'94 KACC (1994. 10. 17 ~ 20)

퍼지신경망을 이용한 로보트의 비쥬얼서보제어

서은택, 정진현
광운대학교 제어계측공학과

Visual Servo Control of Robots Using Fuzzy-Neural-Network

E. T. Seo, C. H. Chung
Dept. of Control and Instrumentation Eng.
Kwangwoon University

ABSTRACT

This paper presents an image-based visual servo control scheme for tracking a workpiece with a hand-eye coordinated robotic system using the fuzzy-neural-network. The goal is to control the relative position and orientation between the end-effector and a moving workpiece using a single camera mounted on the end-effector of robot manipulatort. We developed a fuzzy-neural-network that consists of a network-model fuzzy system and supervised learning rules. Fuzzy-neural-network is applied to approximate the nonlinear mapping which transforms the features and theire change into the desired camera motion. In addition a control strategy for real-time relative motion control based on this approximation is presented. Computer simulation results are illustrated to show the effectiveness of the fuzzy-neural-network method for visual servoing of robot manipulator.

I. INTRODUTION

In traditional visual sensing and manipulation which are combined into an open-loop fashion, 'looking' then 'moving', the performance of the operation directly depends on the accuracy of the visual sensor and the controller. On the other hand, an alternative to increasing the accuracy of the subsystems is to use a visual-feedback control loop, which increases the overall accuracy of the system. Visual servoing is the fusion of results from many elemental areas including high-speed image processing, kinematics, dynamics, control theory, and real-time computing. It has much in common with research into *active vision* and *structure from moving* which are the important issues in the high-level computer vision.

The task in visual servoing is to control the pose of the robot's end-effector, ${}^0\mathbf{x}_{in}$, using visual information, *features*, extracted from the image. Assume that the given robot has 6 DOF(degree of freedom). Pose, $\mathbf{x}$, is represented by a six element vector encoding position and orientation in 3D space. The camera may be fixed, or mounted on the robot's end-effector in which case there exists a constant relationship, ${}^{t6}\mathbf{x}_c$, between the pose of the camera and the pose of the end-effector. The image of the target is a function of the relative pose between the camera and the target, ${}^c\mathbf{x}_t$. Some relivant poses, are shown in Fig. 1.

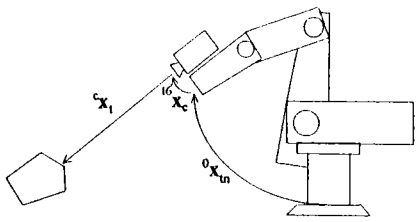


Fig. 1. Relevant coordinate frames; world(0), end-effector(t6), camera(c) and target(t)

The camera forms a 2D projection of the scene onto the image plane where the sensor is located, and during this process direct depth information is lost. Some additional information is needed to determine the 3D coordinate corresponding to an image plane point. This information may come from multiple views or knowledge of the geometric relationship between several feature points on target.

A feature is generally defined as any measurable relationship in an image, e.g., moments, relationships between regions or vertices, polygon face areas, or local intensity patterns. A good feature point is one that can be located unambiguously in different views of the scene, and the coordinates of a feature point or a region centroid are most commonly used. A feature vector, $\mathbf{f}$ is a vector containing feature information as described above.

There is an important classification of visual servo structure; *position-based visual servoing* and *image-based visual servoing*. In position-based control, features are extracted from the image, and used in conjunction with a geometric model of the target to determine the pose of the target with respect to the camera. In image-based servoing the step in which the feature is interpreted to the relative pose of the target is omitted, and servoing is done directly using image features. The image-based approach may reduce the computational delay, eliminate the necessity for image interpretation, and eliminate errors in sensor modeling and camera calibration. However the process is non-linear and highly coupled, and it does cause several problems in controller design.

In this paper, we propose a fuzzy-neural-network(FNN) to

approximate the nonlinear mapping[12] which transform the features and its changes to the desired camera motion. The proposed FNN is a universal function approximator for any real continuous function, and its performance as a function approximator or nonlinear adaptive signal processor is presented in [10]. For the effective application to the visual servo control, appropriate image feature selection process is considered, and the required FNN is designed. Some simulation results to show the successful aplication of our FNN are included.

