A Multi-Resolution Radial Basis Function Network for Self-Organization, Defuzzification, and Inference in Fuzzy Rule-Based Systems

Sukhan Lee

Depts. of EE-Systems and CS
University of Southern California
Los Angeles, California 90089-0781

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

Abstract

The merit of fuzzy rule based systems stems from their capability of encoding qualitative knowledge of experts into quantitative rules. Recent advancement in automatic tuning or self-organization of fuzzy rules from experimental data further enhances their power, allowing the integration of the top-down encoding of knowledge with the bottom-up learning of rules. In this paper, methods of self-organizing fuzzy rules and of performing defuzzification and inference is presented based on a multi-resolution radial basis function network. The network learns an arbitrary input-output mapping from sample distribution as the union of hyper-ellipsoidal clusters of various locations, sizes and shapes. The hyper-ellipsoidal clusters, representing fuzzy rules, are self-organized based on global competition in such a way as to ensure uniform mapping errors. The cooperative interpolation among the multiple clusters associated with a mapping allows the network to perform a bidirectional many-to-many mapping, representing a particular form of defuzzification. Finally, an inference engine is constructed for the network to search for an optimal chain of rules or situation transitions under the constraint of transition feasibilities imposed by the learned mapping. Applications of the proposed network to skill acquisition are shown.

1. Introduction

Fuzzy rule based systems have been regarded as a better, if not best, alternative to conventional expert systems based on symbolic logics due to their descriptive power associated with fuzzy membership functions [19]. They provides more precise encoding of expert knowledge as well as smoother setting of decision boundaries beyond the binary logics that conventional expert systems offer. Furthermore, the capability of transforming qualitative knowledge into quantitative rules with both fuzzy membership functions and defuzzification schemes makes fuzzy rule based systems a powerful tool for connecting between symbolic and numeric world. Consequently, during the past decade, there have been demonstrated numerous applications of fuzzy rule based systems to real world problems that successfully transform expert knowledge into machine intelligence [13,16,18].

Recent advancement in automatic tuning or self-organization of fuzzy rules from experimental data further enhances the power of fuzzy rule based systems, since it allows the integration of the top-down knowledge encoding with the bottom-up rule learning. In other words, rules from expert knowledge of limited precision can be fine-tuned based on learning from experimentation. To this end, researchers have found it attractive to integrate the well-established learning capability of neural networks into the architecture of fuzzy rule based systems [14, 15, 17, 21]. This has brought forth, so called, neuro-fuzzy or fuzzy-neuro architectures, where, e.g., fuzzy membership functions of a Gaussian shape are implemented with neural networks of sigmoidal activation functions. Neural networks are known to be capable of learning arbitrary input-output mappings from training, where the training aims at achieving accuracy as well as generalization of the mapping represented by training samples [7, 8, 9, 10]. Neural networks have been applied more to the bottom-up process of learning mapping relations, in contrast to fuzzy rule based systems which concern primarily the top-down process of implementing expert knowledge. However, a compelling need to integrate the two exists, so as to combine the strengths of both and compensate for the weakness of another. For instance, there is a need to refine fuzzy rules generated from imprecise knowledge, and, also, there is a need to incorporate higher level knowledge in neural network learning to achieve better generalization under a limited number of samples [5, 11].

It is not so difficult to realize that radial basis function neural networks, which describe a mapping in terms of a weighted summation of a set of radial basis functions such as Gaussian or inverse distance functions, may function as a general form of neuro-fuzzy systems [20]. In radial basis function networks, an input-output mapping is represented as a set of locally defined sample clusters described by a radial basis function, such that such local clusters can serve as mapping rules in a fuzzy form [22, 23, 24, 25, 26, 27, 28, 29].

Attempt has already been made to generate fuzzy rules from a radial basis function network [4]. However, the full capacity of radial basis function networks in conjunction with fuzzy rule based systems is yet to be explored. The current practice of radial basis function networks often poses the following problems to solve:

1. The representation of a mapping based on the uniform size of local clusters may be neither efficient, due to the possibility of generating an unnecessarily large number of local clusters, nor easy for achieving a balance between the accuracy and generality in mapping.
2. The network may not be capable of representing a general mapping including a many-to-many mapping, and may not allow an inverse mapping of an arbitrary nonlinear mapping.
3. No clear connection between the mapping in radial basis function networks and the defuzzification in fuzzy rule based systems has been established. Furthermore, no attempt has been made to date to construct an inference engine associated with radial basis function networks. In this paper, methods of self-organizing fuzzy rules and of performing defuzzification and inference is presented based on a multi-resolution radial basis function network. The network learns an arbitrary input-output mapping from sample distribution as the union of hyper-ellipsoidal clusters of various locations, sizes and shapes. The hyper-ellipsoidal clusters, representing fuzzy rules, are self-organized based on global competition in such a way as to ensure uniform mapping errors. The cooperative interpolation among the multiple clusters associated with a mapping allows the network to perform a bidirectional many-to-many mapping, representing a particular form of defuzzification. Finally, an inference engine is constructed for the network to search for an optimal chain of rules or situation transitions under the constraint of transition feasibilities imposed by the learned mapping.

This paper is organized as follows: In Sec. 2, the proposed multi-resolution radial basis function network is presented, where the construction of local clusters based on globally competitive learning and the locally cooperative interpolation for a many-to-many mapping are described in subsections. In Sec. 3, the inference engine based on dynamic path planning is presented. In Sec. 4, applications of the proposed network to skill acquisition are shown as case studies.

2. Multi-Resolution Radial Basis Function Network

An input-output mapping, $y=f(x)$, $x \in R^n$ and $y \in R^m$, can be represented by a collection of samples, (x,y) , distributed in the $(n+m)$ -dimensional input-output space, $z=(x,y)$. The representation of a mapping based on a collection of samples in the input-output space is quite general, which can be applied to arbitrary mappings with nonlinear as well as many-to-many relations.

The proposed Multi-Resolution Radial Basis Function (MRBF) network aims at learning the mapping relation between $\boldsymbol{x}$ and $\boldsymbol{y}$ by approximating the sample distribution, $z=(\boldsymbol{x},\boldsymbol{y})$, based on the union of local sample clusters of a hyper-ellipsoidal shape (refer to Fig. 1). More precisely, a local cluster, C, is represented as a hyper-ellipsoid, $C(z;c,\Sigma,r)$, with $c\in\boldsymbol{R}^{n+m}$ representing the center or reference point of the cluster, Σ, the $(n+m)\times(n+m)$ dimensional positive definite shape matrix, and $r\in\boldsymbol{R}$, the size or radius of the cluster, as follows:

$$C(z;c,\Sigma,r)=\{z\in\boldsymbol{R}^{n+m}|(z-c)^T\Sigma(z-c)\leq r^2\}$$

The center, c, and shape matrix, Σ, of a cluster are respectively from the mean and covariance of the local samples that belong to the cluster. The local clusters thus generated play a role as fuzzy rules.

