiently apply expert knowledge for tuning the PID controllers (in an idealized case, every joint subsystem would behave as an independent oscillator with PID control).
2. Intelligent pre-processor, i.e., a set of knowledge-based observers. Each observer is implemented as a fuzzy system and its outputs are the attributes of response at a corresponding joint (e.g., accuracy, oscillations, error convergence, divergence, and steady-state error).
3. Fuzzy tuner, i.e., a fuzzy controller that is used for tuning the gains of the PID controllers at the lowest level.

de Silva and MacFarlane[8] have tested their approach by simulation of a two-link manipulator with an assumption of idealized effectiveness of the low-level global nonlinear feedback. Therefore, robot dynamics was approximated in their work by a set of joint subsystems modelled as second-order systems with unknown acceleration-type disturbances.

The idea of a hybrid approach to robot control was elaborated in detail by Vukobratović and Karan,[52,53] who have employed fuzzy logic to express control policy and have determined analytically conditions on values of fuzzy control parameters that assure stability of a closed-loop robot control system. The authors have analyzed a hybrid design that is an extension to decentralized control strategy. The proposed controller consists of a set of subsystems closed around individual robot joints where each of the subsystems comprises two components: conventional model-based controller and fuzzy logic-based tuner (see Figure 24.19). Inputs to ith joint subsystem, $i = 1, \ldots, n$, where n is the number of actuated joints, are nominal control signal u_{0i}, joint position error Δq_i, joint velocity error $\Delta\dot{q}_i$, and integral error $\int \Delta q_i dt$. In cases where a highly precise tracking of fast trajectory is necessary, an optional global feedback loop (full dynamic compensation) can be added. Global feedback is generated on the basis of computed or measured deviation of dynamic torque $\Delta\tau_i$ acting at the joint and is synthesized to assure practical system stability.[54] A further refinement introduces the upper level that tunes the gains of the PID controllers. The tuner is designed as a fuzzy controller that monitors joint response characteristics and modifies the gains to provide better responses for large deviations of monitored quantities. Although its general structure permits construction of sophisticated control rules for tuning the gains, Vukobratović and Karan have considered a simple decentralized scheme consisting of independent joint servo tuners operating on the basis of observed joint position error Δq, velocity error $\Delta\dot{q}$, and integral error $\int \Delta q dt$. A rather simple heuristics for synthesizing gain-tuning rules was used:

1. If the observed errors are large and do not show a significant tendency to decrease, the proportional gain is enlarged to speed-up error convergence.
2. When the errors are small, the proportional gain is decreased to prevent resonance oscillations and attenuate undesired noise effects.

3. If errors are large, but the error convergence is satisfactory, the proportional gain is gradually decreased to the appropriate value for small-error conditions.
4. The values of derivative and integral gains are changed simultaneously with the changes in proportional gain so that the stability of the whole system is preserved. The actual values are derived from stability analysis of the closed-loop system. Importantly, the readily available stability conditions for a fixed-gain controller were reused to determine the conditions on parameters of nonlinear fuzzy tuners that are sufficient for overall system stability.

In spite of their simplicity, the rules resulted in significant improvements compared to those of the fixed-gain model-based controller. Simulation experiments on a real-scale six-DOF industrial robot have shown that the resulting variable-gain controllers in many respects outperform constant-gain schemes. The most obvious advantage was the improvement in accuracy demonstrated in both positioning and trajectory tracking tasks. An important feature is that the accuracy improvement was not accompanied by degradation in other performance characteristics, such as energy consumption and maximum developed torques. The second considerable aspect is the possibility of reducing the computational complexity of the nominal robot model (by employing approximate robot models) without the significant loss in control quality that was notable with fixed-gain control. Although the issues related to sensitivity to parameter variations were not explicitly investigated, an improved robustness of the variable-gain controller is implied by the results obtained from experiments with approximate robot models.