II. IMAGE-BASED SERVOING

In image-based visual servo control, the location of features on the image plane is directly used for feedback signal. For a robot with a camera mounted on end-effector, the viewpoint, and hence the feature locations in image will be a function of the relative pose of the camera to the target, ${}^c\mathbf{x}_t$. In general this function is non-linear and cross-coupled such that motion of one end-effector DOF will result in the complex motion of many features, e.g., camera rotation may cause horizontal and vertical translations of the feature points on the image plane. This relationship is described as

$$\mathbf{f} = f({}^c\mathbf{x}_t). \tag{1}$$

If we linearize this equation about the operating point, (1) becomes

$$\delta\mathbf{f} = {}^f\mathbf{J}_c({}^c\mathbf{x}_t)\delta{}^c\mathbf{x}_t \tag{2}$$

where ${}^f\mathbf{J}_c({}^c\mathbf{x}_t)$ is a Jacobian matrix relating rate of change in pose space to rate of change in feature space. This Jacobian is refered to as *feature Jacobian*. Assume that the Jacobian is square and non-singular, then

$${}^c\dot{\mathbf{x}}_t = {}^f\mathbf{J}_c^{-1}({}^c\mathbf{x}_t)\dot{\mathbf{f}} \tag{3}$$

is obtained. In turn, the end-effector rates may be converted to manipulator joint rates using the manipulator's inverse Jacobian

$$\dot{\theta} = {}^m\mathbf{J}_\theta^{-1}(\theta){}^m\dot{x}_t \tag{4}$$

where θ represent the joint angles of the robot. Based on these relationships, we can formulate a visual feedback control law.

Such a closed-loop system is relatively robust in the presence of image distortions and kinematic parameter variations in the manipulator Jacobian. A number of researchers have demonstrated results with this image-based approach to visual servoing. The significant problem is to compute or estimate the feature Jacobian. We can find that geometric optics model will be also reuired to obtain the feature Jacobian of which analytical derivation is a quite difficult process, and each element of the feature Jacobian is a function of ${}^c\mathbf{x}$, which especially requires either the CAD model of the object or the estimation of the distance between the object and the camera. This requirement usually makes the feature Jacobian be computationally complicated and much sensitive to the measurement errors of the distance. Furthermore, if ${}^f\mathbf{J}_c({}^c\mathbf{x}_t)$ is singular, the approach using the inverse Jacobian can not be applied to the control of the robot.

Suh, *et al.*[12], propose a method to overcome these drawbacks when using the feature Jacobian. They show that there exist a nonlinear mapping which transforms the features and their changes to the desired camera motion without measurement of the relative distance between the desired camera and the target, and the nonlinear mapping can eliminate several difficulties encountered when using the inverse of the feature Jacobian as in the usual feature-based visual feedback controls. This nonlinear mapping may be approximated by neural networks or adaptive fuzzy systems.

III. FUZZY-NEURAL-NETWORK; FUZZY SYSTEM WITH LEARNING RULE

We consider a simple fuzzy system as follows;

1) fuzzy system has two input variables,
2) each input variable has two fuzzy sets defined by Gaussian membership functions or bell-shaped functions,
3) four fuzzy rules are made from 1) and 2),
4) and centroid defuzzification method is used.

If this fuzzy system uses product-inference logic and singleton fuzzifier for output variable, it can be described by a network model in Fig. 2.

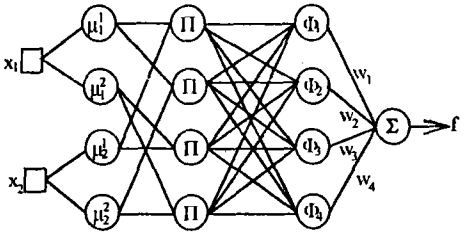


Fig. 2. Fuzzy system as a network model.