The training of the MRBF network for learning an input-output mapping involves the self-organization of the necessary number of local clusters of various locations, shapes, and sizes in such a way as to achieve the desired mapping accuracy uniformly throughout the domain of interest. The self-organization of the network involves the automatic recruitment of local clusters to cover the mapping samples, while the locations, shapes and sizes of clusters are updated iteratively to improve the mapping accuracy uniformly to a desired level. The algorithm for self-organizing the network is referred to here as *Globally Competitive Clustering Algorithm*, as described in detail in Section 2.1.

A mapping is defined in the MRBF network firstly by selecting the local clusters in which the given input, $\boldsymbol{x}$, reside within their boundaries. Each of the selected local clusters generates an output, $\boldsymbol{y}$, by taking the sample point associated with the given input, $\boldsymbol{x}$, that represents the minimum Mahalanobis distance [3] from its center, where the distance, $d(z;C)$, is measured in terms of the shape matrix of individual local clusters:

$$d(z;C)=\left[(z-c)^T\Sigma(z-c)\right]^{\frac{1}{2}}$$

The outputs from individual local clusters corresponding to the given input are then grouped based on their degree of proximity. That is, each group includes those outputs of close proximity that are to be fused into a single output for the group. The existence of multiple groups in a mapping thus implies the existence of multiple outputs, resulting in an one-to-many mapping. The above algorithm for mapping with the MRBF network is referred to here as *Locally Cooperative Interpolation Algorithm* as described in detail in Section 2.2.

2.1 Globally Competitive Learning of Local Clusters

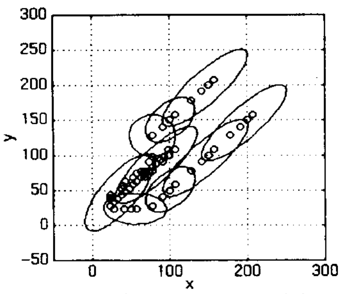

Figure 1: A many to many non-linear mapping represented by a collection of samples (small circles) in the $(\boldsymbol{x},\boldsymbol{y})$ space is approximated by a collection of hyper-ellipsoidal clusters of various sizes, shapes, and locations.

The globally competitive learning of local clusters for an input-output mapping is based on the following concept:

1. The location of a cluster center is determined by the samples assigned to the cluster, where the assignment of a sample to a cluster is based on the winner-take-all inter-cluster competition. That is, a sample will be assigned to the cluster having the minimum Mahalanobis distance from its center to the sample.
2. The number of clusters are determined in such a way that every sample which is not an outlier is covered by at least one cluster. A sample is said to be covered by a cluster if the sample is located inside the cluster boundary.
3. The shape of a cluster is determined based on the covariance of the samples that belong to the cluster.
4. The size of a cluster is determined based on the mapping accuracy of the samples covered by the cluster. Should a cluster fail in the evaluation of mapping accuracy, its size should be reduced accordingly.
5. The locations, shape, and sizes of individual local clusters are determined based on hierarchical learning. It starts with a small number of local clusters of a large size and spherical shape, but refines and reduces their shapes and sizes, respectively, while increasing the number of local clusters by recruiting new members, as necessary for the desired mapping accuracy.

There can be many ways of constructing an algorithm for the implementation of the above learning concept. Here, an algorithm based on the iterative update of the location, shape, and size of a cluster is presented, as follows:

Algorithm: Clustering with Iterative Update

Input:

$\{z_i | z_i=(x_i,y_i), i=1,\ldots,N_d\}$: a set of samples stored in a relational data-base.

r_0 : the initial cluster size.

E_0 : the error threshold for overall mapping performance in L_1 - or L_∞ -norm.

Output:

$\{C_j | C_j=C(z;c_j,\Sigma_j,r_j), j=1,\ldots,N_c\}$: a set of local clusters defined in terms of their centers, shapes, and sizes.

Method:

Step 1: Initialization.

Pick up a sample, z_i, randomly and assign a cluster, C_i, such that $c_1=z_i$, $\Sigma_1=I_{n+m}$, and $r=r_0$. Set the current number of learning cycles, k, as $k=1$, and the current number of clusters, N_c, as $N_c=1$.

Step 2: Self-Organization of Clusters with the Iterative Update of their Locations and Shapes.

Repeat the following process for the set of stored samples at the k th learning cycle:

a. Pick up a sample, z_i, randomly.

b. Determine the cluster, C_w having the minimum Mahalanobis distance from its center to z_i as the winner of the sample, as follows: $d(z_i;C_w)=\min_C d(z_i;C)$.

c. If z_i is within the boundary of C_w, i.e., $(z_i-c_w)\Sigma_w^{-1}(z_i-c_w)\le r_w^2$, then, adjust the current center, c_w, and the current shape, Σ_w of the cluster by

$$c_w \leftarrow c_w + \alpha(n_w) \cdot (z_i - c_w)$$

$$\Sigma_w \leftarrow \Sigma_w + \beta(n_w) \cdot (z_i - c_w)(z_i - c_w)^T,$$

where the learning rates, $\alpha(n_w)$ and $\beta(n_w)$, are monotonically decreasing positive functions of n_w, the winning number representing the number of times that C_w has been selected as the winner. The size of the cluster remains same as before, while the shape matrix should be normalized after the update in order to keep the volume of the cluster unchanged.

d. Otherwise, set $N_c = N_c + 1$ and generate a new cluster, C_{N_c}, such that $c_{N_c} = z_i$, $\Sigma_{N_c} = I_{n+m}$, and $r_{N_c} = r_k$

Step 3. Evaluation of Overall and Local Mapping Performance

Present all or part of samples randomly to the network to evaluate the overall mapping performance, E, defined in terms of L_1 - or L_∞ -norm of the mapping errors between the reference and network outputs. The network outputs are obtained by the cooperative interpolation mapping algorithm described in Section 2.2. At the same time, evaluate the mapping performance, E_j, of individual local clusters, C_j, in terms of L_1 - or L_∞ -norm of the mapping errors for the samples covered by C_j, for $j = 1, \ldots N_c$.

Step 4. Repetition of Update Process for Error Convergence

If the overall performance is satisfactory, i.e., $E \leq E_0$, then stop. Otherwise, delete C_j from the current list of clusters, if $E_j > \frac{1}{N_c} \sum_{l=1}^{N_c} E_l$ for $j = 1, \ldots, N_c$. Set $N_c \leftarrow N_c - N_d$ with N_d representing the number of deleted local clusters, $r_k \leftarrow \Upsilon r_k$ with Υ representing a positive number of less than unity, and $k = k+1$. Then, go to step 2 and repeat the process with surviving clusters of updated shapes and sizes.

