Fuzzy Membership Function Based Neural Networks with Applications to the Visual Servoing of Robot Manipulators

Il Hong Suh, Member, IEEE and Tae Won Kim, Student Member, IEEE

Abstract—It is shown that there exists a nonlinear mapping which transforms image features and their changes to the desired camera motion without measuring of the relative distance between the camera and the object. This nonlinear mapping can eliminate several difficulties occurring in computing the inverse of the feature Jacobian as in the usual feature-based visual feedback control methods. Instead of analytically deriving the closed form of this mapping, a fuzzy membership function (FMF) based neural network incorporating a fuzzy-neural interpolating network is proposed to approximate the nonlinear mapping, where the structure of the FMF network is similar to that of radial basis function neural network which is known to be very effective in the function approximation. Several FMF networks are trained to be capable of tracking a moving object in the whole workspace along the line of sight. For an effective implementation of the proposed FMF network, an image feature selection process is investigated, and the required fuzzy membership functions are designed. Finally, several numerical examples are presented to show the validity of the proposed visual servoing method.

I. INTRODUCTION

RECENTLY, the visual feedback control has received much attention particularly in the intelligent robotics field by virtue of its power of allowing a robot to manipulate and track a randomly moving object without any previous knowledge of the object’s location or motion [1]. Applications of visual feedback systems include the seam tracking [2], the precision part placement [3], the conveyor tracking [4], and the control of space telerobots [5].

The actual image features have been widely employed as feedback signals in visual robot systems. In the image feature-based visual feedback structure [6] is usually required the computation of the inverse of the feature Jacobian, which is differential relationship between a feature space and a camera motion space, for computing the desired camera motion based on the image feature changes of objects. However, using the inverse of the feature Jacobian cannot accommodate rather large feature variations due to its corresponding large motion errors. Thus, other auxiliary techniques may be necessary to compensate for such motion errors [7]-[9].

Feddema and Mitchell [7] proposed a feature-based dynamic look-and-move control structure allowing the visual feedback to have variable sampling and delay times by incorporating the generation of feature-based trajectories as well as the feature Jacobian. However, the feature Jacobian must be manually generated before a task begins. In-depth knowledge of the robot kinematics and the camera modelling is not a trivial task. Hence, in [8] and [9], the model reference adaptive controller (MRAC) has been employed to minimize coupling effects in transforming the coordinate of feature space to that of the joint space. But computational requirements for the real-time parameter identification, and the sensitivity to numerical precision, sensor noise, and choice of model might make the approach infeasible as the number of system variables increases.

In spite of the use of such auxiliary techniques, there are some additional drawbacks in using the feature Jacobian approach [7]-[9]. First, feature Jacobians in [7]-[9] could be the inherent minor error source when the usual pin-hole camera model is assumed. Second, the feature Jacobian approach requires estimation of the distance between the object and the camera, which is usually solved by employing depth measuring techniques such as laser range finders, binocular stereo [10], active monocular stereo [11], or computed from the objects’ CAD model data [1],[7]. This makes the feature Jacobian approach too complicated to implement. And even when equipping the controller with a distance measurement device, the output tracking performance may heavily depend upon the choice of features [1], because the feature Jacobian is very sensitive to feature variations. Finally, in the worst case, the feature Jacobian might be singular, which drives the system uncontrollable.

On the other hand, when we consider that living animals including human beings just see and pick up an object without precise numerical position information, the visual servoing seems to be an innate ability of living animals. Such a skill often improves by repetitive learning while growing up. Insinuated by the above observation, some researchers have attempted to develop not only methods for the robot motion control utilizing the image features, but also schemes enabling a robot to elaborate the motion for itself by repetitive trials [4], [12], [13].

Among them, Miller [4], [12] proposed a neural network based learning control system for the visual servoing, where the CMAC (Cerebeller Model Arithmetic Computer) memory was employed for the learning. In this control system, a priori knowledge of neither the robot kinematics nor the object speed...
or orientation relative to the robot was assumed. However, in the implementation of such supervised learning controllers, network training exercises should be performed to approximate the desired transformation not only over a particular region but over the entire operating space of the system. This is necessary to ensure the general application of the robot to complex nonrepetitive tasks as well as simple repetitive ones, because the robot controller is trained only over a particular region for a specific task, it cannot respond to unexpected changes of the environment and/or tasks. Hashimoto et al. [13] proposed a self-organizing visual servoing system based on two back-propagation neural networks which learn the feature Jacobian regardless of the joint type and geometric dimensions. However, as in Miller's networks [4], [12], this network has to be trained over the whole workspace. In addition, the convergence of weights of the back-propagation neural networks to global minimum is not generally guaranteed. Thus, it seems to be very difficult to practically apply this method to real tasks.

On the other hand, a completely different approach for the feature-based visual servoing was proposed by authors [15], where was proposed a new type of Jacobian, in which element was not a function of the relative distance between the camera and the object but a function of only the image features. Furthermore, to avoid the complexity in deriving the proposed Jacobian, a fuzzy controller with a supervised learning capability was suggested to provide the robot with the path following capability of linear motion. The employment of such a new Jacobian can be expected to eliminate difficulties encountered when the traditional feature Jacobian was used; however, due to the simple gradient method used in the learning algorithm, a satisfactory performance could not be guaranteed.

In this paper, we will first show that there exist a nonlinear mapping which transforms features and their variations to a desired camera motion without information on the relative distance between the camera and the object, and the nonlinear mapping can eliminate several difficulties occurring in computing the inverse of the feature Jacobian as in the usual feature-based visual feedback controls [7], [10]. Instead of analytically deriving the closed form of such a nonlinear mapping, a fuzzy membership function (FMF) based neural network is proposed to approximate the nonlinear mapping. The structure of the proposed networks is similar to that of the radial basis function (RBF) network [16] which is known to be very effective in the function approximation.

The main concept of our fuzzy-neural network hinted by the human's behavioral learning process is to set up the initial knowledge of a function or a mapping by fuzzy rules and then to iteratively improve the degree of approximation by the learning capability of the neural network. Among examples of such a human learning process in visual servoing, "driving a car" could be a good example. When we drive a car, if an object such as a road sign or a traffic light looks small, we feel tempted to drive the car at a high speed to reach the object as we expect that the distance between the object and the car is far. But, if the object looks large, we drive the car slowly since we think that the object is close to the car. Such intuitive rules are refined through repetitive driving experiences to acquire better driving skills, and the refined rules hold in any places where the driver has never been before, even if the rules have been updated only within a limited small training region.

In this paper, we will show how the human visual servoing principle is employed to generate desired input-output training data. If the distance between the camera and the object is larger than a maximum displacement per sampling time, the camera approaches the target at its maximum speed. Otherwise, the camera moves toward the target at a lower speed. The proposed FMF network must be trained so as to be capable of tracking a moving object in the whole workspace along the line of sight. A small training region is chosen within the whole workspace as a visible region determined by considering the view angle of the camera. Next, the visible region is segmented into a number of subspaces. For each subspace, an FMF network is assigned to learn the camera motion along a representative linear path belonging to the subspace, while a target position is fixed until all the FMF networks are completely trained. A fuzzy-neural interpolating network is then augmented to cover the whole training region. For an effective implementation of the proposed FMF networks, an image feature selection process is investigated, and the required fuzzy membership functions are designed. Finally, several numerical examples are presented to show the validity of our proposed visual servoing method.

Note that Wang and Mendal [32] recently proposed the Fuzzy Basis Function (FBF) for function approximation, which is similar to our FMF network. However, in their approach there were no learning trials, which implies that the FBF was not employed as a neural network. Specifically, two arbitrary sets of initial FBF were first constructed from input-output data pairs and informal IF-THEN rules. And then significant portions of the initial FBF's were selected based on their error reduction ratio. Thus the performance of their algorithm critically depends upon the initial choice of basis functions, their membership functions, and the number of basis functions which should be usually selected after many trial and errors for satisfactory performance. Compared to their approach, in our FMF network, the membership functions of the THEN part in the fuzzy IF-THEN rules are adjusted by the gradient descent method as is usual in the RBF and the BPNN. Hence there is no need to determine either the initial basis functions or the number of basis functions, owing to the learning capability of our FMF network. In particular, when a large number of fuzzy variables are needed for the function approximation, the proposed fuzzy-neural interpolating network can be employed to produce simple fuzzy rules, but there is no such a scheme in [32]. In these viewpoints, our approach can be considered as a better solution of the fuzzy-neural fusion compared to the approach in [32].