24.5 Neuro-Fuzzy Approach in Robotics

Although fuzzy logic can directly and easily encode expert knowledge using rules with linguistic labels, it often takes a lot of time to design and tune the membership functions that quantitatively define these linguistic labels. Wrong membership functions can lead to poor controller performance and possible instability. An execellent solution is to apply learning techniques by neural networks that can be used to design membership functions automatically, thus reducing development time and cost while improving performance. These combined neuro-fuzzy networks can learn faster than neural networks. Also, they provide a connectionist architecture that is easy for very large scale integrated (VLSI) implementation of the functions of a traditional fuzzy logic controller with distributed learning abilities.

The most proposed neuro-fuzzy networks are, in fact Takagi-Sugeno controllers,[49] where the consequent parts of linguistic rules are constant values. Figure 24.20 shows the commonly used connectionist fuzzy system. The system has a total of five layers. Nodes at layer one are input nodes (linguistic nodes) which represent input linguistic variables, Nodes at layer two act as membership functions (in Figure 24.20 they are the Gaussain functions) to represent the terms of the respective linguistic variable. Each node at layer three is a rule node that represents one fuzzy rule. Thus, all layer-three nodes form a fuzzy rule base. Layer five is the output layer. Links at layers three and four function as a connectionist inference engine.

There are many different supervising learning methods for neuro-fuzzy networks.[27,40] Many learning methods are application of the backpropagation method to neuro-fuzzy systems. In addition to gradient-descent techniques, reinforcement learning[7] and some hybrid learning techniques[21] are proposed. One of the most important is ANFIS (adaptive-network-based fuzzy inference system).[21] The learning rule is a hybrid method that combines the gradient descent and the least square estimate to identify the parameters of ANFIS. Usually, neuro-fuzzy networks are trained by applying hybrid techniques where the consequent parts of the rules are adapted with a supervised method and the parameters of the antecedent parts are updated with an unsupervised technique (vector quantization). The idea comes from the field of radial basis functions neural networks.

The one of most important applications of fuzzy-neural networks in robotics is in the field of mobile robotics.[22] A mobile robot is a nonlinear plant that is difficult to model. The state variables

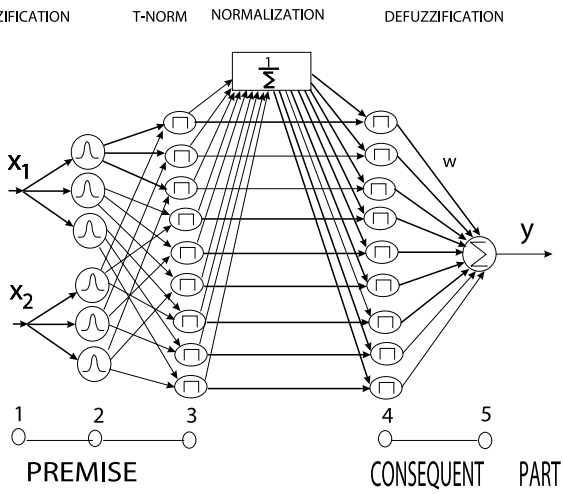

FIGURE 24.20 The structure of fuzzy-neural network.

of a mobile robot are easy to visualize: they have an intuitive relation to the robot's behavior. Therefore, the linguistic if–then rules could be defined in an intuitive way. The problem occurs when a robot has many sensors and actuators. The complexity of the controller increases and the construction of the rule base is more complicated, especially if complex behavior is required. Hence, fuzzy-neural networks are specially applied for the complex task of mobile robot navigation and obstacle avoidance in real time. The network's input data are based on direct or indirect data from many laser, infra-red, and ultrasonic sensors and some other robot velocity sensors that measure the robot's distance from obstacles in the environment, the heading angle between the robot and the specified target, and the velocity of the robot. The network's output values are the control signals for the robot's wheels to determine the appropriate direction of the motion and velocity of the robot. Learning appropriate behavior in the training process defined by the fuzzy-neural controller can be accomplished by the supervisor or by using reinforcement learning for unsupervised learning. In this way, by learning the designer can extract fuzzy rules. Experimental results[55] show that proposed neuro-fuzzy systems can improve navigation performance in complex and unknown environments. This architecture is suitable for robot navigation by multisensor fusion and integration.

