PERGAMON

Mechatronics 10 (2000) 1–18

MECHATRONICS

A visual servoing algorithm using fuzzy logics and fuzzy-neural networks

Il Hong Suh[a,*], Tae Won Kim[b]

[a]*Intelligent Control and Robotics Laboratory, Department of Electronics Engineering, Hanyang University, Sa 1-Dong 1271, Ansan-City, Kyungki-Do, 425-791, South Korea*
[b]*W/B Team, Semiconductor System Division, Samsung Aerospace Industries Ltd, 145-3 Sangdaewon 1-Dong, Jungwon-Ku, Sungnam-City, Kyungki-Do, 462-121, South Korea*

Received 27 October 1998; accepted 14 May 1999

Abstract

A visual servoing algorithm is proposed for a robot with a camera in the hand to track a moving object in terms of image features and their variations, where fuzzy logics and fuzzy-neural networks are involved to learn feature Jacobian-based kinematic control law. Specifically, novel image features are suggested by employing a viewing model of the perspective projection to estimate the relative pitching and yawing angles. Such perspective projection-based features would not interact with the relative distance between the object and the camera, and, desired feature trajectories for learning the visually guided line-of-sight robot motion are obtained by measuring features by the camera in the hand not in the entire workspace, but on a single linear path along which the robot moves under the control of a commercially provided function of linear motion, and then, control actions of the camera are approximately found by fuzzy-neural networks to follow such desired feature trajectories.

To show the validity of the proposed algorithm, some experimental results are illustrated, where a four-axis SCARA robot with a BW CCD camera is used.

* Corresponding author.

0957-4158/00/$ - see front matter
PII: S0957-4158(99)00050-1

1. Introduction

Visual servoing has been considered as one of the powerful tools for intelligent robotic applications. Especially, a feature Jacobian has been mainly used for visual servoing. The feature Jacobian can be classified as a pose-based one, if its elements are represented by a relative pose (translation and orientation) between the camera and the object. Otherwise, it is classified as a feature-based feature Jacobian if its elements are represented by relative features. Since the pose-based feature Jacobian is a function of the relative pose, it requires the depth information from the camera to the object, which is often difficult to obtain [16,19,20]. Since a correctional motion of the robot end-effector with a camera in the hand is given by the inverse of the feature Jacobian, computational complexity and the singularity of the feature Jacobian may be considered for real time control of the robot [7,11,12]. In contrast with the pose-based feature Jacobian, a feature-based feature Jacobian is represented by a function of features. In [12], a feature Jacobian was partially described by features. But, it may require a rather complex computation of the inverse of the feature Jacobian at every visual sampling time, and all of the six-dimensional motions of robot end-effectors represented not as functions of features, but as functions of features and pose. Several approaches using neural networks were proposed to learn the pose-based or feature-based feature Jacobian [10,14,15]. A good related literature survey can be found in [5,9].

On the other hand, 6 degrees-of-freedom (DOF) motions of a robot were directly described in terms of six features and their variations, and feature-based feature Jacobian has been approximated by the fuzzy membership function-based neural network (FMFNN) [13,18] to avoid the use of the inverse of the Jacobian. However, desired feature trajectories were given without consideration of robot dynamics, and thus, the robot had to move slowly in practical applications. Furthermore, orientational motions of the camera were not completely considered.

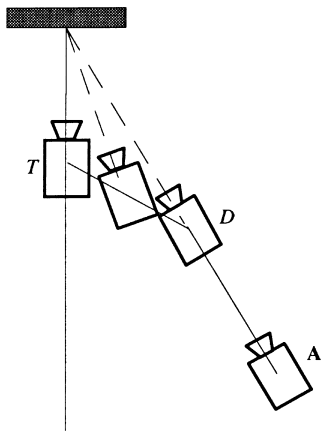

Fig. 1. Schematic diagram of camera motion by employing the proposed visual servoing method.

In this paper, a different type of visual servoing approach from the ones in [1,3,17] is proposed, where two control policies are applied according to the size of the object as in Fig. 1: if the size of the object is measured by a feature to be smaller than a prespecified size at D, then the camera is controlled to move to the object while keeping its gaze holding along the line-of-sight from A to D. Otherwise, the camera is controlled to move along the linear path from the current position to a target position in front of the object, while trying to have a given desired relative orientation between the camera and the object from D to T. It is noted that our proposed visual servoing approach seems to be similar to visual servoing schemes of human or animals. Specifically, to make the desired feature trajectories in such a way that the robot dynamics is involved, the camera in the hand is made to approach the target position T in front of the object, while measuring features and the current position of the robot end-effector. Here, the desired feature trajectories are not obtained in the entire workspace, but on a single linear path along which the robot moves using linear motion in an industrial robot controller. For learning the feature Jacobian-based control law, initial fuzzy rules are given to roughly follow the desired feature trajectory, and then, the motion between D and T, FMFNN in [13,18] is employed to refine fuzzy rules in such a way that the camera in the hand follows the desired feature trajectory. It is noted that novel image features are suggested by employing a viewing model of perspective projection to estimate relative pitching and yawing angles. Such perspective projection-based features would not interact with the relative distance between the object and the camera.

On the other hand, gaze motions are designed for the alignment of the line-of-sight of the camera with the center of the object regardless of the orientation of the camera with respect to the object [2,4]. Thus, it is necessary to adjust the orientation of the camera to become a given desired orientation with respect to the object for correct grasping of the object. However, in our approach such an orientational motion control of the robot end-effector is applied only near the target position. When an orientational motion control as well as a gaze control are simultaneously performed, a line-of-sight of the camera does not usually coincide with the center of the object, and thus the camera will not move along the desired linear path. Thus, corrective motions on the plane perpendicular to the approaching direction of the robot end-effector should be considered. Such corrective motion control algorithms are proposed by fuzzily estimating the current position of the camera with respect to the object.

To show the validity of our proposed algorithms, some experimental results are illustrated, where a four-axis SCARA robot with a CCD camera is utilized. It is noted that the camera is mounted on the rolling axis (S-axis) of the robot in such a way that the line-of-sight of the camera lies in the X–Y plane.