II. A NONLINEAR TRANSFORMATION FROM A DIFFERENTIAL IMAGE SPACE INTO A DIFFERENTIAL CARTESIAN SPACE

Let $X$ be an $m$-dimensional relative position vector of an object with respect to a camera frame $C$, and let $F$ be an
\( n \)-dimensional feature vector of the object image. Then, the forward kinematic relationship between \( \dot{X} \) and \( F \) can be represented as

\[
F = I(\dot{X}),
\]

where \( I(\bullet) \) is a mapping from \( \dot{X} \in \mathbb{R}^n \) to \( F \in \mathbb{R}^n \). If the mapping from \( \dot{X} \) to \( F \) is one-to-one, then the inverse kinematic relationship may be represented as

\[
\dot{X} = \frac{d}{dt} \left[ F \right] = \frac{d}{dt} \left[ G(F, \delta F) \right] = \frac{d}{dt} \left[ I^{-1}(F) \right] = \frac{d}{dt} \left[ \frac{\partial I^{-1}(F)}{\partial F} \delta F \right].
\]

where \( G(F, \delta F) \) is defined as

\[
G(F, \delta F) = \lim_{N \to \infty} \sum_{k=1}^{N} \left( \frac{\partial I^{-1}(F)}{\partial F} \delta F \right)^k.
\]

When \( \delta X = G(F, \delta F) \) can be employed, the difficulties encountered by the Jacobian approaches using \( (4) \) can be eliminated due to the following reasons: i) \( G(F, \delta F) \) does not require the geometric optics model between the camera and the object, ii) the inverse mapping is not necessary, and iii) since \( G(F, \delta F) \) is an \( n \)-dimensional vector function of only \( F \) and \( \delta F \), \( G(F, \delta F) \) is not required to be remodeled whenever the object is changed. In spite of these desirable characteristics, it is usually difficult to analytically obtain \( G(F, \delta F) \). Thus, we will propose a fuzzy membership function based network to approximate the nonlinear mapping \( G(F, \delta F) \) instead of trying to analytically derive the closed form of \( G(F, \delta F) \).

In [15], we proposed a fuzzy controller with a supervised learning capability to approximate \( G(F, \delta F) \), where \( N = 1 \) in (6) and the centers of the fuzzy membership functions for the linguistic values were trained iteratively using a simple gradient method. But the gradient method was too simple to obtain a satisfactory approximation.

III. NONLINEAR FUNCTION APPROXIMATION BY THE FUZZY MEMBERSHIP FUNCTION BASED NEURAL NETWORKS

It is well known that the nonlinear function approximation can be often solved by finding a set of coefficients for a finite number of fixed nonlinear basis functions [17]. Specifically, to approximate a desired scalar function \( f \) over \( p \)-dimensional input, it is necessary to choose proper basis functions \( \Phi \) and their coefficients \( G_i \) such that

\[
\left\| f(y_1, y_2, \ldots, y_p) - \sum_{i=1}^{L} G_i \Phi_i (y_1, y_2, \ldots, y_p) \right\| < \epsilon
\]

is minimized, where \( \left\| \cdot \right\| \) implies a function norm and \( L \) is the number of basis functions to be employed. Three major techniques have been suggested to choose the basis functions: i) a large fixed basis can be chosen which is used to approximate any desired function \( f \); ii) a smaller adaptive basis which depends on the parameters that are varied to obtain an optimal approximation can be chosen; or iii) new basis functions can be added as new data points arrive. The back propagation neural networks (BPNN) in [18]–[21] can be regarded as an example of the second technique and the RBF [16], [28] can be designed for any of the three techniques. An obvious disadvantage in using the BPNN for the function approximation is that they are highly nonlinear in parameters. Learning must be based on nonlinear optimization techniques, and the parameter estimate may become trapped at a local minimum of the chosen performance index during the learning process when the gradient descent algorithm is utilized. In contrast to the BPNN, the RBF network [16], [28] can be considered as a special two-layer network which is linear in the parameters by fixing all RBF centers and nonlinearities in the hidden layer. Thus the hidden layer performs a fixed
nonlinear transformation with no adjustable parameters and it maps the input space onto a new space. The output layer then implements a linear combiner on this new space and the only adjustable parameters are the weights of this linear combiner. These parameters can therefore be determined using the linear least squares (LS) method, which is an important advantage of this approach. Because of a strong connection between the RBF and the neural network, it is reasonable to believe that the RBF network can offer approximation capabilities similar to those of the two-layer neural network, provided that the hidden layer of the RBF network is appropriately fixed. This heuristic belief is strongly supported by the theoretical results on the RBF method as a multidimensional interpolation technique [22].

However, the performance of the RBF network critically depends upon the chosen centers of the basis functions. In practice the centers are often chosen to be a subset of the data. Although researchers are well aware that fixed centers should suitably sample the input domain, most published results simply assume that centers are arbitrarily selected from data points. Usually, resulting RBF networks either perform poorly or have a large size. Furthermore, numerical ill-conditioning frequently occurs owing to the near linear dependency caused by, for example, some centers being too close [23].

To overcome such difficulties in choosing centers of the RBF, fuzzy membership functions are here employed, since centers of fuzzy membership functions can be relatively well determined by considering empirical rules describing input-output relations of an unknown function.

Now, fuzzy relations are first shown to be represented as a linear combination of their fuzzy membership functions, where weightings are centers of singleton membership functions for output fuzzy variables. And then a learning rule is given by applying a gradient descent method. In addition, a fuzzy-neural interpolating network is proposed to overcome difficulties in determining fuzzy rules, especially for the case where a relatively large number of input variables are required for the interpolation, RBF, or have a large size. Furthermore, numerical ill-conditioning frequently occurs owing to the near linear dependency caused by, for example, some centers being too close [23].

Consider the following q fuzzy relations:

\[ R^1: \text{If } y_1 = A_{11}, y_2 = A_{12}, \ldots, y_p = A_{1p}, \text{ then } u = B_1, \]

\[ R^2: \text{If } y_1 = A_{21}, y_2 = A_{22}, \ldots, y_p = A_{2p}, \text{ then } u = B_2, \]

\[ \vdots \]

\[ R^q: \text{If } y_1 = A_{q1}, y_2 = A_{q2}, \ldots, y_p = A_{qp}, \text{ then } u = B_p. \]

Here, \( y_j \), for \( j = 1, 2, \ldots, p \), is the \( j \)-th input variable and \( u \) is the output fuzzy variable fuzzified with a singleton membership function. \( A_{ij} \) and \( B_i \), for \( i = 1, 2, \ldots, q \) and \( j = 1, 2, \ldots, p \) are input and output linguistic (fuzzy-set) values, respectively. Let \( \mu_{A_{ij}}^A(y_j) \) and \( \mu_{B_i}^B(u) \) be the membership functions for \( A_{ij} \) and \( B_i \), respectively. Then the output fuzzy set \( B \) for an input \( (y_1^0, y_2^0, \ldots, y_p^0) \) can be computed by the typical sup-min [27] or sup-product fuzzy inference method [27] as follows;

\[ \text{(Sup-min inference method)} \]

\[ \mu_{B}^B(u) = \sup\{\min\{\mu_{A_{11}}^A(y_1^0), \ldots, \mu_{A_{1p}}^A(y_p^0), \mu_{B_1}^B(u)\}\} \]

\[ \text{(8)} \]

\[ \text{(Sup-product inference method)} \]

\[ \mu_{B}^B(u) = \sup\{\mu_{B_1}^B(u) \min\{\mu_{A_{11}}^A(y_1^0), \ldots, \mu_{A_{1p}}^A(y_p^0)\}\} \]

\[ \text{(9)} \]