Fuzzy-neural networks can be efficiently applied to learning dynamic control and position/force control.[25] It is especially effective in the case when control is applied to an unknown environment. The input data for this type of problem are the appropriate position, velocity, or force errors, while the output of fuzzy-neural network is the control signal.

24.6 Genetic Approach in Robotics

Genetic algorithms (GA) are the global search algorithms for solving optimization problems based on the mechanism of natural selection and natural genetics. It is not a gradient search technique, because the algorithms combine survival of the fittest among string structures (binary or nonbinary type) with a structured yet randomized infomation exchange. Furthermore, GA is not considered a mathematically guided algorithm. It is merely a stochastic, discrete event and a nonlinear process that gives the optima containing the best elements of previous generations. GA is inspired by the biological process in which stronger individuals are likely be the winners in a competing environment. It presumes that the potential solution of a problem is an individual and can be represented by a set of parameters. These parameters are regarded as the genes of a chromosome and can be structured by a string of values in binary or nonbinary form. The fitness value is used to reflect the degree of "goodness" of the chromosome for

solving the problem, and this value is closely related to its objective value. Through genetic evolution, a fitter chromosome tends to yield good-quality offspring, which means a better solution to the optimization problem. In each cycle of genetic operation, an evolving process, a subsequent generation is created from the chromosomes in the current population. This process is achieved through a specific selection routine. The genes of the parents are mixed and recombined to produce offspring. The cycle of evolution is repeated until a desired termination criterion is reached. This criterion can also be set by the number of evolution cycles, the amount of variation of individuals between different generations, or a predefined value of fitness.

To facilitate the GA evolution cycle, two fundamental operators, *crossover* and *mutation,* are required. The crossover process is a reform operation for the survival candidates and is performed by exchanging pieces of string using information from old strings. The pieces are crossed in pairs of strings selected randomly. However, mutation is applied to each offsping individually with random alteration of each bit with a small probability with a typical value of less than 0.1. The choice of crossover and mutation probability can be a complex, nonlinear otimization problem.

The general structure of GA is shown by the following algorithm:

Genetic Algorithm

```
{
*** initial time
t:=0;
*** initialize a random population of individuals
initpopulation P(t);
*** evaluate fitness of all individuals in population
evaluate P(t);
*** test for termination criterion
while not done do
   *** increase the time counter
   t:=t+1;
   *** select a sub-population of offspring
   P':=selectparents P(t);
   *** recombine the genes of selected parents
   recombine P'(t);
   *** mutation of each offspring
   mutate P' (t);
   *** evaluate the new fitness
   evaluate P' (t);
   *** select the survivors from actual fitness
   P:=survive P,P' (t);
od
}
```

GA can be efficiently applied in the various research areas of mobile, industrial, and locomotion robotics. The one of dominant application of GA is the kinematic domain for trajectory optimization and navigation in mobile robotics. Michalewicz[65] adopted the order-based coding in his evolutionary navigator for mobile robot. A chromosome in an evolutionary navigator (EN) is an ordered list of path nodes. Each of path nodes, apart from the pointer to the next node, consists of x and y coordinates of an intermediate knot point along the path, and a Boolean variable b indicating whether the given node is feasible or not. EN unifies off-line and on-line planning with a simple map of high-fidelity and efficient planning algorithms. The off-line planner searches for the optimal global path from the start to the desired destination, whereas the on-line planner is responsible for handling possible collisions of previous objects by replacing a part of the original global path with the optimal subtour.