2. Image features for the design of visual servoing algorithms

In selecting image features for a visual feedback control, image selection criteria

including unique features, feature set robustness, computational inexpensive features, and feature set completeness have been proposed in [8]. Among them, uniqueness should be strongly considered, since it gives effects on the computational complexity of the visual feedback control algorithms. However, it is usually difficult to find a unique feature because camera motion along the i-th axis of the camera frame usually cause not only the i-th feature, F_i, but also other features, F_j $(j \neq i)$, to be changed. To cope with such a difficulty, a viewing model of perspective projection in [6] is employed here.

Specifically, consider a quadrangle (especially, a rectangle for the case of the reference image) in the image as shown in Fig. 2. As shown in [3,5,11,13], features obtained from four points or a quadrangle in the image plane could be generally applicable to real tasks. Hereinafter, an object will be considered as a quadrangle. Let (x_1, y_1), (x_2, y_2), (x_3, y_3) and (x_4, y_4) be the corner points of the rectangle in the image plane as shown in Fig. 2. Then, features, F_i, for $i = 1, 2, \ldots, 6$, are chosen as

$$F_1 = (x_1 + x_2 + x_3 + x_4)/4,$$

$$F_2 = (y_1 + y_2 + y_3 + y_4)/4,$$

$$F_3 = (x_2 + x_4 - x_1 - x_3) \cdot (y_2 + y_4 - y_1 - y_3),$$

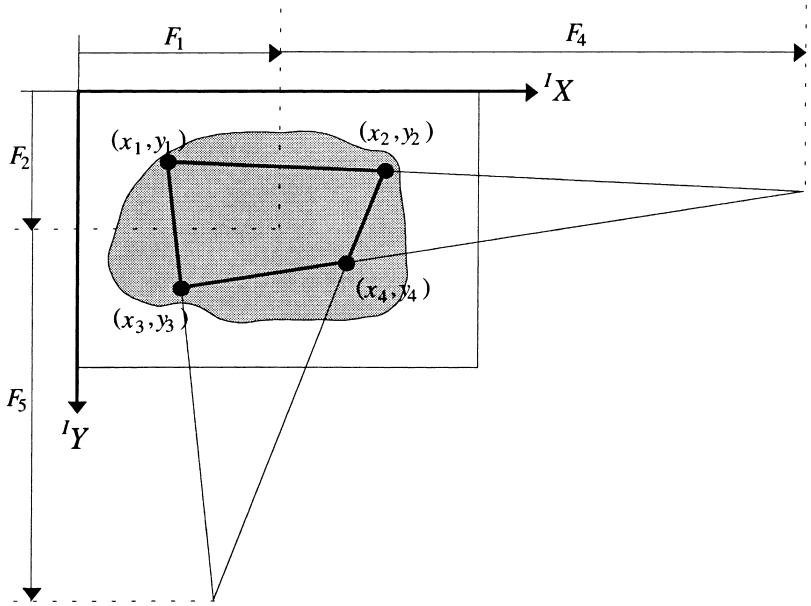

Fig. 2. A quadrangle extracted from simple four dot pattern in the image plane.

$$F_4 = (m_1x_1 - m_2x_3 + y_3 - y_1)/(m_1 - m_2) - F_1,$$

$$F_5 = (m_3[m_4(x_1 - x_2) + y_2 - y_1])/(m_3 - m_4) + y_1 - F_2, \tag{1}$$

and

$$F_6 = \tan^{-1}([y_1 + y_2 - y_3 - y_4]/[x_1 + x_2 - x_3 - x_4]).$$

In Eq. (1), m_1, m_2, m_3, and m_4 are given as $(y_2-y_1)/(x_2-x_1)$, $(y_4-y_3)/(x_4-x_3)$, $(x_1-x_3)/(y_1-y_3)$, and $(x_2-x_4)/(y_2-y_4)$, respectively.

It is noted that F_1 and F_2, respectively, are X and Y coordinates of the center of gravity of the quadrangle in the image plane. F_3 is the area of the quadrangle in the image plane. Let $l_{ij}(x, y)=0$ be the line connecting (x_i, y_i) with (x_j, y_j). F_4 implies the difference between the F_1 and X coordinate of the point of intersection of two lines $l_{12}(x, y)=0$ and $l_{34}(x, y)=0$ as shown in Fig. 2. In computation of F_4, if m_1 becomes equal to m_2, then F_4 is not computed, but given as a prespecified large number, and, F_5 is the difference between the F_2 and Y coordinate of the point of intersection of two lines $l_{13}(x, y)=0$ and $l_{24}(x, y)=0$. F_6 is the degree of rotation of the quadrangle about the normal vector of the image plane. It is also noted that F_1, F_2 and F_3, respectively, are used for translational motions of the camera along the ${}^cX$-, ${}^cY$- and ${}^cZ$-axes. F_4, F_5 and F_6, respectively, are used to determine magnitudes of a pitching, a yawing, and a rolling motion of the camera about the ${}^cX$-, ${}^cY$- and ${}^cZ$-axes [18].

Now consider a viewing model of perspective projection of a regular

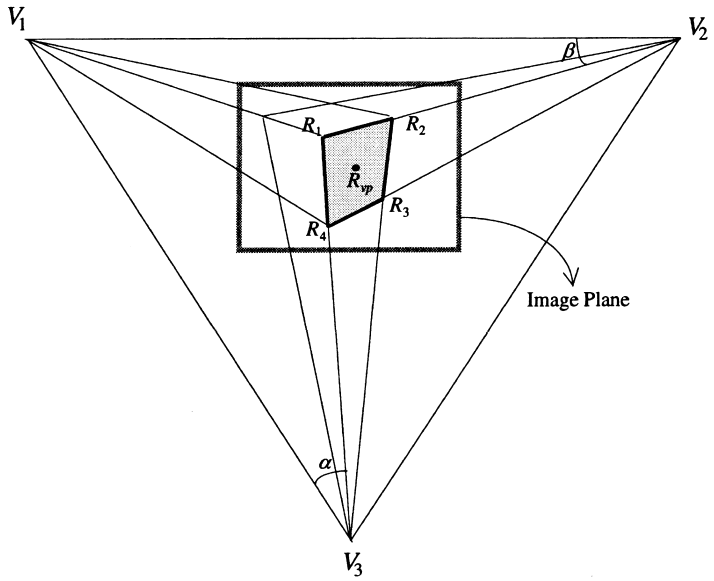

Fig. 3. A viewing model of perspective projection of a regular hexadron.