Fig. 2 illustrates, in a snap shot fashion, the generation of two different sizes of clusters based on the above clustering algorithm. In Fig. 2(a), 7 clusters of a relatively large size are generated initially. In Fig. 2(b), Clusters 3 and 7 are deleted based on the local performance evaluation. Then, in Fig. 2(c), four smaller clusters are newly generated to cover the samples of the deleted clusters. Notice that the shapes of the larger clusters are changed continuously as learning progresses.

2.2 Locally Cooperative Interpolation for Mapping

The set of local clusters self-organized by the MRBF network provides a compact representation of an input-output mapping in an approximated and generalized form. In this section, it will be shown that the MRBF network allows an accurate estimation of mapping output corresponding to a given input based on the cooperative interpolation among those clusters that contain the given input within their boundaries:

Step 1: Forming a Set of Active Clusters.

For a given input x_s, all the clusters that intersect the hyperplane, L_s, defined by $L_s=\{z=(x,y)\in R^{n+m} | x=x_s\}$, become active. Let us denote the set of active clusters for a given input, x_s, as $F(x_s)$:

$$F(x_s)=\{C_j | L_s\cap C_j\neq 0, j=1,\ldots,N_c\}$$

Step 2: Estimation of an Output from Each of Active Clusters.

Estimate an output, y_j, from each of the active clusters, C_j, in the set $F(x_s)$, by identifying the point, $z_j=(y_j,x_s)$, on the hyperplane L_s such that $d(z_j;C_j)=\min_{z\in L_s} d(z_j;C_j)$.

That is, we select the point, z, on the hyperplane, L_s, that represents the minimum Mahalanobis distance from the center of the cluster.

Step 3: Grouping the Outputs from Individual Clusters.

Collect all the outputs, y_j, generated from individual clusters, C_j, in the set $F(x_s)$, and partition them into groups, G_i, based on the measure of their proximity. The groups thus generated are referred to here as the interpolation groups.

Step 4: Fusing the Outputs of Each Interpolation Group.

The outputs, $\{y_j\}$, that belong to an interpolation group, G_i, are fused into a single output, $\hat{y}_i$, based on the following:

$$\hat{y}_i=\sum_{y_j\in G_i}\xi_j y_j ,$$

where ξ_j. $\sum_{y_j\in G_i}\xi_j=1$, is the summation weight determined based on the Mahalanobis distance of y_j, $d(z_j;C_j)$, and the number of samples in C_j for all y_j, $y_j\in G_i$. Note that the above grouping and fusion

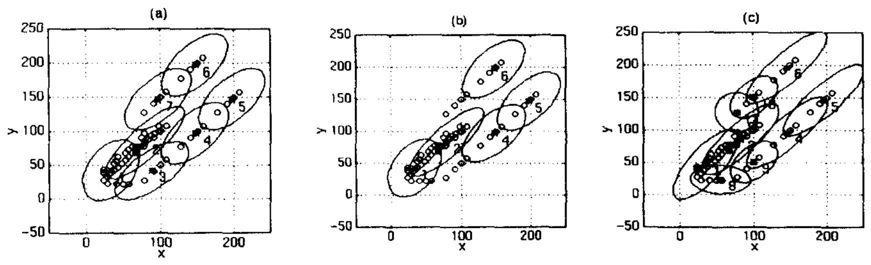

Figure 2: Snap shots representing the generation of clusters of various shapes and sizes. The asterisks and small circles represent respectively the cluster centers and the sample points: (a) The result of clustering based on the data with an initial radius of 40. (b) Clusters 3 and 7 are deleted because of their poor performances compared to other existing clusters. (c) Clusters 8-11 are newly generated with the reduced radius of 30.

processes can be further elaborated in terms of a data fusion technique based on statistics, owing to the fact that each local cluster represents the distribution of samples within. For instance, it is possible to define the proximity between the two outputs based on the concept of statistical support between the two data, such that the grouping can be done by analyzing the supporting structure of outputs[12].

The above algorithm represents a cooperative procedure among the local clusters to estimate mapping outputs accurately. Note that each interpolation group generates its own output, which separates out possible multiple outputs existing in many-to-many mappings. Furthermore, the above algorithm can be applied equally well to both forward and inverse mappings, since x and y can be interpreted respectively either as input and output or as output and input without losing generality. In short, the MRBF network is capable of performing bidirectional many-to-many mappings. This mapping process can be considered as a general form of defuzzification.

Fig. 3 illustrates the locally cooperative interpolation mapping algorithm described above. It shows the retrieval of three outputs for the given input x_s. First, three clusters (Clusters 4, 5, and 6) become active for the given input. Then, each of the three clusters generates its own output by finding the minimum Mahalanobis distance to the given manifold, $x=x_s$. Then, the three estimated outputs from individual clusters are grouped into two interpolation groups: one group by cluster 6, and the other group by cluster 4 and 5, based on the measure of their proximity. The three groups yield two unique outputs $\hat{y}_1$ and $\hat{y}_2$, where $\hat{y}_2$ is obtained by fusing the two outputs in the same group.

3. Inference Engine

A mapping may represent situation transition rules. For instance, x and y may represent the current and next situations, $s(k)$ and $s(k+1)$, respectively, while the current action, $u(k)$, is associated implicitly with each pair of $(s(k),s(k+1))$. In this case, the set of local clusters learned by the MRBF network represents the feasible situation transitions by available actions, and defines the feasible situation transition manifold (FSTM) in the $(x=s(k),y=s(k+1))$ space.

To infer a sequence of actions or situation transitions that transforms the current situation to the goal situation is one of the key functions in rule-based systems, often carried out by a, so called, inference engine. The mapping represented by MRBF is in a continuous domain, such that the search for a sequence of transitions from the initial situation, s_i, to the goal situation, s_g, may become excessive in terms of time complexity. Here, a new inference engine is constructed especially for the MRBF representation of a mapping, which allows to search for an optimal transition sequence efficiently in a continuous domain. The proposed engine is based on the dynamic planning of an optimal path in the situation-time space, (s,k), under

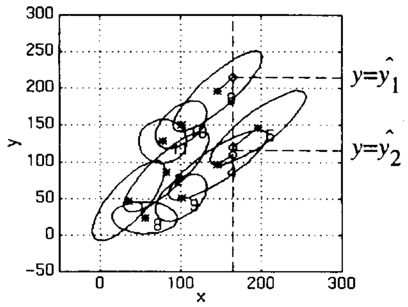

Figure 3: The retrieval of mapping outputs in the MRBF network based on the proposed locally cooperative interpolation mapping algorithm. The asterisks and small circles represent respectively the cluster centers and outputs of active clusters. The solid straight lines represent the minimum Mahalanobis distances.