An interesting approach for GA optimization in robotics is tuning control parameters for some specific robot applications. For many robot controllers there are currently no systematic approaches to choose controller parameters to obtain desired performance. Controller parameters are usually determined by trial-and-error through simulations and experimental tests. In such cases, the paradigm of GA appears to offer an effective way for automatically and efficiently searching for a set of controller parameters for better performance. The effectiveness of this approach is demonstrated by applying a simple and efficient decimal GA optimization procedure for tuning and optimizing robust controllers for position/force control and control of flexible link robots.[41] The robust controller is developed based on stability theory and uses special fitness functions. It is a special GA algorithm with decimal real-number type representation (instead of binary type). The specially used fitness functions are integral time-multiplied absolute value of errors (ITAE) and the normally used integral of squared errors (ISE).

For locomotion robots, GA can be efficiently applied to hierarchical trajectory generation of the natural motion of bipeds using energy optimization.[24,35] The hierarchical trajectory generation method consists of two layers, one is the GA that minimizes the total energy of all actuators, and the other is the evolutionary programming layer that optimizes interpolated configuration of biped locomotion robots. The second example is application of GA to PD local gain tuning and determination of nominal trajectory for dynamic biped walking. Designs to achieve different goals, such as being able to walk on an inclined surface, walk at a high speed, or walk with specified step size, have evolved with the use of GA.

GA are particularly efficient as hybrid techniques with other intelligent soft-computing methods. Together with neural networks, GA can be efficiently applied to determine optimal weighting factors, the topology of networks (number of neurons, number of layers, types of activation functions) and parameters of learning rules. Together with fuzzy rules, GA can be efficently applied to optimization of membership functions, the number of rules, and the parameters of consequent part of rules. On the other hand, fuzzy logic and neural networks can be evaluation functions for GA in the case of complex optimization problems.

In fuzzy-genetic algorithms, it is necessary to solve some problems connected with transformation between the domain of fuzzy knowledge and the GA-coded domain together with using initial expert knowledge for better further searching. In one special example (hierachical fuzzy controller for control of flexible link robots),[3] GA performs optimization of two fuzzy systems: the fuzzy extractor of features at the higher control layer and the fuzzy controller at the lower control layer. For this problem there is very interesting hardware solution where the fuzzy controller works on a DSP board in direct connection with a GA that is executed on a Pentium 133 MHz board. Another typical application of the GA–fuzzy approach in robotics is tuning of local fuzzy-PID controller gains.[2]

GA is applied with the connectionist approach to control visually guided swing motions of a two-armed bipedal robot.[26] The goal is that the neural network learns from the GA swing motion based on visual informatiion from the virtual environment. The goal of visual swing motion is increasing swing amplitude by changing the gravity center in direction of the swing radius acquired through the process of environment recognition using cameras. The inputs in the network are optical signals from cameras, while the outputs are knee joint angles. GA optimizes the three sets of weighting factors of the proposed multilayer perceptron (Figure 24.21). The initial population is 200. The GA–connectionist approach includes determining the weighting factor for recurrent networks for generation of stable biped motion.[47]

24.7 Conclusion

The challenge to future intelligent control system researchers and designers in robotics is to take advantage of the desirable properties of different composite soft-computing control paradigms. It is important to combine the experience and dependability of classic and traditional adaptive control

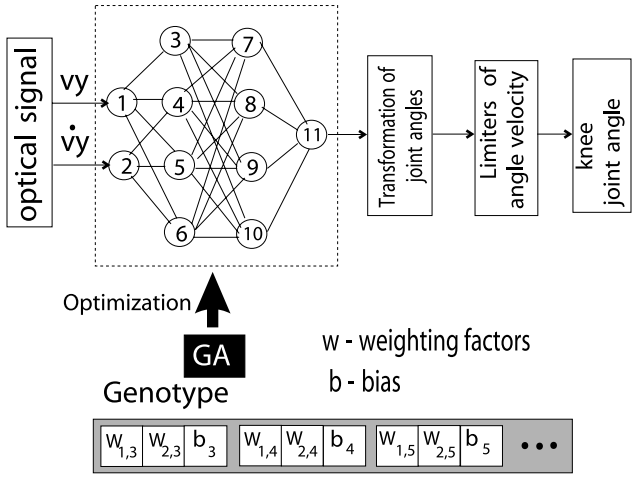

FIGURE 24.21 GA–connectionist approach for robot swing motion optimization.