hexahedron as shown in Fig. 3 to know how relative pitching and yawing angles between the object and the camera affect image features. Let a quadrangle with corner points C_1, C_2, C_3 and C_4 be denoted as $Q(C_1, C_2, C_3, C_4)$. Let $\mathscr{R}(R_1, R_2, R_3, R_4)$ in Fig. 3 be the image of $Q(C_1, C_2, C_3, C_4)$ on the object to be visually tracked, and R_{vp} be the viewpoint of the camera which is transformed from the center of the camera frame to the image plane. Also let the plane including three vanishing points, V_1, V_2 and V_3, be represented as $\mathscr{P}(V_1, V_2, V_3)=0$. Here, note that $\mathscr{P}(V_1, V_2, V_3)=0$ is orthogonal to the line-of-sight of the camera [6,13]. Assume that the viewpoint of the camera coincides with the center of $Q(R_1, R_2, R_3, R_4)$. Then, the angle α between $\mathscr{P}(V_1, V_2, V_3)$ and the line connecting V_3 with C_1, $\overline{V_3C_1}$, which is called a vanishing line, becomes a pitching angle of the camera with respect to the object. The angle β between $\mathscr{P}(V_1, V_2, V_3)$ and $\overline{V_2C_1}$ becomes a relative yawing angle. It is noted here that the angles α and β, respectively, can be simply represented by the difference of the Y coordinates of R_{vp} and V_2 and the difference of the X coordinates of R_{vp} and V_2 [6]. It is also noted that the distance between R_{vp} and V_2 or V_3 is not varied even if the relative distance between the camera and the object is changed as shown in Fig. 4. Thus, to obtain relative pitching and yawing angles, α and β, R_{vp}, V_2 and V_3 should be found. Here, transformed R_{vp} in the image plane can be easily known, since the origin of the camera frame is always mapped onto a fixed point in the image plane, and V_2 and V_3 can be computed by using the intersection of $\overline{R_1R_2}$, and $\overline{R_4R_3}$, and $\overline{R_1R_4}$, and $\overline{R_2R_3}$, respectively. However, V_2 and V_3 are not linearly proportional to the angle α and β, respectively. Thus, the fuzzy rules here are designed to represent the relations between the angle α and V_2, and the angle β and V_3. It is noted that V_2

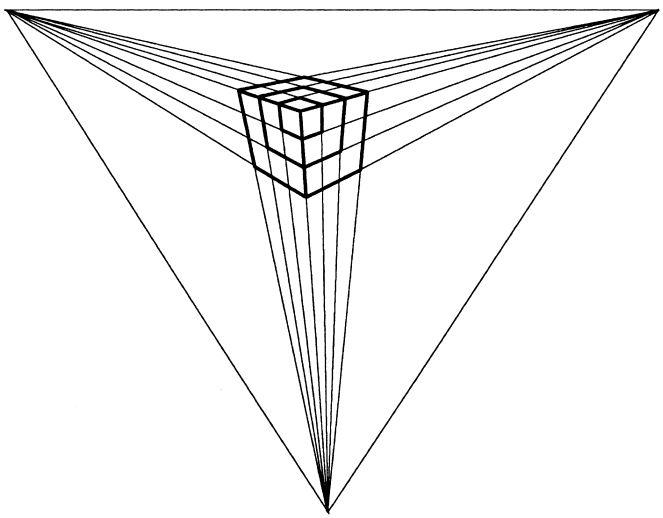

Fig. 4. A schematic diagram of a viewing model of perspective projection to show that all vanishing lines are met into vanishing points even if the viewing scale is changed.

and V_3 in Fig. 3 are the same as F_4 and F_5, respectively. For this, the line-of-sight of the camera is made to be aligned to the center of the object, and the camera is in front of the object as shown in Fig. 5. To get relations between F_4 and the relative yawing angle of the camera with respect to the object, the object is rotated in a clockwise or counter-clockwise direction about the X-axis of the object frame, ${}^{o}X_1$, from $-\beta_{\max}$ to $\beta_{\max}$ by an increment of β_{step}, while computing F_4 at every yawing angle. Then, the fuzzy rules can be given to represent the relations between β and F_4 as follows:

$$\text{If } F_4 \text{ is } \mathbf{NEAR}\ \mathcal{J}_4^i, \quad \text{then } \beta \text{ is } \mathbf{NEAR}\ \Omega_y^i, \quad \text{for } i = 1, 2, \ldots, n, \tag{2}$$

where n is the number of fuzzy rules given as $(2\beta_{\max}/\beta_{\text{step}} + 1)$, **NEAR** $\mathcal{J}_4^i$ is the i-th linguistic value of F_4, and **NEAR** Ω_y^i is the i-th linguistic value of β.

In the case of the relative pitching angle, the difference of the Y coordinate between R_{vp} and V_3 is employed, and, since F_6 is linearly proportional to the relative rolling angle, the scale factor is only required to get a real relative rolling angle.

3. Desired feature trajectories and learning of line-of-sight motion

Desired feature trajectories should be chosen in such a way that learning both line-of-sight motion of a robot end-effector and correct positioning without oscillations at the target position are guaranteed. For this, consider the case, without loss of generality, that a robot end-effector is made to move along a linear path from a position L_1 to the target position T in the camera frame as shown in Fig. 5. Here, such a linear motion is achieved by means of the function of linear move that is basically provided in most of commercial industrial robot controllers [8]. Fig. 6(a) and (b), respectively, show a typical position trajectory and the velocity profile for such a linear motion of the robot end-effector when the dynamics of the robot is assumed as $6.5/(s + 6.5)$. In Fig. 6, (t_A, A) and (t_D, D), respectively, imply the time and the position at which acceleration becomes zero, and deceleration begins to reduce the velocity. To let the end-effector of a

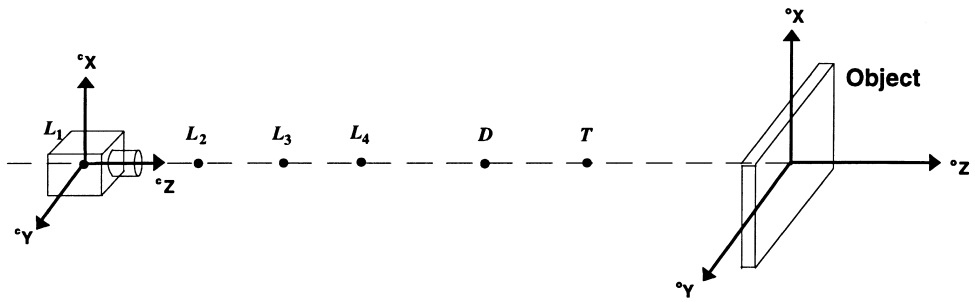

Fig. 5. The line-of-sight of the camera which is aligned to the center of the object.