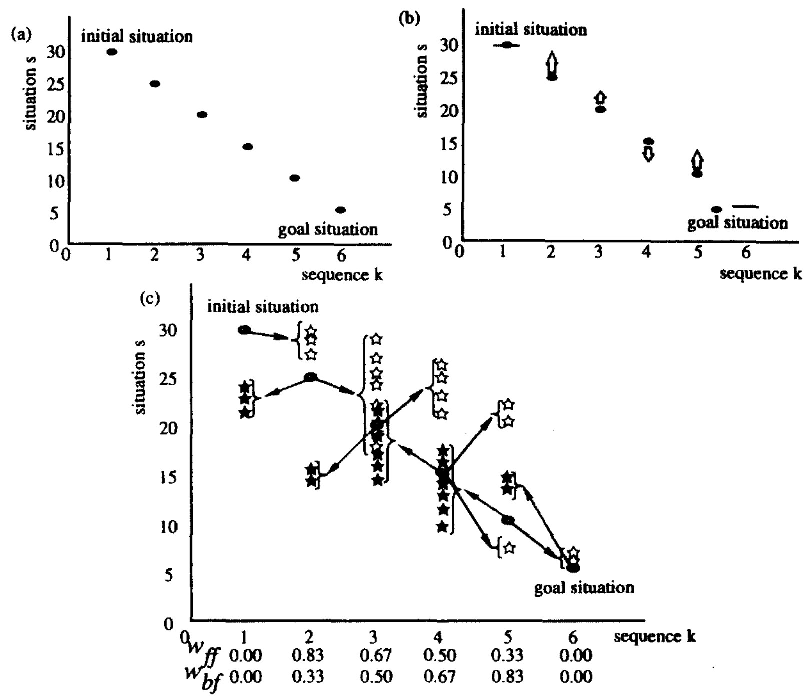

Figure 4: An illustration of the dynamic path planning algorithm. (a) An initial particle assignment along 6 time steps. (b) Each particle moves toward an equilibrium. (c) The feasible situation transition regions defined by forward and inverse mappings. The hollow stars denote the feasible transition regions from forward mappings, while the shadow stars denote the feasible transition regions from inverse mappings.

the constraint imposed by FSTM. To be more specific, the concept of Dynamic Path Planning (DPP) is introduced first in the following:

Let us assume, as the initial step, that an arbitrary sequence of m transitions, such as a straight line path from the initial to the goal situations, is set in the situation-time space, as illustrated in Fig. 4.(a). In this case, neither the constraint imposed by FSTM nor the minimum cost requirement is taken into consideration. Therefore, the initial transition sequence thus set may be neither feasible nor optimal.

Consider now that there exist charged particles distributed along the path, one at each time step, as shown by the black dots in Fig. 6(a). Each particle is allowed to move only on the situation space defined at its designated time. To satisfy the constraint imposed by FSTM, the particle at the k th step (or, briefly, the particle k) must reside in the intersection of two regions, one reachable from the particle $k-1$ and the other reachable to the particle $k+1$ by situation transitions, as illustrated in Fig. 6(c). The region reachable from the particle $k-1$, referred to here as the forward region at k , can be defined from the forward mapping of the MRBF network at $s(k-1)$, whereas the region reachable to the particle $k+1$, referred to here as the backward region at k , can be defined from the inverse mapping of the MRBF network at $s(k+1)$. In summary, each particle along the path must reside in the intersection of the two regions associated with it, the forward and backward regions, defined respectively by the previous and next steps of particles, except those fixed particles at the initial and goal steps.

In the proposed dynamic path planning, the two forward and backward regions associated with a particle play a role as attractors, such that the particle gradually converges to the intersection of the two. It should be noted that the two regions are subject to a constant change along with the continuous update of the locations of individual particles. So named dynamic path planning to represent path planning in an environment with open channels varying dynamically in time.

To move a particle toward the regions of attraction, each region generates an attraction force based on the potential field assigned to each region. A potential field has a zero slope inside the region, while having a sharp slope outside, as shown in Fig. 5(d). As time progresses, particles move toward the intersection of their own forward and backward regions, as seen by Fig. 4, in spite of the continuous variation of the regions of attraction along with the motion of particles. The movement of particles will be stopped when they all reside in the intersection of two attraction regions, representing feasible transitions.

In the case where not only a feasible but also an optimal transition sequence is searched for, the potential fields generated for the feasibility conditions, i.e., the forward and backward regions associated with individual particles, should be combined with the potential field generated for the given optimality conditions such as the minimum in path length. Note that there is no force generated from the feasibility conditions once a particle is reached inside the intersection of forward and backward regions, such that only the force generated for the optimality condition is applied to a particle to drive it to an optimal location.

During the process of dynamic path planning, all the particles along the path, except the initial and goal particles, move simultaneously by the forces generated by the potential fields. The potential fields themselves are subject to continuous variation until particles reach the intersection of their forward and backward regions. To make the dynamic path planning more efficient, the following heuristics is incorporated: 1) the nearer a particle is to the initial one in sequence, the greater it is influenced by the attraction force from its forward region, and 2) the nearer a particle is to the goal one in sequence, the greater it is influenced by the attraction force from its backward region. This heuristics is implemented by assigning a diminishing weight sequence from the initial to the goal to the forces generated by forward regions, while assigning a diminishing weight sequence from the goal to the initial to the forces generated by backward regions, as illustrated in Fig. 4(c).

Since it may not be known initially how many time steps are required for the search, the number of steps are increased one by one, starting from one, as required by the search.

3.1 Dynamic Path Planning Algorithm

The potential field, $P_f(k)$, from the feasibility condition represented by the forward region at k can be formulated as follows: 1) Based on the learned MRBF network, obtain the range of $s(k)$ for the given $s(k-1)$ by forward mapping. The range of $s(k)$, representing the forward region of $s(k)$, is defined in terms of discrete points as a grid, as shown in Fig. 5(a). 2) At each discrete point of the forward region, define a Gaussian kernel of fixed variance and height. Make sum of all the Gaussian kernels generated for the grid, as shown in Fig. 5(b). 3) Flatten the Gaussian sum by cutting the top with the threshold, θ_k, where θ_k

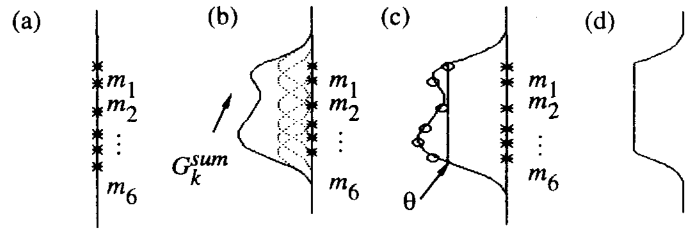

Figure 5: A potential field: (a) the feasible transition region, (b) the sum of the gaussian kernels, (c) flattening with the threshold value, θ, and (d) final potential field.