methods with the potential and promise of soft-computing techniques. One promising idea is the integration of various artificial intelligence paradigms (expert systems, connectionist systems, fuzzy logic, evolutionary algorithms) for the purpose of robot control. The goal of the hybrid approach for robot control is to overcome the weaknesses of each individual intelligent technique by combining it with another complementary intelligent paradigm. This approach is the basis for the development of a generation of intelligent, highly adaptive robotic systems.

References

1. A. Guez and J. Selinsky. Neurocontroller design via supervised and unsupervised leaning. *Journal of Intelligent and Robotic Systems,* 2(2–3):307–335, 1989.
2. A. Homaifar, M. Bikdash, and V. Gopalan. Design using genetic algorithms of hierarchical hybrid fuzzy-PID controllers of two-link robotic arms. *Journal of Robotic Systems,* 14(5):449–463, June 1997.
3. M.-R. Akbarzadeh and M. Jamshidi. Evolutionary fuzzy control of a flexible-link. *Intelligent Automation and Soft Computing,* 3(1):77–88, 1997.
4. J. F. Baldwin and B.W. Pitsworth. Axiomatic approach to implication of approximate reasoning with fuzzy logic. *Fuzzy Sets and Systems,* 3:193–219, 1980.
5. B. Horne, M. Jamshidi, and N. Vadiee. Neural networks in robotics: A survey. *Journal of Intelligent and Robotic Systems,* 3:51–66, 1990.
6. B. Widrow. Generalization and information storage in networks of Adaline neurons. In *Self-Organizing Systems*, pp. 435–461, Spartan Books, New York, 1962.
7. C.-T. Lin and C.S.G. Lee. Reinforcement structure/parameter learning for neural network-based fuzzy logic control system. *IEEE Transactions on Fuzzy Systems,* 2(1):46–63, 1994.
8. C.W. de Silva and A.G.J. MacFarlane. *Knowledge-Based Control with Application to Robots.* Springer, Berlin, 1989.
9. D.E. Rumelhart and J.L. McClelland. *Parallel Distributed Processing* (*PDP*)*: Exploration in the Microstructure of Cognition,* Vol. 1–2. MIT Press, Cambridge, 1986.
10. D.F. Bassi and G.A. Bekey. Decomposition of neural networks models of robot dynamics: A feasibility study. In W. Webster, Ed., *Simulation and AI*, pp. 8–13. The Society for Computer Simulation International, 1989.
11. D. Katić and M. Vukobratović. Connectionist approaches to the control of manipulation robots at the executive hierarchical level: An overview. *Journal of Intelligent and Robotic Systems,* 10:1–36, 1994.