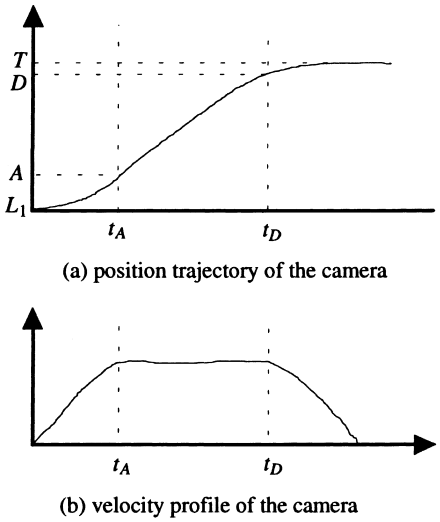

Fig. 6. A position trajectory and a velocity profile of a camera in the hand when the robot moves from L_1 to T.

robot learn how to follow such a position trajectory or a velocity profile under our feature-based feature Jacobian control algorithm, we will let the robot move with its maximum velocity unless the magnitude of features F_3 is smaller than F_3^D, the magnitude of F_3 is measured at D. The reference features necessary for learning visually guided motion are only obtained along the linear path from D to T at every visual sampling time. We will now call them a desired feature trajectory, $F_3^R(t)$.

On the other hand, the robot end-effector can begin to move from an arbitrary position L_i, for $i = 2, 3, \ldots, n$, on the linear path as shown in Fig. 5. Thus, corresponding velocity profiles become different from the case that the robot begins to move from L_1. Thus, feature trajectories corresponding to several velocity profiles are obtained along the same linear path from D to T by letting the end-effector of the robot begin to move from several different positions. Therefore, the robot end-effector velocity as well as features are required to identify which feature trajectory should be chosen as follows. Here, variation of features F_3, δF_3, is to be used instead of robot velocity for such identification.

Now, let δX_3 be the camera motion along the ${}^cX_3$ ($={}^cZ$)-axis during on visual sampling time, and let $G(F_3, \delta F_3)$ be the relationship between $(F_3, \delta F_3)$ and δX_3. To approximately get $G(F_3, \delta F_3)$, a modified version of FMFNN in [13,18] is used. Specifically, $G(F_3, \delta F_3)$ is approximated by fuzzily combining m functions, $G_i(F_{3i}, \delta F_{3i})$, for $i = 1, 2, \ldots, m$. For this, an approximated $G_i(F_{3i}, \delta F_{3i})$ is initially found by fuzzy rules, where δF_{3i} is assumed, without loss of generality, to be given, and then is iteratively improved by FMFNN. To design such initial m fuzzy rules for the coarse tracking, it is noted that precise control actions need to be applied near the set point to prevent an overshoot of the feature trajectory.

Thus, fuzzy values to represent the feature trajectory should be given to be dense near the set point. For the case that δF_3 is similar to δF_{3i}, an example of fuzzy rules can be given as

If F_3 is `Large`, then $\delta^c X_3$ is `Small`, or

If F_3 is `Medium`, then $\delta^c X_3$ is `Medium`, or

If F_3 is `Small`, then $\delta^c X_3$ is `Large`. (3)

Here, F_3 is used for implicitly representing the distance between the robot and the object. Now, for the fine visual tracking, initial fuzzy rules for coarse tracking are represented as a form of a neural network, FMFNN [13,18], given by

$$\delta^c X_3 = \sum_{i=1}^{q} \lambda_i \Phi_i(F_3). \tag{4}$$

The basis function of FMFNN, $\Phi_i(\cdot)$, is a triangular membership function of the input fuzzy variable of the i-th fuzzy rule. The weight of FMFNN, λ_i, is the singleton membership function of the output fuzzy variable. Now, λ_i is iteratively refined by using the reference feature trajectory, $F_3^R(t)$. For this, an error function for an FMFNN is given as

$$E(t) = \tfrac{1}{2}(F_3^R(t) - F_3(t))^2 \tag{5}$$

and F_3 can be represented as

$$F_3 = I({}^cX_3) = \Gamma\left(g\left(\sum_{i=1}^{q} \lambda_i \Phi_i\right)\right), \tag{6}$$

where $I(\cdot)$ is a mapping from ${}^cX \in R^m$ to $F \in R^n$, $\Gamma(\cdot)$ is a mapping from $\delta^c X \in R^m$ to $F \in R^n$, and $g(\cdot)$ is an output node function as usual in a neural network. Without loss of generality, $g(\cdot)$ can be given as

$$g(u) = ku, \tag{7}$$

where k is a slope of linear output node function. From Eqs. (5) and (6), the derivative of the error function $E(t)$ with respect to the weights of the i-th FMFNN, λ_i, can be obtained as

$$\frac{\partial E(t)}{\partial \lambda_i(t)} = -(F_3^R(t) - F_3(t))\frac{\partial F_3(t)}{\partial \lambda_i(t)}. \tag{8}$$

Unfortunately, it is impossible to get the derivative of $F_3(t)$ with respect to $\lambda_i(t)$ as a closed form. But, in the case of F_3, it is obvious that the size of the object in the image plane is always increasing when the camera is approaching the object.