This fuzzy system's output is

$$f(\mathbf{x}) = \mathbf{F}^T\mathbf{w}, \tag{5}$$

where $\mathbf{F} = [\Phi_1 \dots \Phi_4]^T$, $\mathbf{w} = [w_1 \dots w_4]^T$;

$$\Phi_k = \frac{\phi_k}{\sum_{j=1}^{4}\phi_j}, \tag{6}$$

where

$$\phi_1(\mathbf{x}) = \mu_1^1(x_1)\mu_2^1(x_2),$$
$$\phi_2(\mathbf{x}) = \mu_1^1(x_1)\mu_2^2(x_2),$$
$$\phi_3(\mathbf{x}) = \mu_1^2(x_1)\mu_2^1(x_2),$$
$$\phi_4(\mathbf{x}) = \mu_1^2(x_1)\mu_2^2(x_2).$$

We can find that the fuzzy system in Fig. 2 is clearly a multilayer feedforward network, and this description provides the basis on which we can apply the learning algorithms used in training the neural networks. Thus we can call this network-model fuzzy system as a *fuzzy-neural-network(FNN)*.

If we simplify fuzzy IF-THEN rules and describe the fuzzy

systems more systematically as in [7], following fuzzy subsystem is available.

$$f(\mathbf{x}_k) = \mathbf{F}^T\mathbf{w} = \frac{\Sigma_{j=1}^{K} w_j(\Pi_{i=1}^{N}\mu_i^j(x_i))}{\Sigma_{j=1}^{K}(\Pi_{i=1}^{N}\mu_i^j(x_i))} \tag{7}$$

where K is the number of IF-THEN rules, N is the number of input variables, and the Gaussian membership function is of the form

$$\mu_i^j = \exp[-0.5(\frac{x_i - m_i^j}{\sigma_i^j})^2]. \tag{8}$$

m_i^j is the center of a fuzzy set defined by μ_i^j and deviation σ_i^j determines the shape of the membership function μ_i^j. An example of this fuzzy subsystem that has two input variables and two fuzzy rules is described as a following netwrok model in Fig. 3.

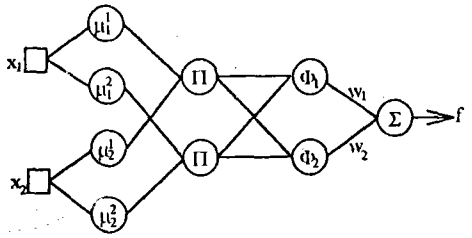


Fig. 3. Subset of fuzzy systems based on simplified rule base.

By Wang[7], it is proved that this is a subset of fuzzy system which may also become a universal approximator for any real continuous function. By using this fuzzy subsystem we can get a remarkable advantage; when we apply the gradient descent method for learning, the output equation of this fuzzy subsystem provides much simpler and more efficient computions of chain rules.

As shown above, fuzzy systems can be represented as feed-forward multi-layer networks in which all parameters are adjustable. Thus some supervised learning algorithm used in training neural networks can be incorporated in fuzzy systems, e.g., back propagation fuzzy systems are available[4][6][7]. On the other hand, if all parameters determining the membership functions of input variables are fixed in (5) or (7), the only free design parameters are w_k and the fuzzy system is linear in these parameters. By adopting this point of view, we are able to use some efficient linear parameter estimation methods, e.g., least square algorithms.

If all parameters determining the membership functions of input variables are fixed, we have a simple *linear regression* model,

$$\begin{aligned} y_k &= f(\mathbf{x}_k) + \varepsilon_k \\ &= \mathbf{F}(\mathbf{x}_k)^T\mathbf{w} + \varepsilon_k \\ &= \sum_{i=1}^{K} w_i\Phi_i(\mathbf{x}_k) + \varepsilon_k \end{aligned} \tag{9}$$

where $x = [x_k\ x_{k-1} \dots x_{k-N+1}]^T$, N is the number of input variables, and K is the number of rules. The problem in *least squares* method is to obtain the estimates of $\{w_i\}$ which minimize the squared error $\left\|\mathbf{F}_k^T\mathbf{w} - y_k\right\|^2$.