represents the minimum value of the Gaussian sum at grid points, as shown in Fig. 5(c). 4) The flattened Gaussian sum represents the potential field from the forward region, as shown in Fig. 5(d). The potential field, $\boldsymbol{P}_{bf}(k)$, due to the backward region at k can be formulated by the same way as above, except that the backward region of $s(k)$ is defined from the inverse mapping with the given $s(k+1)$. Now, the total potential field, $\boldsymbol{P}_f(k)$, at k due to the feasibility condition can be defined as the weighted sum of $\boldsymbol{P}_{ff}(k)$ and $\boldsymbol{P}_{bf}(k)$, as follows:

$$\boldsymbol{P}_f(k)=w_{ff}(k)\boldsymbol{P}_{ff}(k)+w_{fb}(k)\boldsymbol{P}_{fb}(k)$$

where $w_{ff}(k)$ and $w_{fb}(k)$ are monotonically decreasing and increasing functions of k, respectively. Then, the force that drives the particle k to the intersection of forward and backward regions can be defined as the gradient of $\boldsymbol{P}_f(k)$ in the situation space.

On the other hand, the optimality condition such as the minimum path length can be incorporated into the search by formulating a potential field, $\boldsymbol{P}_o$, representing the square sum of the distances between the two consecutive particles. Note that $\boldsymbol{P}_o$ is a global measure over k, whereas $\boldsymbol{P}_f(k)$ is a local measure at k. The force that drives the particle k to the minimum of $\boldsymbol{P}_o$ can be defined by the negative gradient of $\boldsymbol{P}_o$ with respect to the situation associated with the particle k. This gradient turns out to be a function of only $s(k-1)$ and $s(k+1)$, the immediate neighbors of the particle k, same as the force from the feasibility conditions. Then the total force, $f(k)$, that drives the particle k can be obtained by the weighted sum of the two forces, $f_f(k)$ and $f_o(k)$, respectively from the feasibility condition and optimality condition:

$$f(k)=w_f f_f(k)+w_o f_o(k),$$

where w_f is set to be much larger than w_o.

The following steps summarize the algorithm:

Step 1. Assign 2 to the required number of steps, n. If an estimate of the minimum required number of steps is available, then assign the estimate to n. Set s_1 and s_n to be the initial and goal situations, respectively.

Step 2. Assign an arbitrary initial path. However, it is preferred that the initial path is set to be the one that maximize the given performance criteria, e.g., a straight line in the case of the minimum length criterion, but without taking the feasibility condition into consideration. Set $Loop=1$.

Step 3. For each k, $1<k<n$, calculate the forces at k due to the feasibility condition, $f_f(k)$ and due to the optimality condition, $f_o(k)$, based on the methods described above.

Step 4. For each k, $1<k<n$, move the particle k toward the direction specified by the total force, $f(k)$ by a small amount proportional to the strength of the total force.

Step 5. Check whether or not an equilibrium is reached, where an equilibrium is said to be reached when the amount of total force at each k is less than the predefined threshold. If an equilibrium is reached, then check whether or not the generated path is acceptable. If yes, stop.

Step 6. If $Loop$ is less than the predefined maximum loop count, then set $Loop=Loop+1$ and go to Step 3.

Step 7. Set $n=n+1$ and $Loop=1$. Set up a new particle at a new step inserted between the arbitrary chosen time steps failed in feasible transition with a random assignment of a situation. Go to Step 3.

Note that the computation of driving forces at each k can be done simultaneously in parallel. Fig. 6 illustrates an implementation of the proposed dynamic path planning algorithm based on massive parallel

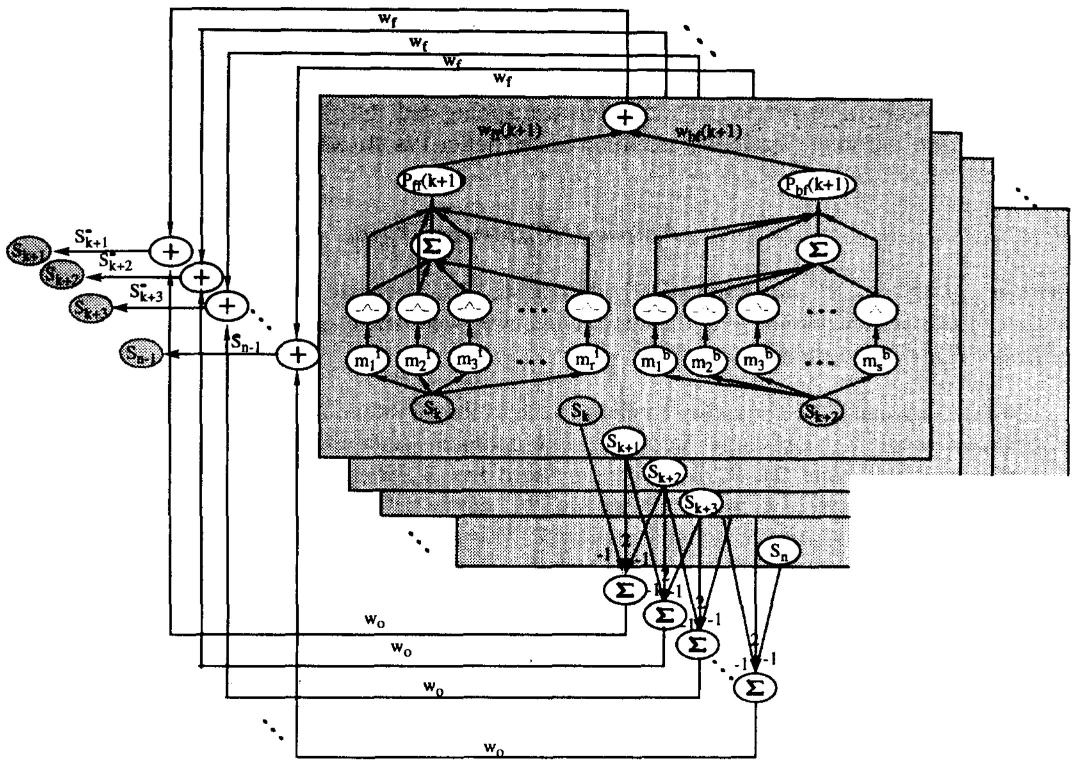
Figure 6: The neural network architecture for computing the dynamic path planning algorithm. For clarity, a state s_k, $1 \leq k \leq n$, is denoted by more than one node.