Thus, since we can assume that the sign of λ_i's variation is the same as $F_3(t)$'s variation, $\partial F_3(t)/\partial\lambda_i(t)$ can be replaced as $\partial\hat{F}_3(t)/\partial\lambda_i(t)$ given by

$$\frac{\partial\hat{F}_3(t)}{\partial\lambda_i(t)} = \eta \ \mathrm{sgn}\left(\frac{F_3(t)-F_3(t-1)}{\lambda_i(t)-\lambda_i(t-1)}\right). \tag{9}$$

It is noted that the convergence speed of $E(t)$ when using $\partial F_3(t)/\partial\lambda_i(t)$ is not the same as the one of $E(t)$ when using the estimated derivative in Eq. (9), but the convergence direction of $E(t)$ will be the same. Thus, the derivative of $F_3(t)$ with respect to $\lambda_i(t)$ can be obtained as

$$\frac{\partial E(t)}{\partial\lambda_i(t)} \approx -k\eta(F_3^R(t)-F_3(t))\mathrm{sgn}\left(\frac{F_3(t)-F_3(t-1)}{\lambda_i(t)-\lambda_i(t-1)}\right) \tag{10}$$

and, the learning rule for adapting the weight of the modified FMFNN is given as follows:

$$\lambda_i(t+1) = \lambda_i(t) + k\eta(F_3^R(t)-F_3(t))\mathrm{sgn}\left(\frac{F_3(t)-F_3(t-1)}{\lambda_i(t)-\lambda_i(t-1)}\right). \tag{11}$$

It is noted that convergence of the FMFNN was discussed in [18]. It is also noted that feature trajectories that are not learnt a priori can be encountered. Such cases can be handled by using previously learnt line-of-sight motions. For this, a simple linear interpolation method is used here, since visual servoing dynamics for line-of-sight motions are not expected to be much different. That is, δX_3, the motion command for the ${}^cZ$-axis of the camera is determined as

$$\begin{aligned}\delta X_3 &= \Psi(F_3)\,|_{\delta F_3} \\ &= \frac{|\,\delta F_3^i - \delta F_3\,|\,G_{i+1}(F_3,\delta F_3) + |\,\delta F_3^{i+1} - \delta F_3\,|\,G_i(F_3,\delta F_3)}{|\,\delta F_3^i + \delta F_3^{i+1}\,|},\end{aligned} \tag{12}$$

when δF_3 is measured as a value between δF_3^i and δF_3^{i+1}.

4. Visual servoing control law using feature-based feature Jacobian

It is noted that our design problem is to find a visual servoing control law by using a feature-based feature Jacobian. We first found the motion command, δX_3, for the ${}^cZ$-axis of the camera as $\Psi(F_3)$ in Eq. (12) by utilizing FMFNN. Now, we will determine the translational motion commands, $\delta\,{}^cX_1$, $\delta\,{}^cX_2$, and $\delta\,{}^cX_3$, of the camera along the linear path from D to T during a visual sampling time by using $\Psi(F_3)$.

To endow our visual controller with a gaze holding capability for the case that the line-of-sight of the camera does not coincide with the center of an object, it is necessary to determine how much the camera should be rotated about the

perpendicular axes with respect to the ${}^{c}Z$-direction along which the camera approaches the object. Without loss of generality, consider a simple 2-D configuration of a camera and an object as shown in Fig. 7.

Then, by using the geometric relationship between the camera and the object, F_1 in Eq. (1) can be represented as

$$F_1 = k_s \frac{f}{{}^{c}Z_o} {}^{c}Y_o, \tag{13}$$

where f and k_s, respectively, denote the focal length of the camera and an image scaling factor and can be known a priori. In Eq. (13), ${}^{c}Y_o$ and ${}^{c}Z_o$, respectively, are the ${}^{c}Y$ and ${}^{c}Z$ directional positions of the object with respect to the camera frame. Thus, by using Eq. (13), the angle g_β for gaze holding can be obtained using only feature F_1 as

$$g_\beta = \tan^{-1}\left(\frac{{}^{c}Y_o}{{}^{c}Z_o}\right) = \tan^{-1}\left(\frac{F_1 \, {}^{c}Z_o/k_s f}{{}^{c}Z_o}\right) = \tan^{-1}\left(\frac{F_1}{k_s f}\right). \tag{14}$$

It is noted here that since dynamics for rotational motions of the camera are relatively faster than those for translational motions of the camera, orientational motions of the camera can be controlled without considering the robot dynamics. To be specific, let $g_{\beta m}$ be the angle of the camera to be maximally rotated during one visual sampling time, which can be determined by a mechanical and electrical specification of the robot to be used. Then, if g_β in Eq. (14) is smaller than $g_{\beta m}$, the camera is made to be rotated by the angle of g_β in one visual sampling time. Otherwise, the camera is made to be rotated by the angle of $g_{\beta m}$ until the current gaze angle g_β becomes smaller than $g_{\beta m}$. It is also noted that the pitching angle of the camera for gaze holding in a 3-D configuration, g_α, can be determined by

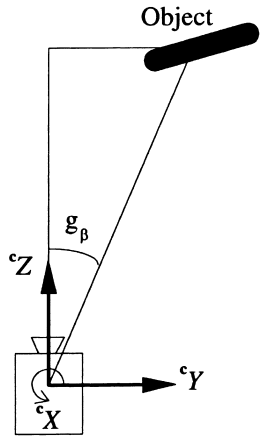

Fig. 7. 2-D configuration for representing the relative yawing angle of the camera with respect to the object.

$$g_\alpha = \tan^{-1}\left(\frac{F_2}{k_s f}\right). \tag{15}$$

It is further noted that g_α and g_β in Eqs. (15) and (14) are used for controlling the approaching direction of the camera in the hand to follow the linear path from A to D as shown in Fig. 1. Recall, that in our visual servoing problem, a desired path to be visually followed is not given as an actual line-of-sight path from the position D to the center of the object, but given as the linear path from D to the target position T. Thus, for the visually guided linear motion from D to T, $D({}^{o}X_D, {}^{o}Y_D, {}^{o}Z_D)$ is necessary to be estimated. To avoid use of a rather complex and/or expensive distance measuring technique in [16], a fuzzy-based position–estimation technique is now proposed, which can estimate the relative position of the camera with respect to the object. Specifically, fuzzy rules are designed by using empirically obtained relational data between F_3 and the actual distance, where relative pitching and yawing angles are fixed to be null. For this, let d_{DT} be the distance between D and T, and let F_3^D and F_3^T be the features measured at D and T, respectively. Also let δF_3^{TD} be defined as $(F_3^T - F_3^D)/n$, where n is an integer implying the number of fuzzy rules to be designed. Then, choose d_{DTi} on a linear path from D to T in such a way that

$$d_{DTi} = F_3^D + i \cdot \delta F_3^{TD}, \quad \text{for } i = 1, 2, \ldots, n. \tag{16}$$