It is remarked that the supremum combination techniques in (8) and (9) tend to produce a uniform distribution for \( B \) as the number of combined fuzzy sets increases [24]. A uniform distribution always has the same mode and centroid. Thus as the number of fuzzy relations increases, system sensitivity decreases. In [25], the additive combination technique was shown to be more reasonable than the supremum combination technique, since the additive combination technique tends to invoke the fuzzy version of the central limit theorem. In other words, the added fuzzy waveforms pile up to approximate a symmetric unimodal, or bell-shaped membership function. Different fuzzy waveforms produce similarly shaped output distributions but centered about different places on the real line. When applying the additive combination technique for the inference, \( \mu_{B}^B(u) \) can be obtained by either

\[ \mu_{B}^B(u) = \sum_{i=1}^{q} \min\{\mu_{A_{i1}}^A(y_1^0), \ldots, \mu_{A_{ip}}^A(y_p^0), \mu_{B_i}^B(u)\}, \]

\[ \text{(10)} \]

or

\[ \mu_{B}^B(u) = \sum_{i=1}^{q} \mu_{B_i}^B(u) \min\{\mu_{A_{i1}}^A(y_1^0), \ldots, \mu_{A_{ip}}^A(y_p^0)\}. \]

\[ \text{(11)} \]

Since \( \mu_{B}^B(u) \) in (8)–(11) should be defuzzified to produce a single scalar output value, a popular defuzzification technique such as the centroid of mass would be here employed. If we let \( \mu_{B_i}^B(u) \) be a normal singleton located at \( u = \lambda_i \) for each \( i \), and apply the centroidal defuzzification technique to \( \mu_{B_i}^B(u) \) in (8)–(11), then \( \mu_{B}^B(u) \) becomes \( \mu_{B_i}^B(\lambda_i) \). Thus, regardless of types of the inference, the scalar output \( u \) can be obtained by

\[ u = \sum_{i=1}^{q} \lambda_i \frac{\Phi_i(y_1^0, y_2^0, \ldots, y_p^0)}{\sum_{k=1}^{q} \Phi_k(y_1^0, y_2^0, \ldots, y_p^0)} = \sum_{i=1}^{q} \lambda_i \Phi_i(y_0^0), \]

\[ \text{(12)} \]

where \( \Phi_i(y_1^0, y_2^0, \ldots, y_p^0) \) and \( \Phi_i(y_0^0) \) are defined as

\[ \Phi_i(y_1^0, y_2^0, \ldots, y_p^0) \triangleq \min\{\mu_{A_{ij}}^A(y_j^0)\} \quad j = 1, 2, \ldots, p, \]

\[ \text{(13a)} \]

and

\[ y_0^0 \triangleq (y_1^0, y_2^0, \ldots, y_p^0), \]

\[ \text{(13b)} \]

and

\[ \Phi_i(y_0^0) \triangleq \frac{\Phi_i(y_1^0, y_2^0, \ldots, y_p^0)}{\sum_{k=1}^{q} \Phi_k(y_1^0, y_2^0, \ldots, y_p^0)}. \]

\[ \text{(13c)} \]
SUH AND KIM: FUZZY MEMBERSHIP FUNCTION

The schematic diagram of an FMF based neural network.

Fig. 1. The schematic diagram of an FMF based neural network.

The radial basis function network to implement a mapping $f_{c}: R^{p} \rightarrow R$ can be written as

$$ f_{c}(y) = \alpha_{0} + \sum_{i=1}^{q} \alpha_{i} \Psi(||y - c_{i}||), $$

(14)

where $y \in R^{p}$ is an input vector, $\Psi(\bullet)$ is a given function from $R^{q}$ to $R$, $|| \bullet ||$ denote the Euclidean norm, $\alpha_{i}, 0 \leq i \leq n$, are the weights or parameters, $c_{i} \in R^{p}, 1 \leq i \leq n$, are known as RBF centers and $q$ is the number of centers [23]. Then by letting $\alpha_{0} = 0$, $\alpha_{i} = \lambda_{i}$, and

$$ \Psi(||y - c_{i}||) = \Phi_{i}(y^{0}), $$

(15)

Equation (12) can play the role of approximating a function as the RBF network can do. Now, we will call (12) as “Fuzzy Membership Function (FMF) based neural network,” where $\lambda_{i}$’s are the neural weights to be trained by using the linear least square method.

In applying (12) to a function approximation, as in [16], we put a nonlinear scalar function $g: R \rightarrow R$ given by

$$ g(u) = (1 - \exp(-\beta_{1} u))/(1 + \exp(-\beta_{1} u)) $$

(16)

at the output node together with the scaling factor $K$ to effectively account for the maximum magnitude of the function output. In (16), $\beta_{1}$ is a constant implying the slope of the output node function. It is remarked that the choice of the output node function depends on how we choose to represent the output data. For example, if we want the output units to be binary, we use a sigmoid output node function, since the sigmoid is output-limiting and quasi-binary but is also differentiable. In other cases, either a linear or a linear output node function of the type in (16) is appropriate [26]. For the function approximation, let $\tilde{f}(y)$ be a scalar function to be approximated, and let $u(y)$ be an approximation of $f(y)$. Then $u(y)$ can be represented as

$$ u(y) = K g \left( \sum_{i=1}^{q} \lambda_{i} \Phi_{i}(y) \right) $$

(17)

if the FMF network is utilized for the function approximation.

Let us define the error function which is proportional to the square of the difference between the actual and the desired output given by

$$ E = \frac{1}{2} (f(y) - u(y))^{2}. $$

(18)

From (17) and (18), the derivative of the error function $E$ with respect to the weights of the $i$-th FMF network, $\lambda_{i}$, can be obtained as

$$ \frac{\partial E}{\partial \lambda_{i}} = - \frac{\partial f(y)}{\partial u(y)} \frac{\partial f(y)}{\partial \lambda_{i}} = \frac{g'}{g} \sum_{j=1}^{q} \lambda_{j} \Phi_{j}(y) \Phi_{i}(y), $$

(19)

where $g'(\bullet)$ is the derivative of $g(\bullet)$ given by

$$ g'(u) = \frac{\beta_{1} e^{-\beta_{1} u}}{1 + e^{-\beta_{1} u}} $$

(20)

By plugging (20) into (19), we obtain

$$ \frac{\partial E}{\partial \lambda_{i}} = - \beta_{1} K \left[ f(y) - u(y) \right] \left\{ 1 - g^{2} \left( \sum_{j=1}^{q} \lambda_{j} \Phi_{j}(y) \right) \right\} \Phi_{i}(y). $$

(21)

To decrease $E$ with respect to $\lambda_{i}$, the weight changes, $\Delta \lambda_{i}$, can be chosen to be proportional to $\partial E/\partial \lambda_{i}$;

$$ \Delta \lambda_{i} = \eta \beta_{1} K \left[ f(y) - u(y) \right] \left\{ 1 - g^{2} \left( \sum_{j=1}^{q} \lambda_{j} \Phi_{j}(y) \right) \right\} \Phi_{i}(y) $$

(22)

where $\eta$ is the learning-rate parameter. Thus, the learning rule for adapting weight can be given as

$$ \lambda_{i}^{t+1} = \lambda_{i}^{t} + \eta \beta_{1} K \left[ f(y) - u(y) \right] \left\{ 1 - g^{2} \left( \sum_{j=1}^{q} \lambda_{j} \Phi_{j}(y) \right) \right\} \Phi_{i}(y) $$

(23)

where $t$ is an integer implying the number of learning trials. The schematic diagram of our FMF network with $p$ inputs and a scalar output is depicted in Fig. 1.