Optimum estimates of the parameters, $\mathbf{w}^*$, are given by

$$\mathbf{w}_m^* = [\mathbf{F}_m^T\mathbf{F}_m]^{-1}\mathbf{F}_m^T\mathbf{y}_m \tag{10}$$

where

$$\mathbf{y}_m = \mathbf{F}_m\mathbf{w}_m$$

and

$$\mathbf{y}_m = [y_0\ y_1 \dots y_m]^T, \tag{11}$$
$$\mathbf{w}_m = [w_1\ w_2 \dots w_K]^T, \tag{12}$$
$$\mathbf{F}_m = [\mathbf{F}_0^T\ \mathbf{F}_1^T \dots \mathbf{F}_{m-1}^T]^T. \tag{13}$$

In (13), $\mathbf{F}_k^T = [\Phi_1(\mathbf{x}_k)\ \Phi_2(\mathbf{x}_k) \dots \Phi_K(\mathbf{x}_k)]$, $k = 0, 1, \dots, m-1$. The suffix m indicates that each matrix above is obtained using all m data points. Equation (10) gives the optimum least squares estimate of $\mathbf{w}_m$ which can be obtained using any suitable matrix inversion technique.

The computation of $\mathbf{w}_m$ in equation (10) requires the time-consuming computation of the inverse matrix. Clearly, the LSE method above is not suitable for real-time or on-line filtering. In practice, when continuous data is being acquired and we wish to improve our estimate of $\mathbf{w}_m$ using the new data, recursive methods are prefered. A well-known *RLS(Recursive Least Squares) algorithm* is available.

On the other hand, backpropagation is an algorithm for learning in feedforward networks using *mean squared error* and *gradient descent*. Letting $\mathbf{W}$ be the set of all network parameters, backpropagation employs a nice simplification that makes it easy to find the network's parameter gradient $\nabla\varepsilon(\mathbf{W})$. This allows us to update the current $\mathbf{w}$ by a small step to form new weights $\mathbf{W}^*$ using

$$\mathbf{W}^* = \mathbf{W} - \rho\nabla\varepsilon(\mathbf{W}) \tag{14}$$

The small positive coefficient of ρ controls the step size.

We have

$$\varepsilon_k = y_k - f(\mathbf{x}_k) \tag{15}$$

where $x = [x_k\ x_{k-1} \dots x_{k-N+1}]^T$; N is the number of input variables of fuzzy system, and $f(\mathbf{x})$ is in the form of (5). In *LMS algorithm*[3][8], we would estimate the gradient of mean squared error, $MSE = E[\varepsilon_k^2] \approx \varepsilon_k^2$. At each iteration in the adaptive process, we have a gradient estimate of the form

$$\hat{\nabla}_k = \frac{\partial\varepsilon_k^2}{\partial\mathbf{W}} = 2\varepsilon_k\frac{\partial\varepsilon_k}{\partial\mathbf{W}} \tag{16}$$

where $\mathbf{W}$ is the set of adjustable parameters of given network. If our fuzzy system has Gaussian membership functions in the form of (8), the adjustable parameters are m_i^j, σ_i^j, and w_i. With this simple estimate of the gradient, we can specify a gradient descent algorithm by *chain rule* as follows

$$\mathbf{w}_{k+1} = \mathbf{w}_k - 2\rho\varepsilon_k\mathbf{F}_k, \tag{17}$$
$$\mathbf{m}_{k+1} = \mathbf{m}_k - 2\rho\varepsilon_k\frac{\partial\varepsilon_k}{\partial\mathbf{F}_k}\frac{\partial\mathbf{F}_k}{\partial\mathbf{m}_k}, \tag{18}$$

$$\mathbf{s}_{k+1} = \mathbf{s}_k - 2\rho\varepsilon_k \frac{\partial \varepsilon_k}{\partial \mathbf{F}_k} \frac{\partial \mathbf{F}_k}{\partial \mathbf{s}_k} \tag{19}$$

where

$$\mathbf{m} = \begin{bmatrix} m_1^1 & m_1^2 & \dots & m_1^M \\ m_2^1 & m_2^2 & \dots & m_2^M \\ \vdots & \vdots & & \vdots \\ m_N^1 & m_N^2 & \dots & m_N^M \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} \sigma_1^1 & \sigma_1^2 & \dots & \sigma_1^M \\ \sigma_2^1 & \sigma_2^2 & \dots & \sigma_2^M \\ \vdots & \vdots & & \vdots \\ \sigma_N^1 & \sigma_N^2 & \dots & \sigma_N^M \end{bmatrix},$$