Since the position measuring rules to be proposed are designed under the assumption that the relative orientation between the camera and the object is near zero, some errors may be generated if the relative orientation is not zero. That is, F_3 can be decreased due to nonzero yawing and pitching angles. To reduce such errors, consider the 3-D configuration of the camera and the object, where if an object whose size is L is declined to α and β with respect to ${}^{c}X$ and ${}^{c}Y$, respectively, the length of the object is scaled down as $L \cos\alpha \cos\beta$. Since α and β can be estimated by the fuzzy rules in Eq. (2), the actual size of the object can be estimated by $\hat{F}_3$ given as

$$\hat{F}_3 = F_3 \cos^{-1}(\alpha) \cos^{-1}(\beta). \tag{17}$$

Then, fuzzy rules for estimating the relative position between the camera and the object can be given as

$$\text{If } \hat{F}_3 \text{ is } \textbf{near } F_{3i}, \quad \text{then } {}^{o}Z_c^* \text{ is } \textbf{near } ({}^{o}Z_T + d_{DTi}), \quad \text{for } i = 1, 2, \ldots, n, \tag{18}$$

where ${}^{o}Z_c^*$ and ${}^{o}Z_T$, respectively, are the estimated relative positions of the camera and the pre-measured target position with respect to the object. Here, linguistic values **near** F_{3i} and **near** $({}^{o}Z_T + d_{DTi})$, respectively, are given as triangular and singleton membership functions.

After getting the estimated position between the camera and the object, the moving direction of the camera has to be calculated to follow the linear path between D and T as shown in Fig. 1. Thus, since T is given a priori with respect to the object frame and the current camera position is estimated by employing

Eqs. (17) and (18), the unit vector for the line from the current camera position to T, $\mathbf{u}_c$, can be obtained as

$$\mathbf{u}_c = \begin{bmatrix} u_{cx} \\ u_{cy} \\ u_{cz} \end{bmatrix} = \frac{1}{\sqrt{\tan^2 \alpha + \tan^2 \beta + 1}} \begin{bmatrix} \tan \alpha \\ \tan \beta \\ 1 \end{bmatrix}. \tag{19}$$

Then, the translational motion commands, $\delta^c X_1$, $\delta^c X_2$, and $\delta^c X_3$, of the camera along the linear path from D to T during a visual sampling time can be computed by using $\Psi(\hat{F}_3)$ in Eq. (12) as

$$\begin{bmatrix} \delta^c X_1 \\ \delta^c X_2 \\ \delta^c X_3 \end{bmatrix} = \Psi(\hat{F}_3)\mathbf{u}_c. \tag{20}$$

Controls of $\delta^c X_1$, $\delta^c X_2$ and $\delta^c X_3$ for camera motions along a linear path from D to T might cause the camera to lose gaze holding. Thus, it is necessary that the orientation of the robot should be readjusted for the gaze holding. For this, a pitching angle, E_α, and a yawing angle, E_β, during the camera motion along a linear path from D to T can be obtained geometrically by using ${}^oZ_c^*$, $\delta^c X_1$, $\delta^c X_3$, α and β as follows:

$$E_\alpha = \tan^{-1}\left(\frac{{}^oZ_c^* \cos \alpha - \delta^c X_1}{{}^oZ_c^* - \delta^c X_3}\right), \tag{21}$$

and

$$E_\beta = \tan^{-1}\left(\frac{{}^oZ_c^* \cos \beta - \delta^c X_1}{{}^oZ_c^* - \delta^c X_3}\right). \tag{22}$$

E_α and E_β in Eqs. (21) and (22) are valid for the case that the camera is modeled as a pin-hole lens and the output values of the fuzzy rules in Eqs. (2) and (18) are exactly the same as actual values. Thus, in addition to E_α and E_β in Eqs. (21) and (22), an additional orientational motion may be required to reduce orientational errors. To comply with such a requirement, pitching and yawing control commands, δX_4 and δX_5 are given by incorporating gazing angles g_β and g_β, respectively, in Eqs. (14) and (15) as follows:

$$\delta X_4 = E_\alpha + k_p g_\alpha, \tag{23}$$

and

$$\delta X_5 = E_\beta + k_p g_\beta, \tag{24}$$

where k_p is given as a positive constant less than unity.

Now, the rolling control commands, δX_6, of the camera in the hand is simply given to be proportional according to the error between the desired and the actual

rolling degree, where the actual rolling degree is estimated by using feature F_6, which is independent to the relative distance between the camera and the object.

5. Experimental results

To show the validity of the proposed visual servoing method, some experimental results are illustrated. For this, a four-axis SCARA robot, SPR-600 [25], with a B/W CCD camera [23] is utilized, where the camera is mounted on the rolling axis (S-axis) of the robot in such a way that the line-of-sight lies in the X–Y plane of the robot frame. Specifically, the CCD camera is composed of a signal module, IK-M41MK, and a lens module, IK-M30 M, of which the focal length is 7.5 mm. The control system employs an MC68030/68882-based commercial CPU board (Force CPU 30 [21]) and a commercialized real-time multitasking O.S, VxWorks [24]. An additional FORCE CPU 30 and a B/W frame grabber, DT-1451 [22], are embedded into the control system for image acquisition and feature extraction. On the other hand, to reduce computational complexity for image processing the object is given as a 4×4 cm white square on a dark background. The target position T is chosen as (25.3, 55.8, and 10.8 cm) in the robot frame. The maximum distances $\delta X_{\max}$, $\delta Y_{\max}$, and $\delta Z_{\max}$ for the camera to move along the X-, Y- and Z-axes of the camera frame during one visual sampling time of 160 ms are given as 11 mm. Here, the sampling time for the control of the robot is chosen as 40 ms.