By comparing the FMF network with the RBF network, it may be worthwhile to observe the following. First, the structure of the FMF network is completely the same as that of the RBF network, where $\Phi_{i}(y)$ plays the role of the basis function. Second, the number and the centers of the basis functions are determined by the number of fuzzy relations and linguistic values of the fuzzy input variables in the ‘IF’ part of the fuzzy relations, respectively. Third, weighting values $\lambda_{i}$’s to be updated by (23) correspond to locations of singleton fuzzy membership functions for the ‘THEN’ part of fuzzy relations. In these respects, the FMF network may be considered as an effective fusion of the RBF neural network.
assigning a constant value $y^*_{pi}$ to a variable $y_p$ of the function $f(y_1, y_2, \ldots, y_p)$. Let $H_i(y_1, y_2, \ldots, y_{pi-1})$, for $i = 1, 2, \ldots, M$, be approximated as $\hat{H}(y_1, y_2, \ldots, y_{pi-1})$ by utilizing $M$ FMF networks. Then for a given input $(y_1, y_2, \ldots, y_p)$, if $\hat{y}_p$ is different from $y^*_{pi}$ for all $i$, the output value of $f(y_1, y_2, \ldots, y_p)$ needs to be estimated by interpolating $\hat{H}(y_1, y_2, \ldots, y_{pi-1})$, for $i = 1, 2, \ldots, M$. For this, we employ fuzzy rules which inform how much $\hat{y}_p$ is similar to each $y^*_{pi}$, $i = 1, 2, \ldots, M$. To be specific, let $S_i$ be the fuzzy similarity between $\hat{y}_p$ and $y^*_{pi}$. Then, an example of such necessary fuzzy rules can be written as

If $|\hat{y}_p - y^*_{pi}|$ is small, then $S_i$ is large.

or

If $|\hat{y}_p - y^*_{pi}|$ is medium, then $S_i$ is medium.  \hspace{1cm} (24)

or

If $|\hat{y}_p - y^*_{pi}|$ is large, then $S_i$ is small.

Let $n_a$ be the number of fuzzy rules for $S_i$, and let $\pi_{ij}: R \rightarrow [0, 1]$ for $i = 1, 2, \ldots, M$, and $j = 1, 2, \ldots, n_a$ be the membership function for the $j$th linguistic value of $|\hat{y}_p - y^*_{pi}|$. Also let $\gamma_{ij}$ be the location of singleton membership function for the $j$th linguistic value of $S_i$. In the example given by (24), $n_a$ is 3. Figs. 2 and 3 show examples of membership functions for $\pi_{ij}$ and $\gamma_{ij}$, respectively. Then $S_i$ can be represented as

$$S_i = \sum_{j=1}^{n_a} \gamma_{ij} \pi_{ij}(\hat{y}_p).$$  \hspace{1cm} (25)

Now, for the fuzzy-neural interpolation, the sigmoid function given by

$$\hat{g}(x) \triangleq \frac{1}{1 + \exp(\beta x)}$$  \hspace{1cm} (26)

is used as an output node function of each $S_i$. Note that the larger $\hat{g}(S_i)$ is, the more $\hat{H}(y_1, y_2, \ldots, y_{pi-1})$ are contributed to finding values of $f(y)$. Then we can know that $f(y)$ can be found by

$$f(y) = Kg \left[ \sum_{i=1}^{M} \hat{g}(S_i) \hat{H}(y_1, y_2, \ldots, y_{pi-1}) \right].$$  \hspace{1cm} (27)

where $g(\bullet)$ and $\hat{g}(\bullet)$, respectively, are the scalar functions defined as in (16) and (26). The schematic diagram of the FMF networks incorporating the fuzzy-neural interpolating network is depicted in Fig. 4.

Since only $M$ representative functions are available, at most $M$ function values can be approximated. Thus, to cover the whole input space, we need to divide the input space by $M$ subspaces. By choosing the performance index $J_i$ for the $i$th subspace $B_i$, as

$$J_i \triangleq \sum_{y \in B_i} (u(y) - f(y))^2,$$  \hspace{1cm} (28)

and by applying the gradient descent method to minimize $J_i$, our updating rule for the weight $\gamma_{ij}$ in (25) can be obtained. Specifically, since $\frac{\partial}{\partial u} (g(u) - f(u))^2 = 2(g(u) - f(u)) g'(u)$, the derivative
The performance index $J_i$ with respect to the weight $\gamma_{ij}$ can be obtained as follows:

$$
\frac{\partial J_i}{\partial \gamma_{ij}} = 2 \sum_{y \in B_i} (u(y) - f(y))' (u(y) - f(y))
$$

$$
= 2 \sum_{y \in B_i} (u(y) - f(y))' \times \left\{ \kappa(y) - K \hat{g} \left[ \sum_{k=1}^{M} \hat{g} \left( \sum_{i=1}^{m} \gamma_{ik} \pi_k(y_p) \right) \times H_k(y_i, y_2, \ldots, y_{p-1}) \right] \right\}'
$$

$$
= 2 \sum_{y \in B_i} (u(y) - f(y))' \times \left\{ -K \hat{g} \left[ \sum_{k=1}^{M} \hat{g} \left( \sum_{i=1}^{m} \gamma_{ik} \pi_k(y_p) \right) \times H_k(y_i, y_2, \ldots, y_{p-1}) \right] \right\}
$$

where $\hat{g}$ is the learning-rate parameter given as a positive constant not greater than or equal to unity.

To demonstrate the capability of the function approximation of our FMF network incorporating the fuzzy-neural interpolating network, a simulation is performed with a function known as the Mexican hat, *sombrero*, given by

$$
f(x, y) = \begin{cases} 
\frac{40 \sin \left( \pi \sqrt{x^2 + y^2} / 35 \right)}{\pi \sqrt{x^2 + y^2} / 35}, & \text{for } x \neq 0, y \neq 0, \\
\frac{40 \pi}{\sqrt{x^2 + y^2} / 35}, & \text{for } x = y = 0,
\end{cases}
$$

which is depicted in Fig. 5. The input and the output universes of discourse are given as $x \in [-120, 120]$, $y \in [-120, 120]$. 

The learning rule for adapting weights $\gamma_{ij}$ of the fuzzy-neural interpolating network can be given as

$$
\gamma_{ij}^{t+1} = \gamma_{ij}^t + K \beta_1 \beta_2 \pi_{ij}(y_{ij}) \hat{H}_i(y_1, y_2, \ldots, y_{p-1}) \times (f(x, y) - \hat{g}(\gamma_{ij}^t \pi_{ij}(y_{ij}))) \times \pi_{ij}(y_{ij}) H_i(y_1, y_2, \ldots, y_{p-1})
$$

$$
\times \hat{g}(\gamma_{ij}^t \pi_{ij}(y_{ij}))(1 - \hat{g}(\gamma_{ij}^t \pi_{ij}(y_{ij}))) \times \sum_{y \in B_i} (u(y) - f(y))
$$
TABLE I
SAMPLED TRAINING DATA FOR APPROXIMATING THE FUNCTION IN (31)