$$\mathbf{w} = [w_1\ w_2 \dots w_K];$$

N is the number of input variables, M is the number of fuzzy sets of each variable, and K is the number of rules made from M^N. If we consider the fuzzy systems in the form of (7), M becomes equal to K and the computations of chain rules become much simpler as in [7]. ρ is the gain constant that regulates the speed and stability of adaptation.

We can combine the gradient descent method and the least sqares estimate to update the parameters in a fuzzy system. This *hybrid learning* procedure is composed of a forward pass and a backward pass. In the forward pass, we supply input data and functional signals go forward to calculated each node output, and linear parameters $\mathbf{w}_k$ of output layer are identified by the RLS algorithm. After identifying parameters $\mathbf{w}_k$, the functional signals keep going forward till error measure is calculated. In the backward pass, the error rates propagate from the output end toward the input end. and the parameters $\mathbf{m}_k$ and $\mathbf{s}_k$ are updated by the gradient descent method in equations (18) and (19). Not only can this hybrid learning rule decrease the dimension of the search space in the gradient method, but, in general, it will also cut down substantially the convergence time.

IV. DESIGN OF FNN FOR VISUAL SERVOING AND SIMULATION RESULTS

The differential nonlinear mapping $\mathbf{g}(\mathbf{f}, \delta\mathbf{f})$[12] relating the image features and their changes to the desired changes in the camera's pose is difficult to analytically obtained, and may be approximated by the FNN developed in the previous section.

Let $(\delta x_1, \delta x_2, \delta x_3)$ and $(\delta x_4, \delta x_5, \delta x_6)$, respectively, represent the differential change in translation along and rotation around the X_c, Y_c, and Z_c axes of the camera frame. Then each relative pose element is desribed as

$$\delta x_i = g_i(f_1, \dots, f_N, \delta f_1, \dots, \delta f_N), \quad i = 1, \dots, 6, \tag{20}$$

where f_k is the feature element and N is the number of features. Each nonlinear mapping g_i is approximated by FNN which is designed considering the dependencies on the characteristics of each feature parameter and learned through all the possible situation in the workspace of the robot.

The test object used in the computer simulation is shown in Fig. 4. We select the following four feature elements from the target object in the image;

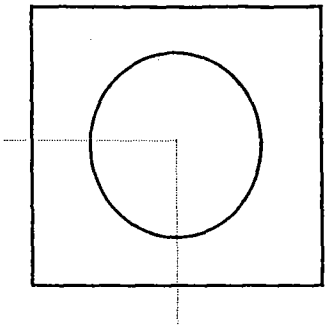


Fig. 4. Target object for simulation

$\mathbf{f}_1$ = the X coordinate value of the region centroid

$\mathbf{f}_2$ = the Y coordinate value of the region centroid

$\mathbf{f}_3$ = the size of the cirlce in the current image / the size of the circle in the reference image

$\mathbf{f}_4$ = the rotation angle of the square in the image frame.

The selection of the features depends on a blend of recognition and control criteria. In typical feature-based servoing, the most important consideration is the effect that changes in the feature positions have on the elements of the feature Jacobian. But in our FNN approach, major interest is the effectiveness in constructing the fuzzy rule base, i.e., the number of input variables included in a rule, and the number of fuzzy rules in a FNN, *etc.* Considering the dependencies of each pose element on the given features, we design the following six FNN;

$$\begin{aligned} \delta x_1 &= g_1(f_1, f_3, \delta f_1), \\ \delta x_2 &= g_2(f_2, f_3, \delta f_2), \\ \delta x_3 &= g_1(\delta f_3), \\ \delta x_4 &= g_1(f_2, \delta f_2, \delta f_3), \\ \delta x_5 &= g_1(f_1, \delta f_1, \delta f_3), \\ \delta x_6 &= g_1(\delta f_4), \end{aligned} \tag{21}$$

Here, each g_i is a FNN which has the form of (7) in the previous section, and hybrid learnin rule that uses both the gradient-descent and the least square method is used.