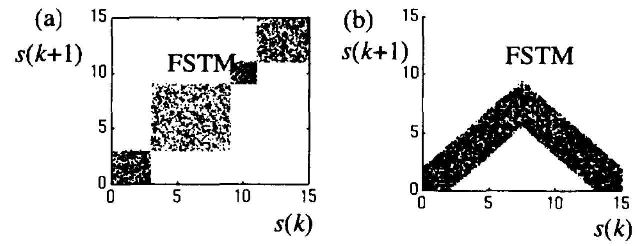

Figure 7: Examples of feasible situation transition manifolds (FSTM) representing two hypothetical dynamic systems.

computation of driving forces with a neural network.

The validity of the proposed dynamic path planning algorithm is verified based on the hypothetically generated FSTM shown in Fig. 7. The hyper-ellipsoidal clusters of various sizes and shapes, learned by the MRBF network to represent FSTM, are shown in Fig. 8. Then, a feasible transition sequence from the initial and goal situation is searched for by applying the dynamic path planning algorithm to the learned MRBF network. The results are shown in Fig. 9, where optimal paths with 5 situation transitions are found for both cases, starting with the initially assigned 2 transitions. Note that the search for an optimal path especially for the case shown in Fig. 9(a) is rather difficult due to the fact that the optimal transitions occur at the corner of FSTM.

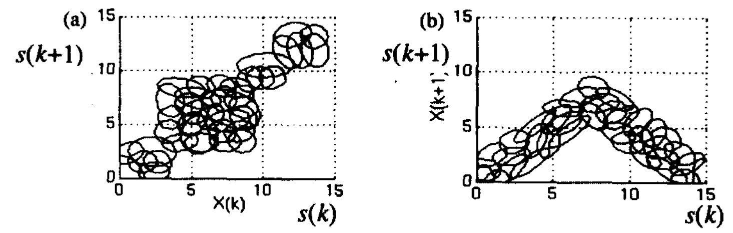

Figure 8: The result of multi-resolution clustering based on the data of Fig. 7.

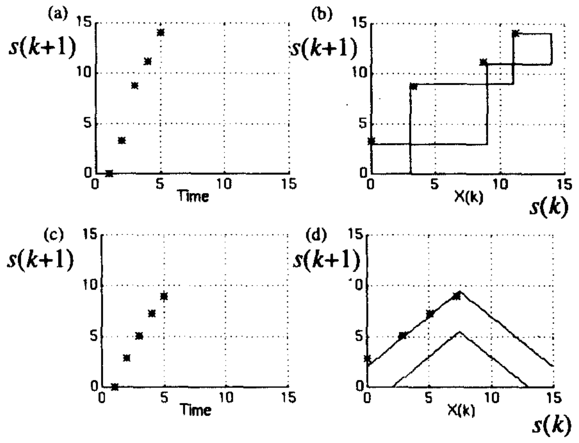

Figure 9: The optimal path found for the dynamic systems shown in Fig. 7.(a) s_1=0 and s_n=14 .(b) s_1=0 and s_n=9 Notice that the lines in (b) and (d) are drawn only to show the approximate boundaries of the FSTM in Fig. 7.

Finally, the dynamic path planning algorithm may be subject to local minima. However, the algorithm may be less vulnerable to the local minima problem, since, at a local minimum, the algorithm recruits a new step to be inserted into a path such that an ill condition caused the local minimum can be broken. Furthermore, the dynamic change of potential fields during the search process helps to break up the local minima at individual particles.

4. Case Study

4.1 Skill for Non-Holonomic Motion Planning

Controlling a dynamic system involving nonholonomic constraints, such as a car-like robot, draws much attention from many researchers [4,6,7]. Here, we apply the proposed MRBF network to the acquisition of a motion planning skill by a car-like robot navigating in a cluttered environment.

A car-like robot, $\mathcal{A}$, navigating in the $2d$ workspace with four wheels is shown in Fig. 10. $\mathcal{A}$ has three state variables, (x,y,θ), where x and y are the position of the midpoint between the two rear wheels, and θ denotes the angle of the robot around the z-axis, and two control variables, (v,φ), where v denotes the speed of the car, and φ is the turning angle of the steering wheel. $\mathcal{A}$ is subject to non-holonomic constraints:

C1: $-\dot{x}\cos(\theta)+\dot{y}\cos(\theta)=0$

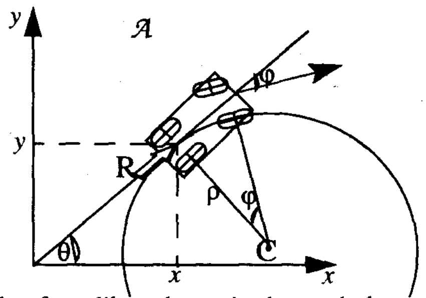

Figure 10: The classic example of car-like robot $\mathcal{A}$ in the nonholonomic motion planning. R is the midpoint between two rear wheels. C is the rotation center of $\mathcal{A}$.

C2: $\dot{x}^2+\dot{y}^2-\left(\rho_{min}^2\dot{\theta}^2\right)\geq 0$, where ρ_{min} is the minimum of the turning radius, ρ, of the car. (Refer to Fig. 10.)

C1 and C2 are non-integrable and referred to non-holonomic equality constraint and nonholonomic inequality constraints respectively [2].

Data Collection

An input-output space is formed by the Cartesian product of the current situation space, $(x(k),y(k),\theta(k))$, the current action space, $(v(k),\varphi(k))$, and the next situation space, $(x(k+1),y(k+1),\theta(k+1))$. The data representing FSTM can be constructed may be collected either from the kinematic and dynamic model of the car-like robot, or from the actual experimentation.

There are two ways of representing the FSTM for the above example. First, we define the feasible transitions of situations to be dependent not only on the available control actions but also on the environment such as obstacles. This implies that the transition is situation-dependent. For instance, the next robot pose that can be reached from the current robot pose depends on the obstacles near the current robot pose, in addition to the available current action. As shown in Fig. 11, to construct an FSTM, we need data collected from a large number of robot poses distributed over the workspace. Second, we define the feasible transitions of situations to be dependent only on the available control actions but independent of the environment. For instance, we can define the FSTM of a car-like robot at its particular pose, as shown in Fig. 12 (a) and (b). Then, the feasible transitions of the robot at the current pose can be identified from the FSTM by applying the kinematic transformation (i.e., rotation and translation) between the reference and the current pose. and by taking into account the obstacles near the current robot pose, as shown in Fig. 12.