<table>
<thead>
<tr>
<th>i</th>
<th>x</th>
<th>y</th>
<th>y^*</th>
<th>y_1</th>
<th>y_2</th>
<th>y_3</th>
<th>y_4</th>
<th>y_5</th>
<th>y_6</th>
<th>y_7</th>
<th>y_8</th>
<th>y_9</th>
<th>y_10</th>
<th>y_11</th>
<th>y_12</th>
<th>y_13</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-120</td>
<td>-1.714</td>
<td>-1.714</td>
<td>-1.714</td>
<td>-1.714</td>
<td>-1.714</td>
<td>-1.714</td>
<td>-1.714</td>
<td>-1.714</td>
<td>-1.714</td>
<td>-1.714</td>
<td>-1.714</td>
<td>-1.714</td>
<td>-1.714</td>
<td>-1.714</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-100</td>
<td>-0.531</td>
<td>-0.531</td>
<td>-0.531</td>
<td>-0.531</td>
<td>-0.531</td>
<td>-0.531</td>
<td>-0.531</td>
<td>-0.531</td>
<td>-0.531</td>
<td>-0.531</td>
<td>-0.531</td>
<td>-0.531</td>
<td>-0.531</td>
<td>-0.531</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-80</td>
<td>0.533</td>
<td>0.533</td>
<td>0.533</td>
<td>0.533</td>
<td>0.533</td>
<td>0.533</td>
<td>0.533</td>
<td>0.533</td>
<td>0.533</td>
<td>0.533</td>
<td>0.533</td>
<td>0.533</td>
<td>0.533</td>
<td>0.533</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-60</td>
<td>1.714</td>
<td>1.714</td>
<td>1.714</td>
<td>1.714</td>
<td>1.714</td>
<td>1.714</td>
<td>1.714</td>
<td>1.714</td>
<td>1.714</td>
<td>1.714</td>
<td>1.714</td>
<td>1.714</td>
<td>1.714</td>
<td>1.714</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-40</td>
<td>2.132</td>
<td>2.132</td>
<td>2.132</td>
<td>2.132</td>
<td>2.132</td>
<td>2.132</td>
<td>2.132</td>
<td>2.132</td>
<td>2.132</td>
<td>2.132</td>
<td>2.132</td>
<td>2.132</td>
<td>2.132</td>
<td>2.132</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-20</td>
<td>2.714</td>
<td>2.714</td>
<td>2.714</td>
<td>2.714</td>
<td>2.714</td>
<td>2.714</td>
<td>2.714</td>
<td>2.714</td>
<td>2.714</td>
<td>2.714</td>
<td>2.714</td>
<td>2.714</td>
<td>2.714</td>
<td>2.714</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>3.531</td>
<td>3.531</td>
<td>3.531</td>
<td>3.531</td>
<td>3.531</td>
<td>3.531</td>
<td>3.531</td>
<td>3.531</td>
<td>3.531</td>
<td>3.531</td>
<td>3.531</td>
<td>3.531</td>
<td>3.531</td>
<td>3.531</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>4.531</td>
<td>4.531</td>
<td>4.531</td>
<td>4.531</td>
<td>4.531</td>
<td>4.531</td>
<td>4.531</td>
<td>4.531</td>
<td>4.531</td>
<td>4.531</td>
<td>4.531</td>
<td>4.531</td>
<td>4.531</td>
<td>4.531</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>5.531</td>
<td>5.531</td>
<td>5.531</td>
<td>5.531</td>
<td>5.531</td>
<td>5.531</td>
<td>5.531</td>
<td>5.531</td>
<td>5.531</td>
<td>5.531</td>
<td>5.531</td>
<td>5.531</td>
<td>5.531</td>
<td>5.531</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>40</td>
<td>7.531</td>
<td>7.531</td>
<td>7.531</td>
<td>7.531</td>
<td>7.531</td>
<td>7.531</td>
<td>7.531</td>
<td>7.531</td>
<td>7.531</td>
<td>7.531</td>
<td>7.531</td>
<td>7.531</td>
<td>7.531</td>
<td>7.531</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>50</td>
<td>8.531</td>
<td>8.531</td>
<td>8.531</td>
<td>8.531</td>
<td>8.531</td>
<td>8.531</td>
<td>8.531</td>
<td>8.531</td>
<td>8.531</td>
<td>8.531</td>
<td>8.531</td>
<td>8.531</td>
<td>8.531</td>
<td>8.531</td>
<td></td>
</tr>
</tbody>
</table>

TABLE I
SAMPLED TRAINING DATA FOR APPROXIMATING THE FUNCTION IN (31)

and \( f(x,y) \in [-27,298,125,665] \). Assume that 169 input-output relations are available as in Table I. The sampled training data in Table I is sketched as in Fig. 6. Then 13 FMF networks are assigned in such a way that each FMF network learns input-output mappings given in a column of Table I. This implies that for each input-output relation, the 13 FMF networks are assigned to approximate \( f(x,y_i) \). The fuzzy rules for designing the i-th FMF network can be generated by observing the training data in the i-th column of Table I. For example, when \( y_i \) is near zero, the fuzzy rules can be given to approximate \( f(x,0) \) as follows:

If input \( x \) is near \(-120\), then output function value \( F \) is NS.
If input \( x \) is near \(-100\), then output function value \( F \) is PS.
If input \( x \) is near \(-80\), then output function value \( F \) is PS.
If input \( x \) is near \(-60\), then output function value \( F \) is NM.
If input \( x \) is near \(-40\), then output function value \( F \) is NM.
If input \( x \) is near \(-20\), then output function value \( F \) is PL.
If input \( x \) is near \(0\), then output function value \( F \) is PH.
If input \( x \) is near \(20\), then output function value \( F \) is PL.
If input \( x \) is near \(40\), then output function value \( F \) is NM.
If input \( x \) is near \(60\), then output function value \( F \) is NM.
If input \( x \) is near \(80\), then output function value \( F \) is PS.
If input \( x \) is near \(100\), then output function value \( F \) is PS.
If input \( x \) is near \(120\), then output function value \( F \) is NS.

where membership functions of the input and the output fuzzy linguistic variables are shown in Figs. 7 and 8, and 'NM', 'NS', 'ZO', 'PS', 'PM', 'PL', and 'PH' imply 'Negative Medium', 'Negative Small', 'Zero', 'Positive Small', 'Positive Medium', 'Positive Large', and 'Positive Huge', respectively.

It is remarked that the whole input space is divided into 13 subspaces and 13 fuzzy rules of the type in (32) are required for each subspace. If each set of fuzzy rules is determined according to the data characteristics in each column, the network may quickly converge to the global minimum. Specifically, for example, let the learning-rate parameter \( \beta_1 \) in (23) be given as 0.2 and let \( \beta_2 \) in (16) be given as 5. If the initial linguistic value of output variable \( F \) is assigned to 'PM', \( \lambda_1 = 20 \). 269 learning trials of FMF network are required to reduce the square of error to be less than 0.1, but if the initial linguistic value of \( F \) is assigned to 'PH', \( \lambda_1 = 120 \), only single learning trial is required.

After completely training 13 FMF networks, the proposed fuzzy-neural interpolating network with the learning rule in (30) is applied to improve the degree of the function approximation. To verify the capability of approximation of our FMF networks incorporating the fuzzy-neural interpolating network, the approximated function is retrieved with 49 \times 49 segmented input data as sketched in Fig. 9. It may be observed from Fig. 9 that our FMF networks incorporating the fuzzy-neural interpolating network can be used as a function approximator, but the network could not reproduce the given...
Fig. 7. Membership functions of the input fuzzy variables in (32).

Fig. 8. Membership functions of the output fuzzy variables in (32).

Fig. 9. Graphical representation of the function approximated by 13 FMF networks incorporating the fuzzy-neural interpolating network.

function completely due to insufficient training data. If the number of subspace is sufficiently large, the given function is expected to be satisfactorily approximated by the proposed FMF networks incorporating the fuzzy-neural interpolating network.

IV. DESIGN OF THE VISUAL SERVO USING THE FMF NETWORKS.

The differential nonlinear mapping \( G(F, \delta F) \) relating the image features and their changes to the desired changes in the camera's position and orientation can be used in the resolved motion rate control (RMRC) scheme [31] as shown in Fig. 10. Since it is difficult to obtain \( G(F, \delta F) \) analytically, \( G(F, \delta F) \) is here approximated by the FMF networks and the fuzzy-neural interpolation network described in Section III. For this, the following issues are discussed in order: i) feature selection for control, ii) determination of desired input-output training data for tracking moving objects, and iii) establishment of necessary fuzzy rules and membership functions for fuzzy set values employed in the fuzzy rules.
In selecting the image features for the visual feedback control, the image recognition criteria including unique features, feature set robustness, computational inexpensive features, and feature set completeness have been proposed in [1], where the feature set completeness was the major issue and was given by the singularity measure of the feature Jacobian matrix. Since the inverse operation is not required in our approach, the singularity measure in [1] is not necessary. Instead, easiness in making up the fuzzy rules is considered, where easiness may be evaluated by means of the number of fuzzy rules. To reduce the number of fuzzy rules, the number of input variables should be reduced. In our case, at least six image features seem to be identified for the 3-D translational and orientational camera motion, which implies that

$$\delta x_i =
G_i(F_1, F_2, \ldots, F_6, \delta F_1, \delta F_2, \ldots, \delta F_6), \text{ for } i = 1, 2, \ldots, 6.$$  (33)

To be specific, let $(\delta x_1, \delta x_2, \delta x_3)$ and $(\delta x_4, \delta x_5, \delta x_6)$, respectively, represent the differential changes in translation along and rotation about $X, Y,$ and $Z$ axes of the camera frame. And let $X, Y,$ and $Z$ denote $X, Y,$ and $Z$ axes of the object frame as shown in Fig. 11. Now consider a quadrangle (specially, a rectangle for the case of the reference image) in the image as shown in Fig. 12, where $F_i,$ for $i = 1, 2, \ldots, 6,$ are given as

- $F_1 = X$ coordinate value of the center of gravity of the quadrangle in the image plane,
- $F_2 = Y$ coordinate value of the center of gravity of the quadrangle in the image plane,
- $F_3 =$ size of the quadrangle in the current image/size of the quadrangle in the reference image,
- $F_4 =$ ratio of lengths of two opposite sides of the quadrangle about the $X$ axis in the image plane,
- $F_5 =$ ratio of lengths of two opposite sides of the quadrangle about the $Y$ axis in the image plane,
- $F_6 =$ degree of the axis of minimum moment of inertia of the quadrangle in the image plane.