The test relative motion trajectories and the output results produced by our FNN approximator are presented in Fig. 5. We can find that our FNN approximator tracks the trajectory in each pose parameter successfully. It can be shown that the FNN tracking accuracies in x are within $\pm 0.8\,cm$, y within $\pm 1\,cm$, z within $\pm 1.5\,cm$, *yaw* and *pitch* with ± 0.6 *degree*, and *roll* with ± 0.8 *degree*.

V. CONCLUSION

We develop FNN(fuzzy-neural-network) and apply it to approximate the nonlinear mapping for visual servoing. The eveloped FNN is network-model fuzzy system and has the extensive learning capabilities with gradient descent method and least square estimation. We design the required FNN based on the nonlinear

mapping between the changes in features and the changes in relative pose of object, and train the FNN so that it can track the moving target in the whole workspace. From the simulation results, we can find that the visual servoing based on FNN successfully works, and the trajectory tracking errors are very small. Study on the practical control problem in robot dynamics and experimentation will be a future work.

VI. REFERENCES

[1] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, John Wiley & Sons Ltd., 1993.

[2] B. Kosko, *Neural Network and Fuzzy System*, Prentice Hall, Inc., New Jersey., 1992.

[3] B. Widrow and S. Stearns, *Adaptive Signal Processing*, Prentice Hall: Englewood Cliffs, NJ 1985.

[4] C. C. Jou, 'Supervised Learning in Fuzzy Systems: Algorithms and Computational Capabilities', *IEEE International Conf. on Fuzzy Systems*, 1993.

[5] J. J. Craig, *Introductin to Robotics Mechanics and Control*, 2nd ed., Addison-Wesley Publishing Company, Inc., 1989

[6] J. S. Jang, 'Self-Learning Fuzzy Controllers Based on Temporal Back Propagation', *IEEE Trans. on Neural Networks, Vol. 3, No. 5.*, 1992.

[7] L. X. Wang, 'Backpropagation Fuzzy Systems as Nonlinear Dynamic System Identifiers', *IEEE International Conference on Fuzzy Systems*, 1992.

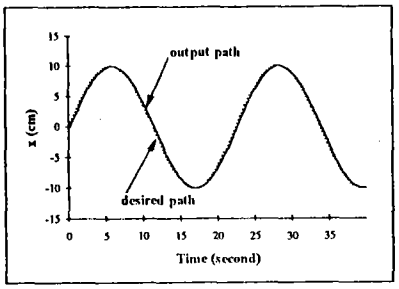
[8] N. Kalouptsidis and S. Theodoridis, *Adaptive Identification and Signal Processing Algorithms*, Prentice Hall International (UK), Ltd., 1993.

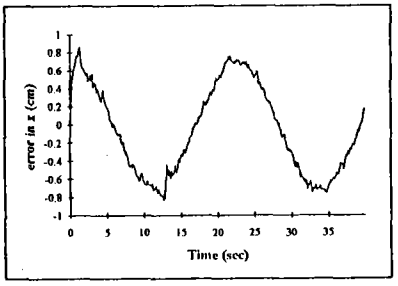
[9] S. I. Gallant, *Neural Network Learning and Expert System*, The MIT Press., 1993.

[10]E. T. Seo and C. H. Chung, 'Application of Sefl-Tuning Filter Based-on a Fuzzy Logic with MC68000', *ASCC*, 1994

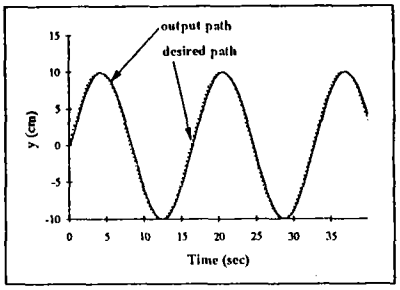
[11]Y. R. Seo and C. H. Chung, 'Application of a fuzzy controller with a Self-learning Structure', Journal of KICS, No.6, Vol.19, 1994