12. D. Katić and M. Vukobratović. Highly efficient robot dynamics learning by decomposed connectionist feedforward control structure. *IEEE Transactions on Systems, Man and Cybernetics*, 25(1):145–158, 1995.
13. D. Katić and M. Vukobratović. Robot compliance algorithm based on neural network classification and learning of robot-environment dynamic models. In *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 2632–2637, Albuquerque, NM, April 1997.
14. D. Dubois and H. Prade. Fuzzy logics and the generalized modus ponens revisited. *Cybern. Syst.,* 15:3–4, 1984.
15. S. Fukami, M. Mizumoto, and K. Tanaka. Some considerations of fuzzy conditional inference. *Fuzzy Sets and Systems,* 4:243–273, 1980.
16. G.A. Bekey and K.Y. Goldberg. *Neural Networks in Robotics.* Kluwer, Norwell, MA, 1993.
17. G.A. Carpenter and S. Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, pp. 77–88, March 1988.
18. H. Asada and S. Liu. Transfer of human skills to neural network robot controllers. In *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 2442–2448, Sacramento, April 1991.
19. S. Haykin. *Neural Networks — A Comprehensive Foundation,* Macmillan, Englewood Cliffs, 1994.
20. J. Albus. A new approach to manipulator control: The cerebellar model articulation controller. *Journal of Dynamic Systems, Measurement and Control,* 97:220–227, September 1975.
21. J.-S.R. Jang. ANFIS, adaptive-network-based fuzzy inference systems. *IEEE Transactions on Systems, Man and Cybernetics,* 23(3):665–685, 1992.
22. J. Godjevac. *Neuro-Fuzzy Controllers*. Presses Polytechniques et Universitaires Romandes, Lausanne, 1997.
23. J. Hopfield. Neural networks and physical systems with emergent collection computational abilities. In *Proceedings of the National Academy of Science,* pp. 2554–2558, April 1988.
24. J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski. Adaptive evolutionary planner/navigator for mobile robots. *IEEE Transactions on Evolutionary Computation,* 1(1):18–28, April 1997.
25. K. Kiguchi and T. Fukuda. Robot manipulator contact force control application of fuzzy-neural network. In *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 875–880, Nagoya, May 1995.
26. K. Nagasaka, A. Konno, M. Inaba, and H. Inoue. Acquisition of visually guided swing motion based on genetic algorithms and neural networks in two-armed bipedal robot. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation,* pp. 2944–2949, Albuquerque, NW, April 1997.
27. L.-X. Wang. *Adaptive Fuzzy Systems and Control: Design and Stability Analysis.* Prentice Hall, 1994.
28. C.C. Lee. Fuzzy logic in control systems: Fuzzy logic controller. *IEEE Transactions on Systems, Man and Cybernetics,* 20(2):404–435, April 1990.
29. C.M. Lim and T. Hiyama. Application of fuzzy logic control to a manipulator. *IEEE Transactions on Robotics and Automation,* 7(5):688–691, October 1991.
30. E.H. Mamdani. Application of fuzzy algorithms for control of simple dynamic plant. *Proc. IEEE Control and Science,* 121(12):1585–1588, December 1974.
31. N.J. Mandič, E.M. Scharf, and E.H. Mamdani. Practical application of a heuristic fuzzy rule-based controller to the dynamic control of a robot arm. *IEEE Proceedings on Control Theory and Applications,* 132(4):190–203, July 1985.
32. M. Mizumoto. Fuzzy controls under various approximate reasoning methods. In *Proceedings of the 2nd IFSA Congress,* pp. 143–146, Tokyo, 1987.
33. M. Kuperstain. Adaptive visual-motor coordination in multijoint robots using parallel architecture. In *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 1595–1601, Raleigh, March 1987.
34. M. Rucci and P. Dario. Autonomous learning of tactile-motor coordination in robotics. In *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 3230–3236, San Diego, May 1994.