For $i = 1, 2, \ldots, 6,$ each $\delta F_i$ is given as the difference between $F_i$ and the reference image feature values $F_i^0.$ $F_i^0$ is the $F_i$'s value measured by the teach-by-showing method\footnote{It means teaching the visual information of the object to the robot when the camera mounted on the robot end-effector is placed a location near by the object.} at the target location $(x_{t1}, y_{t1}, z_{t1})$ as shown in Fig. 13.

It is remarked that use of a quadrangle rather than a triangle or a pentagon allows features representing the position and the
Fig. 14. (c) Relations between $\delta X_1$ and $\delta F_j$, and between $\delta X_3$ and $F_j (j = 1, 2, and 3)$.

Fig. 14. (d) Relations between $\delta X_1$ and $\delta F_j$, and between $\delta X_1$ and $F_j (j = 4, 5, and 6)$.

orientation of the object to be easily obtained. It may not be difficult to choose four points in the image for constructing a quadrangle. However, in the case of spherical or cylindrical objects, construction of a quadrangle in the image is impossible since there are no distinguishable image points. In that case, it may be required to put four markings on the object or to determine other appropriate features as in most of the visual servoing literatures [1], [6]–[9].

Now, note that in (33), if $F_j$ and $\delta F_j$ for some $j \neq i$ can be selected in such a way that $F_j$ and $\delta F_j$ cannot contribute to the generation of $\delta x_i$, then $G_i$ becomes independent of $F_j$ and $\delta F_j$. Thus the fuzzy rules for $G_i$ can be easily established. To investigate the effect of $F_j$ and $\delta F_j$ on the generation of $\delta x_i$, simulations are performed for the object given as a 0.2 m $\times$ 0.2 m square under the assumptions that the focal length and the image scale factor of the pin-hole camera are given as 14 mm and 0.05 mm/pixel, respectively. Fig. 14 shows how $F_i$ and $\delta F_i$, for $i = 1, 2, \ldots, 6$, vary according to the camera motions along and about X, Y, and Z axes. It can be observed from Fig. 14 that $\delta X_1$ and $\delta X_2$ are primarily dependent on $(F_1, F_3, \delta F_1)$ and $(F_2, F_3, \delta F_2)$, respectively, and $\delta X_3, \delta x_4, \delta x_5,$ and $\delta x_6$ are heavily dependent only on $\delta F_3, \delta F_4, \delta F_5,$ and $\delta F_6$, respectively. From these observations, $\delta X_1, \delta X_2, \ldots, \delta X_6$ can be written as $\delta X_1 \cong G_1 (F_1, F_3, \delta F_1), \delta X_2 \cong G_2 (F_2, F_3, \delta F_2),$ and $\delta X_i \cong G_i (\delta F_i)$, for $i = 3, 4, 5,$ and 6, respectively. This property holds for any object with a quadrangle in its image, since such simulation results are generally dependent not on the object shapes but on the chosen features.

It is noted that the dependency of $\delta X_i$ on $F_i$ and $F_3$, for $i = 1$ and 2, can be further simplified by using a single variable $d_i$ defined as

$$d_i \triangleq \text{angle between } ^0X_i \text{ axis and the position vector obtained from the projecting the camera position vector with respect to object frame } (^0X, ^0Y, ^0Z) \text{ on the } (^0X, ^0Y) \text{ plane.}$$

The $d_i$ can be simply estimated by

$$d_i = \tan^{-1} \left( f_i \sqrt{F_i} \left[ K_i F_i \left( \sqrt{F_i} - 1 \right) \right] \right) \text{ for } i = 1 \text{ and } 2, \ (34)$$

where $K_i$ and $f_i$ denote the image scale factor and the focal length of the camera, respectively. Thus $\delta X_i$, for $i = 1, 2, \ldots, 6$, can be simply given as $\delta X_i = G_i (d_i, \delta F_i)$, for $i = 1, 2,$ and $\delta X_i = G_i (\delta F_i)$, for $i = 3, 4, 5,$ and 6, respectively.
Now, we will propose a method to determine the desired input-output training data for moving object tracking. For this, FMF networks will be trained to be capable of tracking a moving object in the whole workspace along the line of sight. Then, input-output training data are generated by considering the view angle of the camera which makes it possible to have a surprisingly small conic training space instead of the whole workspace as a training region as in [12], [13]. To be specific, for a fixed target at \((X_t, Y_t, Z_t)\), consider the conic visible region \(VR_c\) in which the camera can look at the whole object as shown in Fig. 15, where the height of the cone is determined by the maximum moving distance of the camera during one sampling along the line of sight. The top of \(VR_c\) is denoted as \(TVR_c\). It is easily observed that in the viewpoint of camera motion, \(VR_c\) should be replaced with the quadrangular pyramid in Fig. 16 which will be denoted as \(VR\). The top of \(VR\) is denoted as \(TVR\). Then input-output training data are generated from the linear paths connecting the locations on the \(TVR\) to the target location \((X_t, Y_t, Z_t)\), in the object frame. Here the world frame is chosen as the object frame.

It is remarked that most of the neural network based visual serving methods [12], [14], [30] require many training exercises to cover the whole workspace, and thus might be difficult to implement. On the contrary, human visual servoing such as driving a car utilizes only visible region in both training and retrieving phases. That is, when driving a car, if an object such as a road sign or a traffic light looks small, we feel tempted to drive the car at a high speed to reach the object as we expect that the distance between the object and the car is far. But, if the object looks large, we drive the car slowly since we think that the object is close to the car. Such intuitive rules are refined through repetitive driving experiences to acquire better driving skills, and the refined rules hold in any places where the driver has never been before, even if the rules have been updated only within a limited small training region. This principle has been partially employed in [13] and [30] where a global neural network learns the control signal for larger object distances and a local network for smaller object distances. However, the approach in [13] and [30] requires many a learning trial to cover the whole workspace. Specifically, 160 patterns for the global neural
network and 100 patterns for the local neural network were necessary. In addition, to teach the property of the manipulator in the whole workspace, the learning process with the back propagation algorithm was iterated 30,000 times for the global network and 50,000 times for the local network, which seems to be ineffective. Thus, the approach in [13] and [30] can be thought of as an imperfect application of the human visual servoing principle.

Here, we will employ the human visual servoing principle to generate desired input-output training data. If the distance between the camera and the object is larger than a maximum displacement per sampling time, the camera approaches the target at its maximum speed. Otherwise, the camera moves toward the target at lower speed. Thus, learning can be performed by using input-output data within VR in Fig. 16.

To sample the desired input-output data within VR easily, TVR is divided into \( M_1 \times M_2 \) grids and the linear path is generated in such a way that the center of each grid is connected with the target location. Then, VR is eventually divided into \( M_1 \times M_2 \) subspaces. The desired input-output data are obtained on each linear path, where the number of sampling locations on each linear path are decided by fuzzy rules to be employed. It is noted that at each sampling location, the camera can reach the target in one sampling period. Then, we know that the sampling location implies \( \delta x_i \), for \( i = 1, 2, \ldots, 6 \). Since the desired \( \delta x_i \) is given, the desired input-output data can be generated by acquiring features \( F_i \) and \( \delta F_i \) for \( i = 1, 2, \ldots, 6 \) at each sampling location.