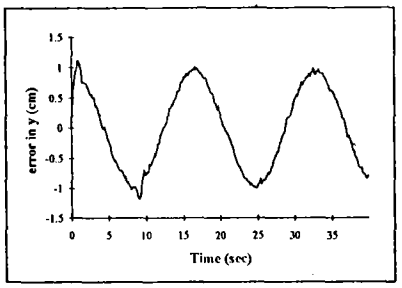
[12]I. H. Suh and T. W. Kim, 'Visual Servoing of Robot Manipulators by Fuzzy Membership Function Based Neural Networks', *Visual Servoing*, pp285-315, World Scientific Publishing Co. Pte. Ltd., 1993



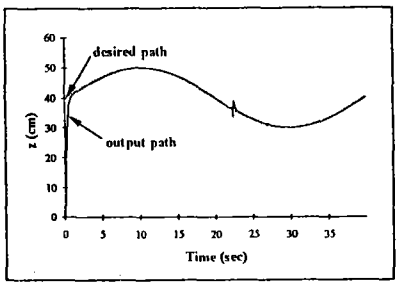


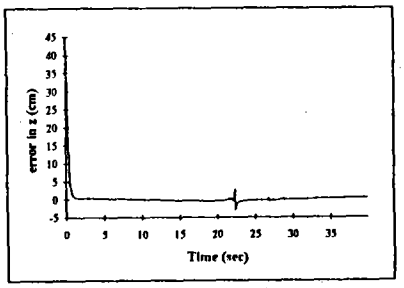
(a)



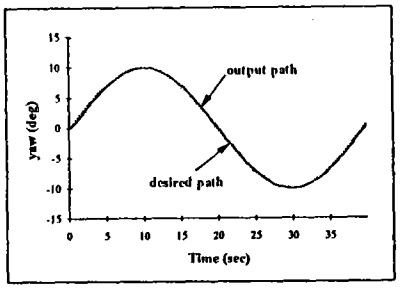


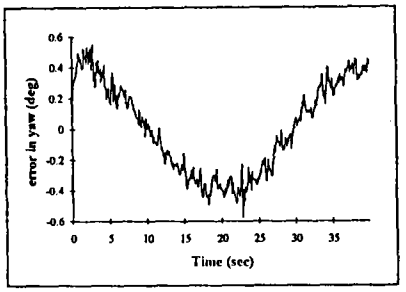
(b)



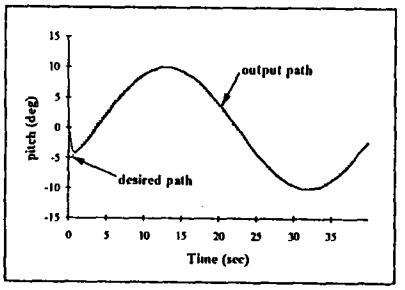


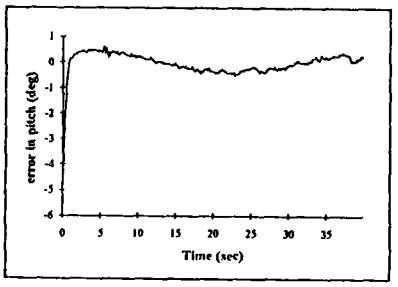
(c)



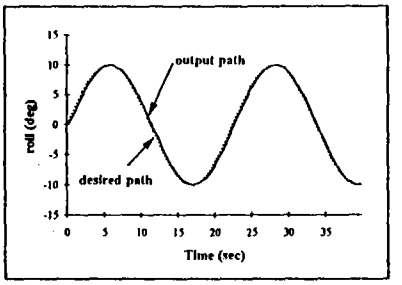


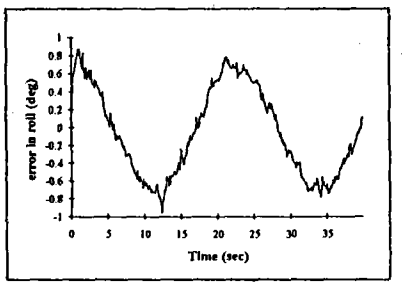
(d)





(e)





(f)

Fig. 5. Simulation Results