To compute the position trajectory, the camera at L_1 (13.0, 55.8, and 10.8 cm) is made to move to T with its maximum speed. Then, the deceleration position D is chosen from the position trajectory of the camera, and, F_3^D is computed at D. To compute the feature trajectory, the camera is also made to move from L_1 to T, while F_3 is computed, where F_3's larger than F_3^D are memorized for the feature trajectory. For learning the line-of-sight motion, 11 initial fuzzy rules for the relative yawing angle between the camera and the object in Eq. (2) are used, and membership functions of their input variables are shown in Fig. 8. Now, for the fine visual servoing, the weight of FMFNN, λ_i, in Eq. (4) is adapted by employing Eq. (11). Fig. 9 shows the trajectory of F_3 while the FMFNN is trained. It is observed from Fig. 9 that the steady state error converges to near zero within 30 trials. To compute other feature trajectories for various robot speeds passing

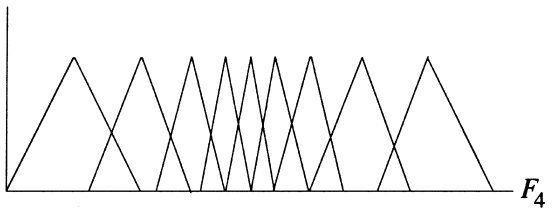

Fig. 8. Membership functions for input.

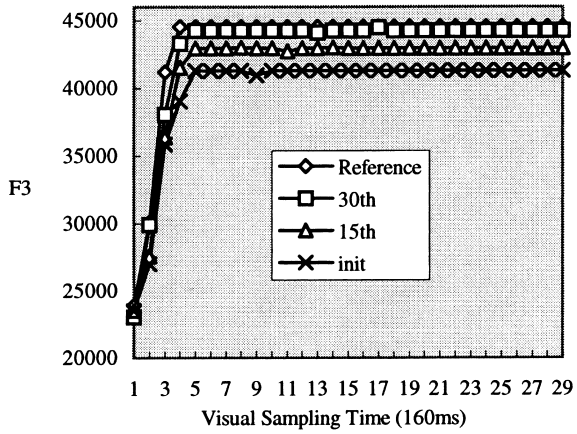

Fig. 9. Feature trajectories with respect to the number of learning trials, when the FMFNN is trained in such a way that the camera moves from D to T.

through the position D, the camera in the hand at L_2 (15.0, 55.8, and 10.8 cm), L_3 (17, 55.8, and 10.8 cm) and L_4 (19, 55.8, and 10.8 cm) is moved to target position T. After learning such feature trajectories, the linear interpolation method in Eq. (12) is employed to handle feature trajectories that are not learnt a priori.

Fig. 10 shows the servoing performances of the proposed algorithm, when the camera is visually controlled to move from untrained locations, A_1 (13.0, 58.5, and 10.8 cm), A_2 (13.1, 61.2, and 10.8 cm), A_3 (13.0, 63.96, and 10.8 cm) and A_4 (13.0, 66.87, and 10.8 cm) to the target location T. Specifically, at the starting positions A_i, $i = 1$, 2, 3, and 4, the camera is turned by the gaze angle, g_β, in

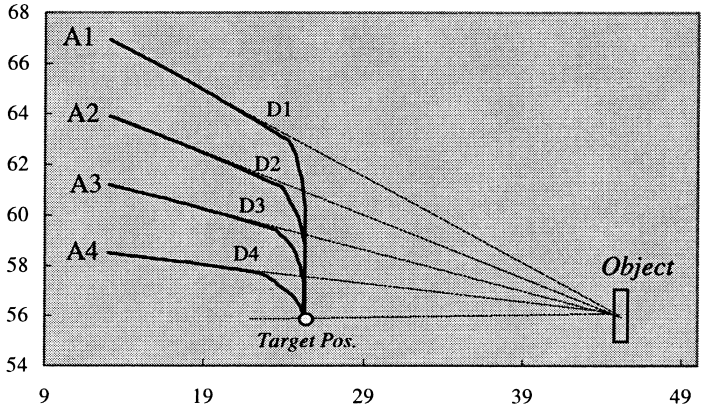

Fig. 10. The servoing performances of the proposed algorithm, when the camera is visually controlled to move from A_1 (13.0, 58.5, and 10.8 cm), A_2 (13.1, 61.2, and 10.8 cm), A_3 (13.0, 63.96, and 10.8 cm) and A_4 (13.0, 66.87, and 10.8 cm) to the target location T.

Eq. (14) for the initial gaze holding. Then, the camera is controlled to move from A_i to D_i along the line-of-sight with its maximum speed. It is recalled that the visual servoing controller determines whether it reduces speed commands by comparing the current feature F_3 and F_3^D. When the current F_3 is larger than F_3^D, translational motions of the camera to the target position T are controlled by incorporating Eqs. (2), (12), and (17)–(20). The orientational motions are controlled by Eqs. (23) and (24). It is observed from Fig. 10 that actual paths of the camera from A_i to D_i, $i = 1$, 2, 3, and 4, are almost linear. This implies that the line-of-sight motion control with gaze holding works successfully in the whole workspace as expected. It is also observed from Fig. 10 that the actual paths from D_i, $i = 1$, 2, 3, and 4, to T shows a tolerable average error of 2 mm, which might be caused by untrained camera motions along the ${}^cX$- and ${}^cY$-axes, but shows little positioning errors at T. Thus, the proposed visual servoing algorithm for the control phase from D_i to T seems to be valid on any paths in the whole workspace, even though FMFNN is trained only on a single path.

On the other hand, we investigate the capability of the proposed visual servoing method to track the moving object along the line-of-sight. For this, we consider the case that the object moves along a linear path from (43.0, 55.8, and 43.0 cm) to (44.7, 55.2, and 41.2 cm) with a velocity of 3 mm/s, and the camera traces the object from (39.0, 80.0, and 39.0 cm). The desired and the actual trajectories of the camera are shown in Fig. 11, where a steady state error of 0.5 mm is observed for the Y-axis of the camera frame. It is noted that the steady state error is mainly caused by computational time delay of one visual sampling time. It is also noted that the camera motions in 6 DOF can be easily extended by using similar approaches used in this experiment.