Observe that among the nonlinear mappings \( G_i \), for \( i = 1, 2, \ldots, 6 \), input variables \( d_1 \) and \( d_2 \) in \( G_1(d_1, \delta F_1) \) and \( G_2(d_2, \delta F_2) \) can be neglected when the fuzzy rules are to be established, since \( d_1 \) and \( d_2 \) for \( \delta X_1 \) and \( \delta X_2 \) in \( G_1 \) and \( G_2 \) are constants on a given linear path. Then, we know that fuzzy rules for \( G_1 \) and \( G_2 \) can be generated by using only one input variable \( \delta F_1 \) and \( \delta F_2 \), respectively. However, fuzzy rules should be independently generated for every linear path. To avoid such design complexities, we utilize the same linguistic expressions for every linear path, but independently design the membership functions of fuzzy rules for each linear path. An example of such fuzzy rules is given as follows for \( i = 1, 2, \ldots, 6 \):

- If \( F_i \) is negative large, then \( \delta x_i \) is positive large, or
- If \( F_i \) is negative medium, then \( \delta x_i \) is positive medium, or
- If \( F_i \) is negative small, then \( \delta x_i \) is positive small, or
- If \( F_i \) is zero, then \( \delta x_i \) is zero, or
- If \( F_i \) is positive small, then \( \delta x_i \) is negative small, or
- If \( F_i \) is positive medium, then \( \delta x_i \) is negative medium, or
- If \( F_i \) is positive large, then \( \delta x_i \) is negative large.

To design membership functions of fuzzy variables, let \( d_{ij}^l \), for \( l = 1, 2, \ldots, M_1 \), and \( d_{ij}^2 \), for \( l = 1, 2, \ldots, M_2 \), be the constant values of \( d_1 \) and \( d_2 \) at the \( l \)-th linear path, respectively. Then, fuzzy relations for approximating the scalar functions of a single variable \( G_1(d_{ij}^l, \delta F_1) \), for \( l = 1, 2, \ldots, M_1 \), \( G_2(d_{ij}^2, \delta F_2) \), for \( l = 1, 2, \ldots, M_2 \), \( G_3(\delta F_3) \), \( G_5(\delta F_5) \), and \( G_6(\delta F_6) \) can be given as in (7). For notational unification, \( G_3(\delta F_3) \), \( G_5(\delta F_5) \), \( G_6(\delta F_6) \) are denoted as \( G_3(d_{ij}^l, \delta F_3) \), for \( l = 1, 2, \ldots, M_1 \).
G_4(d_{4i}, \delta F_4), for l = 1, 2, ..., M_4, G_5(d_{5i}, \delta F_5), for l = 1, 2, ..., M_5, and G_6(d_{6i}, \delta F_6), for l = 1, 2, ..., M_6, respectively, where M_3 = M_4 = M_5 = M_6 = 1. Let \( G_i \triangleq G_i(d_{4i}, \delta F_4), \) and let \( R_l^1, R_l^2, ..., R_l^{R_l} \) be the \( q_i \) (odd positive integer) fuzzy relations for the approximation of \( G_i \), for \( i = 1, 2, ..., 6 \) and \( l = 1, 2, ..., M_l \). Let \( A_l^i \) and \( B_l^i \) be the fuzzy set value employed in the 'IF' part and the 'THEN' part of the fuzzy relation \( R_l^i \), respectively. As shown in Fig. 16, note that the maximum feature value on the \( l \)th linear path will be less than \( h(M_a, l)S_{F_{\max}} \), where \( h(M_a, l) \) is the maximum feature value for \( M_a \), and \( S_{F_{\max}} \) is the maximum feature value on the \( Z \)th path, respectively. Let \( h(M_a, l)S_{F_{\max}} \) be the fuzzy set value employed in the 'IF' part and the 'THEN' part of the fuzzy relation \( R_l^i \), respectively. For each function approximation, the membership functions of each fuzzy rule can be produced by observing the input data. For the sake of simplicity, membership functions \( \mu_{j,i}^k(\delta F_i) \) are given as a triangular shape shown in Fig. 17, whose equation is described by

\[
\mu_{j,i}^k(\delta F_i) = \begin{cases} \frac{a_{j,i} - b_{j,i}}{c_{j,i} - a_{j,i}} (\delta F_i - b_{j,i}), & \delta F_i \leq a_{j,i} \\ \frac{c_{j,i} - a_{j,i}}{c_{j,i} - b_{j,i}} (\delta F_i - c_{j,i}) + 1, & \delta F_i \geq c_{j,i} \\ 0, & \text{otherwise} \end{cases} \quad (35)
\]

where \( a_{j,i}, b_{j,i} \) and \( c_{j,i} \) are real numbers to be determined. To determine \( a_{j,i}, b_{j,i} \) and \( c_{j,i} \) for \( u_j^k(\delta F_i) \) in (35), let \( \delta F_{\max} \) be the maximum feature value for \( \delta F_i \) and let \( \delta F_{\min} \) be the minimum feature value for \( \delta F_i \) as given by \( \delta F_{\max} = \delta F_{\min} = \delta F_{\max} \). In addition, let \( h(M_a, l) \) be defined by

\[
h(M_a, l) = \begin{cases} (2l - M_a - 1)/(M_a - 1), & l = 1, 2, ..., M_a, \\ 1, & \text{otherwise}. \end{cases} \quad (36)
\]

Note that the maximum feature value on the \( l \)th linear path will be less than \( \delta F_{\max} \). To consider such an effect, we let \( h(M_a, l)S_{F_{\max}} \) be the maximum feature value on the \( l \)th path, in spite of the nonlinear dependency of \( h(M_a, l) \) on \( d_{4i}^2 \). Then, for \( i = 1, 2, ..., 6 \), \( j = 1, 2, ..., q_i \) and \( l = 1, 2, ..., M_l \), \( a_{j,i}, b_{j,i} \) and \( c_{j,i} \) are given by

\[
a_{j,i} = (q_i - 2j + 1)h(M_a, l)S_{F_{\max}}/(q_i - 1), \quad j = 1, 2, ..., q_i, \quad (37)
\]

\[
b_{j,i} = \begin{cases} a_{j,i+1}, & j = 1, 2, ..., q_i - 1, \\ \text{any constant less than } a_{q_i}, & j = q_i, \end{cases} \quad (38)
\]

and

\[
c_{j,i} = \begin{cases} \text{any constant less than } a_{q_i}, & j = 1, \\ a_{j-1}, & j = 2, 3, ..., q_i. \end{cases} \quad (39)
\]

if all the fuzzy-set values are to be equally distributed from \(-h(M_a, l)S_{F_{\max}}\) to \(h(M_a, l)S_{F_{\max}}\). In (37), \( (q_i - 2j + 1)/(q_i - 1) \) plays the role of uniformly dividing \( a_{j,i} \) according to \( j \) in such a way that \( a_{j,i} = -h(M_a, l)S_{F_{\max}} \) and \( a_{q_i} = h(M_a, l)S_{F_{\max}} \). It is remarked that \( q_i \) implying the number of fuzzy rules for \( G_i \) should be determined by observing the relative magnitudes of \( \delta F_{\max} \) for \( i = 1, 2, ..., 6 \). For example, to determine \( q_1, q_2 \) and \( q_3 \), \( \delta F_{\max} \), \( \delta F_{\max} \) and \( \delta F_{\max} \) should be compared with each other. As shown in Fig. 16, \( \delta F_{\max} \), \( i = 1, 2, 3 \) are often computed at different locations \( L_1, L_2 \) and \( L_3 \) on \( VR \) according to the \( \delta X_{\max}, \delta Y_{\max} \) and \( \delta Z_{\max} \), respectively, implying the maximum distances for the camera to move along the \( X, Y \) and \( Z \) axes during one visual sampling time. For the case shown in Fig. 16, \( q_1 = 5 \), \( q_2 \) and \( q_3 \) are chosen so that \( q_1 = 5, q_2 = 3, q_3 = 2 \).

Now, to determine the singleton membership functions \( u_j^k(\delta F_i) \) for \( i = 1, 2, ..., 6 \), \( j = 1, 2, ..., q_i \) and \( l = 1, 2, ..., M_l \), the maximum and minimum magnitudes of the output \( u_i \) are given as \( u_{\min} \) and \( u_{\max} \), respectively, can be used. That is, \( u_j^k(\delta F_i) \) is given by

\[
u_j^k(\delta F_i) = \begin{cases} 1, & \delta F_i \leq \lambda_{j,i} \triangleq (q_i - 2j + 1)u_{\max}h(M_a, l)/(q_i - 1), \\ 0, & \text{otherwise}. \end{cases} \quad (40)
\]

for \( i = 1, 2, ..., 6 \), \( j = 1, 2, ..., q_i \) and \( l = 1, 2, ..., M_l \).