In this paper, the data are generated from the kinematic model of a car-like robot. This is done by selecting a robot pose and control action randomly at time k and computing its pose at time k + dt based on the kinematic model, while checking whether or not the result satisfies the nonholonomic constraints as well

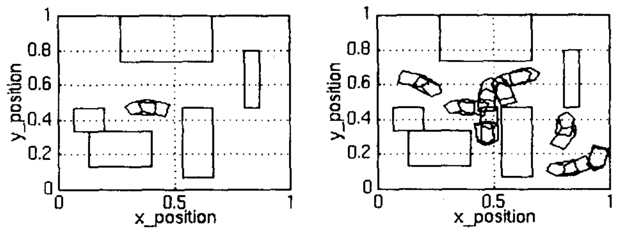

Figure 11: Ten examples of feasible situation transitions randomly generated from the kinematic model. The rectangles denote obstacles in the workspace.

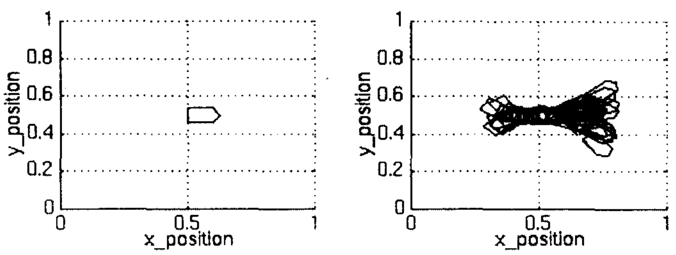

Figure 12: (a) The reference pose where the FSTM is defined without taking the environment into consideration. (b) The projection of the FSTM defined in the augmented situation onto the situation space.

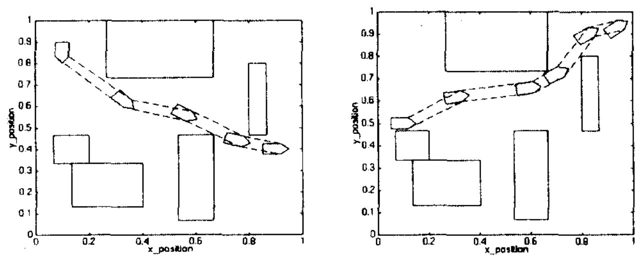

Figure 13: Two example results for the car-like robot motion planning.

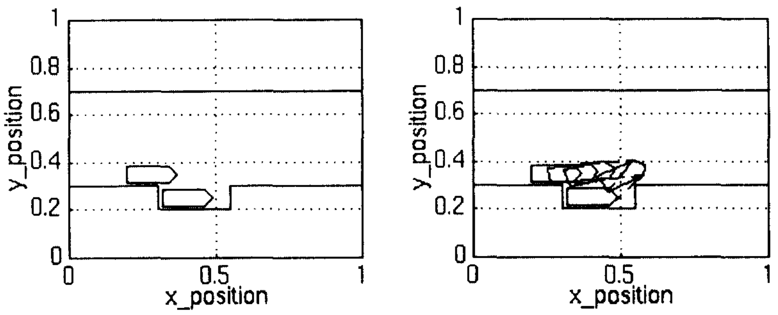

Figure 14: (a) Initial and goal configurations are given. (b) A sequence of configuration transitions are generated connecting the initial and gaol configurations.

as the obstacle avoidance conditions.

Experimental Result

Fig. 13 illustrates two examples of the feasible paths generated based on the FSTM and the dynamic path planning algorithm. Fig. 14 illustrates the classic parallel parking skill discovered by the robot. Note that Fig. 13 and Fig. 14 show the snap shots of the transitions from the given initial and goal configurations. However, the dashed lines drawn between configurations are used to illustrate the connection of the snap shots, but not the actual path taken. There are total of five states depicted in the trajectory in the parallel parking example, including the initial and goal states. The backward motion to the goal configuration from its previous state is shown.

Note that the dynamic path planning algorithm functions as a bidirectional A* search algorithm, but with the feasibility of massive parallel computation. Furthermore, for the search-based approaches with the discretization of configuration space and control parameters, the search space becomes potentially very large due to the small size of discretization required for precision. For the proposed approach, although the input data are represented in a discrete form, the locally cooperative algorithm is able to reconstruct feasible transitions in a continuous domain. This capability allows the proposed approach to represent a skill in a continuous domain.

4.2 Skill for Telemanipulation

To further verify the validity of the proposed approach, we have obtained experimental data of teleoperation at the Advanced Teleoperation Laboratory of Jet Propulsion Laboratory (JPL). The data collected at JPL are based on actual human subjects engaged in the training of tool retrieval and storage processes for space teleoperation, where the traces of data provide the gradual improvements of human skills through training. There are total of 60 traces collected from 6 persons. Each person contributes about 10 traces. The actual position data collected after normalization are shown in Fig. 15. The augmented situation space is a six dimensional space where (x_k,y_k,z_k) represents the current situation space and $(x_{k+1},y_{k+1},z_{k+1})$ represents the next situation space. The path is generated based on forward mapping from $(x(k),y(k),z(k))$ to $(x(k+1),y(k+1),z(k+1))$, and inverse mapping from $(x(k+1),y(k+1),z(k+1))$ to $(x(k),y(k),z(k))$. Fig. 16 shows the final trajectory obtained. The final trajectory contains 38 situations, while the best trajectory from the input data contains 77 situations. The results obtained by our approach may not necessarily the same as the results from HMM model [11].

5. Conclusion

This paper presented a novel MRBF network and established a firm connection between the MRBF network and fuzzy rule based systems. More precisely, 1) A method of self-organizing a set of hyper-ellipsoidal local clusters of various size and shape is established as a means of self-organizing fuzzy rules. 2) A method of performing accurate many-to-many mapping based on cooperative interpolation of local clusters is presented as a means of defuzzification. 3) A method of searching for an optimal transition sequences based on dynamic path planning as a means of implementing an inference engine. The presented MRBF network is much more efficient and accurate in mapping representation than the conventional RBF networks, since the non-uniform size and shape of local clusters are better and easier to fit for an arbitrary mapping relation. The MRBF network is also more convenient for achieving a balanced trade-

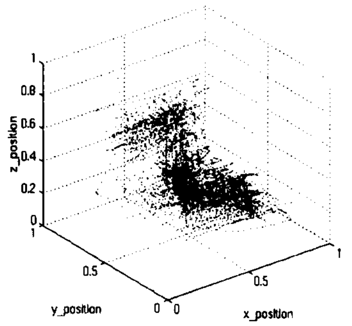

Figure 15: The position data collected for the tool retrieval and storage task at JPL. Total of 60 traces are collected.

off between accuracy and generalization. Neural networks including RBF networks need a sufficient number of training samples to achieve good learning and generalization, especially when their dimensions go up higher. It would be interesting to see what further advantages can be resulted if the proposed MRBF network is combined with expert rules such that the network can achieve a good generalization in spite of a limited number of available samples.

Reference

[1] Th. Fraichard and A. Scheuer, "Car-like robots and moving obstacles", IEEE International Conference on Robotics and Automation, 1994.