35. M. Zhao, N. Ansari, and E. Hou. Mobile manipulator path planning by a genetic algorithm. In *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 681–688, Raleigh, NC, July 1992.
36. D. Popović and R.S. Shekhawat. A fuzzy expert tuner for robot controller. In *Proceedings of the 1991 IFAC/IFIP/IMACS Symposium on Robot Control,* pp. 229–233, Vienna, 1991.
37. T.J. Procyk and E.H. Mamdani. A linguistic self-organizing process controller. *Automatica*, 15(1):15–30, January 1979.
38. J-N. Hwang and S-Y. Kung. Neural network architectures for robotic applications. *IEEE Transactions on Robotics and Automation,* 5(5):641–657, October 1989.
39. S.H. Huang and H.-C. Zhang. Artificial neural networks in manufacturing: Concepts, applications, and perspectives. *IEEE Transactions on Components, Packaging, and Manufacturing Technology — Part A*, 17(2):212–228, June 1994.
40. S. Horikawa, T. Furuhashi, and Y. Uchikawa. On identification of structures in premises of a fuzzy model using a fuzzy neural network. In *Proceedings of the 2nd IEEE International Conference on Fuzzy Systems,* pp. 661–666, 1993.
41. S.S. Ge, T.H. Lee, and G. Zhu. Genetic algorithm tuning of Lyapunov-based controllers: An application to a single-link flexible robot system. *IEEE Transactions on Industrial Electronics,* 43(5):567–574, October 1996.
42. M. S. Stachowicz and M.E. Kochanska. Fuzzy modeling of the process. In *Proceedings of the 2nd IFSA Congress,* pp. 86–89, Tokyo, 1987.
43. S. Tzafestas. Fuzzy and neural approaches to robot control. In *Proceedings of the 1st ECPD International Conference on Advanced Robotics and Intelligent Automation,* pages 34–55, Athens, September 1995.
44. M. Sugeno and M. Nishida. Fuzzy control of a model car. *Fuzzy Sets and Systems,* 16:103–113, 1985.
45. T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15(1):116–132, January 1985.
46. R. Tanscheit and E.M. Scharf. Experiments with the use of a rule-based self-organizing controller for robotics applications. *Fuzzy Sets and Systems,* 26(1):195–214, 1988.
47. T. Fukuda, Y. Komata, and T. Arakawa. Stabilization control of biped locomotion robot based learning with GAs having self-adaptive mutation and recurrent neural network. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation,* pp. 217–222, Albuquerque, NM, April 1997.
48. T. Ozaki, T. Suzuki, T. Furuhashi, S. Okuma, and Y. Uchikawa. Trajectory control of robotic manipulators using neural networks. *IEEE Transactions on Industrial Electronics,* 38(3):195–202, June 1991.
49. T. Takagi and M. Sugeno. Derivation of fuzzy control rules from human operator's control actions. In *Proceedings of the IFAC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis,* pp. 55–60, 1983.
50. S. Tzafestas and N.P. Papanikolopoulos. Incremental fuzzy expert PID control. *IEEE Transactions on Industrial Electronics,* 37(5):365–371, October 1990.
51. V. Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks,* 3:671–692, 1990.
52. M. Vukobratović and B. Karan. Experiments with fuzzy logic robot control with model-based dynamic compensation. In *Proceedings of the 25th International Symposium on Industrial Robots,* pp. 215–222, Hanover, 1994.
53. M. Vukobratović and B. Karan. Experiments with fuzzy logic robot control with model-based dynamic compensation in nonadaptive decentralized control scheme. *International Journal on Robotics and Automation,* 11(3):118–131, 1996.
54. M. Vukobratović, D. Stokić, and N. Kirćanski. *Non-Adaptive and Adaptive Control of Manipulation Robots.* Springer, Berlin, 1985.
55. W. Li, C. Ma, and F.M. Wahl. A neuro-fuzzy system architecture for behaviour-based control of a mobile robot in an unknown environment. *Fuzzy Sets and Systems,* 87(2):133–140, 1997.

56. W.T. Miller. Sensor-based control of manipulation robots using a general learning algorithm. *IEEE Journal on Robotics and Automation,* 3:157–165, April 1987.
57. C.W. Xu and Y.Z. Lu. Fuzzy model identification and self-learning for dynamic systems. In *IEEE Transactions on Systems, Man and Cybernetics,* 17(4):683–689, August 1987.
58. L.A. Zadeh. Fuzzy sets. *Information and Control,* 8:338–353, 1965.
59. L.A. Zadeh. Fuzzy algorithms. *Information and Control,* 12:94–102, 1968.
60. L.A. Zadeh. Quantitative fuzzy semantics. *Information Sciences,* 3:159–176, 1971.
61. L.A. Zadeh. A rationale for fuzzy control. *Journal of Dynamic Systems, Measurement, and Control,* 94:3–4, 1972.
62. L.A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, 3(1):28–44, January 1973.
63. L.A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences,* 8:199–249, 301–357, 1975.
64. L.A. Zadeh. Fuzzy logic. *IEEE Computer,* 21(4):83–93, April 1988.
65. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Program.* Springer Verlag, Berlin, 1994.