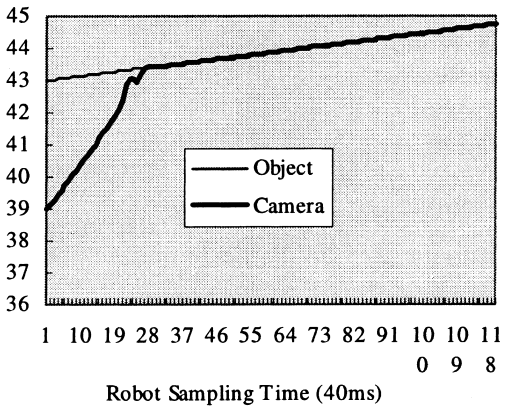

Fig. 11. Tracking performance of the proposed algorithm when the object moves along a linear path from (43.0, 55.8, and 43.0 cm) to (44.7, 55.2, and 41.2 cm) with a velocity of 3 mm/s, and the camera traces the object from (39.0, 80.0, and 39.0 cm).

6. Concluding remarks

In this paper, novel features and an improved visual servoing algorithm were proposed. A viewing model of perspective projection was used to get the image features F_4 and F_5 with relatively high noise-immunity and size invariant characteristics. Owing to the uniqueness of the proposed image features, at most two input variables were employed for the design of fuzzy logics and/or fuzzy-neural networks. To compensate dynamic characteristics of the robot, desired feature trajectories for learning visually guided line-of-sight robot motions were obtained by measuring features by the camera in the hand not in the entire workspace, but on a single linear path along which the robot moves under the control of a commercially provided function of linear motion, and then, the control actions for the line-of-sight motion of the camera are approximately found by fuzzy-neural networks to follow desired feature trajectories when the orientational motions of the camera are controlled by gaze holding. From the experimental results, it was shown that the proposed visual servoing method worked successfully on any paths in the whole workspace.

The advantages of our proposed visual servoing can now be summarized as follows: (1) the robot dynamics can be effectively considered by using the feature trajectories; (2) a novel perspective model based image feature is robust against image noises and/or computation errors; (3) relatively little geometric information is required on the camera, the object and the environment; (4) the amount of learning is small since the line-of-sight motions of a robot-mounted camera are required to be learnt not in the entire workspace, but on a single linear path; (5) line-of-sight motions of the robot can be guaranteed in the whole workspace by using both the line-of-sight motion on a single linear path and the gaze holding; and (6) comparing the conventional feature Jacobian, relatively fast computation can be achieved since no computations of the inverse of the feature Jacobian are required.

References

[1] Allen PK, Yoshimi B, Timcenko A. Real-time visual servoing. In: Proceedings of the IEEE International Conference on Robotics and Automation, 1991. p. 851–6.
[2] Brown C. Gaze controls with interactions and delays. IEEE Trans on System, Man, and Cybernetics 1990;20(1):518–27.
[3] Chaumette F, Rives P, Espiau B. Positioning of a robot with respect to an object, tracking it and estimating its velocity by visual servoing. In: Proceedings of the IEEE International Conference on Robots and Automation, Sacramento, CA, 1991. p. 2248–22553.
[4] Coombs DJ, Brown CM. Cooperative gaze holding in binocular vision. IEEE Control Systems 1991;11(4):24–33.
[5] Corke PI. Video-rate robot visual servoing. In: Hashimoto K, editor. Visual servoing: real-time control of robot manipulators based on visual sensory feedback. World Scientific, 1993.
[6] Doblin J. Perspective — a new system for designers, 1956.
[7] Feddema JT, Mitchell OR. Vision-guided servoing with feature-based trajectory generation. IEEE Trans Robotics and Automation 1989;5(5):691–700.

[8] Feddema JT, Lee CSG, Mitchell OR. Weighted selection of image features for resolved rate visual feedback control. IEEE Trans Robotics and Automation 1991;7(1):31–47.
[9] Hager GD, Hutchinson S. Visual servoing: achievements, issues, and applications. Workshop Notes of IEEE International Conference on Robotics and Automation on Visual Servoing, San Diego, CA, 1994.
[10] Hashimoto H, Kubota T, Lo W-C, Harashima F. A control scheme of visual servo control for robotic manipulators using artificial neural network. In: Proceedings of the IEEE International Conference on Control and Applications, Jerusalem, Israel, 1989. p. TA3–TA6.
[11] Hashimoto K, Kimoto T, Ebine T, Kimura H. Manipulator control with image-based visual servo. In: Proceedings of the IEEE International Conference on Robotics and Automation, 1991. p. 2267–72.
[12] Jang W, Bien Z. Feature-based visual servoing of an eye-in-hand robot with improved tracking performance. In: Proceedings of the IEEE International Conference on Robotics and Automation, 1991. p. 2254–60.
[13] Kim TW. A study on fuzzy-neural network-based visual servoing of a robot manipulator. Ph.D. thesis, Hanyang University, Seoul, Korea (in Korean), 1995.
[14] Kuperstein M. Generalized neural model for adaptive sensory-motor control of single postures. In: Proceedings of the IEEE International Conference on Robotics and Automation, 1988. p. 140–3.
[15] Miller WT. Sensor-based control of robotic manipulators using a general learning algorithm. IEEE Trans Robotics and Automation 1987;3(2):157–65.
[16] Nevatia R. Depth measurement by motion stereo. Computer Graphics and Image Processing 1976;5:203–14.
[17] Papanikolopoulos NP, Khosla PK. Shared and traded telerobotic visual control. In: Proceedings of the IEEE International Conference on Robotics and Automation, 1992. p. 878–85.
[18] Suh IH, Kim TW. Fuzzy membership function based neural networks with applications to the visual servoing of robot manipulators. IEEE Trans Fuzzy Systems 1994;2(3):203–20.
[19] Vernon D, Tistarelli M. Using camera motion to estimate range for robotic parts manipulation. IEEE Trans Robotics and Automation 1990;6(5):509–21.
[20] Yuan JS-C. A general photogrammetric method for determining object position and orientation. IEEE Trans Robotics and Automation 1989;5(2):129–42.
[21] CPU-30 user's manual, Force Computers Inc, 1991.
[22] User manual for DT-1451, Data translation, 1993.
[23] User manual for IK-M41MK, Toshiba, 1993.
[24] VxWorks programmer's guide, Wind River System Inc, 1990.
[25] Operating manual for SPR-600 SCARA robot, Samsung Aerospace Ind, 1990.