[2] J. Latombe, "Robot motion planning", Kluwer Academic Publishers, 1991.

[3] J. Laumond, P. Jacobs, M. Taix and R. Murray, "A motion planner for nonholonomic mobile robots", IEEE Transactions on Robotics and Automation, Vol 10, No. 5, October 1994.

[4] B. Kosko, "Fuzzy function approximation," International Joint Conference on Neural Networks, Baltimore, 1992.

[5] W. Miller, R. Sutton, and P. Werbos, "Neural networks for control," MIT Press, 1990.

[6] J. Yang, Y. Xu, and C. Chen, "Hidden Markov model approach to skill learning and its application to telerobotics," IEEE Transactions on Robotics and Automation, vol. 10, No. 5, Oct. 1884.

[7] P. Cardaliaguet and G. Euvrard, Approximation of a function and its derivative with a neural network, Neural Networks, 5 (1992) 207-220.

[8] K. Funahashi, On the approximate realization of continuous mappings by neural networks, Neural Networks, 2 (1989) 183-192).

[9] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks, 4(2) (1991) 251-257.

[10] K. Hornik and M. Stinchcombe and H. White, Multilayer feedforward networks are universal approximators, Neural Networks, 2 (1989) 359-366.

[11] K. S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Transactions on Neural Networks, 1(1) (1990) 4-27.

[12] R. Luo, M. Lin and R. Scherp, Dynamic multi-sensor data fusion system for intelligent robots, IEEE Journal of Robotics and Automation, RA 4(4), pp. 386-396, 1988.

[13] B. Kosko, Neural Networks and Fuzzy Systems: A dynamic systems approach to machine intelligence, Prentice-Hall, Inc., New Jersey, 1992.

[14] C. T. Lin and C.S.G. Lee, "Neural network based fuzzy logic control and decision system,", IEEE Transaction on

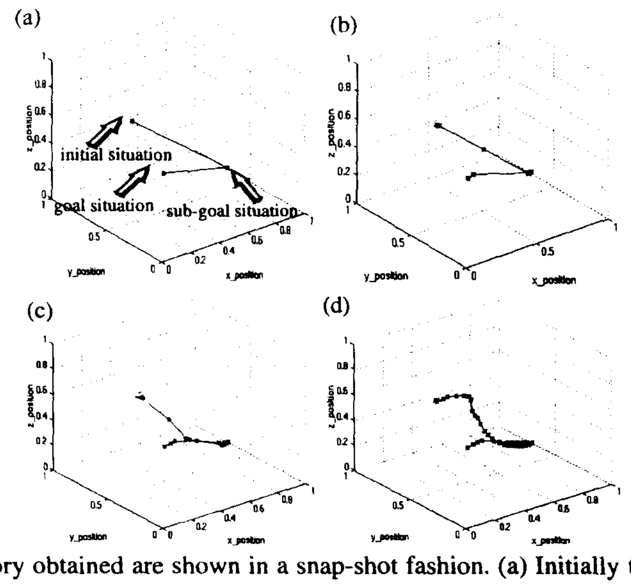

Figure 16: The trajectory obtained are shown in a snap-shot fashion. (a) Initially three situations are defined, initial, sub-goal and goal situations. (b) 8 situations are included. (c) 18 situations are included. (d) The final trajectory obtained contains 38 situations.

Computers, Vo. C-40, No. 12, Dec. 1991.

[15] J.M. Keller and H. Tahani, "Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks," International Journal of Approximate Reasoning, Vo. 6, pp. 221-240, 1992.

[16] J. J. Buckley, "Theory of fuzzy controller: an introduction," Fuzzy Sets and Systems, 51, pp. 249-258, 1992.

[17] M. Sugeno and G. T. Kang, "Structure identification fuzzy model," Fuzzy Sets and Systems, 28, pp.15-33, 1988

[18] C. C. Lee, "Fuzzy logic in control systems: fuzzy logic controller, "Part II, IEEE Transaction on Systems, Man, and Cybernetics, 20, pp. 419-435, 1990.

[19] L. A. Zadeh, "Outline of a new approach to the analysis of complex system and decision process, " IEEE Transaction on System, Man, and Cybernetics, Vol. SMC-1, pp. 28-44, 1973.

[20] R. Katayama, Y. Kajitani, K. Kuwata, and Y. Nishida, "Self generating radial basis function as neuro-fuzzy model and its application to nonlinear prediction of chaotic time series," Proceedings of the 2nd IEEE International Conference on Fuzzy Systems, San Francisco, pp. 407-414, 1993.

[21] T. Yamakawa and M. Furukawa, "A design Algorithm of membership functions for a fuzzy neuron using example-based learning," Proceeding of IEEE International Conference on Fuzzy Systems, San Diego, CA, pp. 75-82, March, 1992.

[22] S. Lee and R. M. Kil, ``Multilayer Feedforward Potential Function Network,'' Artificial Neural Networks: Concepts and Theory, P. Mehra and B. W. Wah, editors, IEEE Computer Society Press, 1992, pp. 83-93.

[23] S. Lee, ``Supervised Learning with Gaussian Potentials,' Neural Networks for Signal Processing, B. Kosko, editor, Prentice Hall, Englewood, Cliffs, NJ., Oct. 1991, pp. 189-227.

[24] S. Lee and R. M. Kil, ``A Gaussian Potential Function Network with Hierarchically Self-Organizing Learning,'' Neural Network, the Official Journal of the International Neural Network Society, Vol. 4, pp. 207-224, 1991.

[25] S. Lee and J.M. Lee, ``Nonlinear System Control Based on Multi-Resolution Radial-Basis Competitive and Cooperative Networks,'' Journal of Neurocomputing,, Elsevier Science, Vol.9 (1995).

[26] S. Lee and S. Shimoji, ``Self-Organization of Probabilistic Network with Gaussian Mixture Model,'' Proceedings of the 1994 IEEE International Conference on Neural Network, June-July 1994, Orlando, Florida.

[27] S. Lee and S. Shimoji, ``RCCN: Radial Basis Competitive and Cooperative Network for Bidirectional Many-to-Many Mapping,'' Proceedings of the 1992, International Joint Conference on Neural Networks,Nov. 1992, Beijing, China.

[28] S. Lee and S. Shimoji, ``Radial Basis Competitive and Cooperative Network with Hierarchical Network Self-Organization,'' Proceedings of the 1992 IEEE International Conference on Tools with Artificial Intelligence, Nov. 1992, Arlington, VA.

[29] S. Lee and J. Chen, `` Robot Skill Learning from Observations,'' Proceedings of the 1994 IEEE International Conference on Robotics and Automation, May 1994, San Diego, California, pp. 3245-3250.