Each FMF network for visual servoing can be constructed and trained as follows. First, using (35)-(39), obtain for \( i = 1, 2, ..., 6 \), \( j = 1, 2, ..., q_i \), and \( l = 1, 2, ..., M_l \) the maximum and minimum magnitudes of the output \( u_i \) given as \( u_{\max} \) and \( u_{\min} \), respectively, can be used. That is, \( u_j^k(\delta F_i) \) is given by

\[
\begin{array}{c|c|c|c|c|c|c}
\delta X_{\max} & \delta Y_{\max} & \delta Z_{\max} & \delta X_{\min} & \delta Y_{\min} & \delta Z_{\min} \\
1.0 & 0 & 0 & 1.0 & 0 & 0 \\
0.5 & 0 & 0 & 0.5 & 0 & 0 \\
0.3 & 0 & 0 & 0.3 & 0 & 0 \\
0.2 & 0 & 0 & 0.2 & 0 & 0 \\
0.1 & 0 & 0 & 0.1 & 0 & 0 \\
0.0 & 0 & 0 & 0.0 & 0 & 0 \\
\end{array}
\]

V. SIMULATION RESULTS

Computer simulations are performed to illustrate the performance of the proposed FMF neural network based visual servoing method. In these simulations, only the kinematic relationship between the object and the camera is considered, and without loss of generality only the position of the camera is controlled. Since the dynamics of the system are not included, the commanded changes in the camera's position are assumed to happen instantly. The imaging camera is modeled as a pinhole lens. The image features are derived from the idealized nondistorted 2-D image points under additive noisy imaging.
conditions, since image distortions are difficult to model and vary widely with lighting, transducer resolution, and linearity. The noisy imaging conditions are modeled as varying levels of a uniformly distributed noise added to the idealized extracted feature values.

For the simulations, the object is given as a 0.2 m × 0.2 m square. The focal length and the image scale factor of the pin-hole camera are assumed to be 14 mm and 0.05 mm/pixel, respectively. The target location (x1, y1, z1) for the reference image is chosen as (0.0 m, 0.0 m, 0.3 m) in the object frame. Maximum distances δX_max, δY_max and δZ_max for the camera to move along X, Y and Z axes during 200 ms visual sampling time are given as 50 mm. By the feature selection process as in Section IV the features F1, F2 and F3, respectively, are chosen as the X and Y coordinates of the center of gravity of the object image, and the size ratio of the object images taken at current camera location and the target location (0.0 m, 0.0 m, 0.35 m) as 0.9272. Since θ12 and θ13, respectively, are chosen as the 0.05 m, 0.0 m, 0.35 m) moves to the target location (0.0 m, 0.0 m, 0.3 m) moves to the target location. It is here observed that the errors converge to almost zero within 250 trials. Similar error convergency has been observed from other cases. Fig. 20 shows the variations of λ122, λ123, and λ311, with respect to the number of learning trials, when the FMF networks are trained in such a way that the camera at (0.0333 m, 0.0167 m, 0.35 m) moves to the target location.

Fig. 19 shows the values of the sum of the error squares given as ∑(δx_i, j - δx_i, j)² with respect to the number of learning trials, when the FMF networks are trained in such a way that the camera at (0.0333 m, 0.0167 m, 0.35 m), (0.0222 m, 0.0111 m, 0.3333 m), and (0.0111 m, 0.00557 m, 0.3167 m) moves to the target location (0.0 m, 0.0 m, 0.3 m) in 200 ms, respectively. It is observed from Fig. 19 that the errors converge to almost zero within 250 trials. Similar error convergency has been observed from other cases. Fig. 20 shows the variations of λ122, λ123, and λ311, with respect to the number of learning trials, when the FMF networks are trained in such a way that the camera at (0.0333 m, 0.0167 m, 0.35 m) moves to the target location. It is here observed that λ311, which implies the fuzzy-set value corresponding to ‘negative large’ in the ‘THEN part’ of the 1st fuzzy rule for the approximation of G3(d1, SF3) which generates the differential motion of the camera along the Z-axis, tends to decrease further at the 250th trial, while λ122 and λ213 converge to 1.9 × 10⁻⁶ and 2.0 × 10⁻⁷ in about 50 trials, respectively. Tendency of decreasing λ311 comes from the exponential saturation property of the function g(·) in (16), which is employed in the FMF neural networks. In other words, the FMF network for G3(d1, SF3) reduces λ311 to exactly produce the desired velocity of the Z-axis motor given as the maximum velocity (50 mm/200 ms) for this case. In cases of λ122 and λ213, the desired outputs are not the maximum velocities, and thus λ122 and λ213 rapidly converge to some constants. Thus, from Figs. 19 and 20, the design of the proposed FMF neural network for the visual servoing is believed to be quite successful. It is remarked that 250, which is the number of trials needed to learn an input-output relationship, is surprisingly small compared to other methods using neural networks.

Now, we let the FMF neural networks with the interpolating networks be trained to follow the untrained linear paths in the averaged sense as described in Section IV. Figs. 21 and 22 show that the path following performance of the FMF neural...
networks with the interpolating network is much superior to that of the FMF neural network without the interpolating network, when the camera is forced to follow the linear path from (0.35 m, 0.25 m, 0.7 m) to (0.0 m, 0.0 m, 0.3 m). Thus, it is believed that the FMF neural networks with interpolating network completely trained only over a small portion of the whole working space can be generally applied to tracking of any other path in the whole space without any further learning trials.

To support such a belief, we will investigate the capability of the proposed FMF neural network based visual servo to track the moving object along the line of sight. For this, we consider the following two cases. i) The case when the object moves along a linear path given as \( Y = X/2 \) with a velocity of 55.9 mm/s and ii) the case when the object moves around a circular path given as \( X^2 + Y^2 = (500 \text{ mm})^2 \) with the angular velocity of 2 revolutions/min. Here, the object frame at the start-up of the visual servoing is chosen as the world frame. The desired and the actual trajectories of the camera for the case i) are shown in Fig. 23, where steady state trajectory errors of 10 and 5 mm are observed for the camera motors of \( X \) and \( Y \) axes, respectively. It is remarked that steady state trajectory errors are totally caused by the computational time delay of one sampling period. Thus, to get rid of the trajectory error, a predictive fuzzy control techniques may be required. Fig. 24(a) and (b) shows the object trajectories and the output trajectories of the camera motors for the case ii), where similar observations to the case i) can be made. It is remarked
that in spite of the constant trajectory error, the path errors are negligibly small as shown in Fig. 24(c).

Fig. 25 shows the path following performances of the controller using our proposed FMF neural network with the interpolating network and for the controller using the feature Jacobian approach [7], [9], when the object image is corrupted by noises. It is observed from Fig. 25 that our proposed visual feedback controller is robust to the noises, while the visual controller using the inverse of the feature Jacobian is seriously sensitive to the noises.

From these simulations, it can be concluded that our proposed visual servoing method based on FMF neural networks works successfully, and is superior to any other previous feature based visual servoing methods in the sense of simple training mechanism, noise-immunity and the tracking capability of moving object in the whole robot workspace.

VI. CONCLUSION

A fuzzy membership function (FMF) based neural network incorporating a fuzzy-neural interpolating network was proposed to approximate the nonlinear mapping for visual servoing. The structure of the proposed networks was similar to that of the radial basis function neural network. The proposed FMF network was trained to be capable of tracking a moving object in the whole workspace along the line...
of sight. For the effective implementation of the proposed networks, the image feature selection mechanism has been investigated and the required fuzzy membership functions have been designed. From the simulation results, it can be concluded that the proposed visual servoing method based on the FNN neural network incorporating the fuzzy-neural interpolating network works quite successfully, and is superior to any previous feature based visual servoing method in the sense of a simple training mechanism, better noise-immunity, and more capability of tracking moving objects in the whole robot workspace. In addition, it is believed that our visual servo controller is practical and can be easily implemented through contemporary microcomputers owing to its simplicity of the design, fast learning speed, and little computation required when retrieving. In our future research work, we are investigating and experimenting with an effective technique for dealing with the computational time-delay and the robot dynamics.

REFERENCES


