Robotic Inserting a Moving Object using Visual-based Control with Time-Delay Compensator

Jianhua Su, Senior Member, IEEE, Lili Wang, Chuankai Liu and Hong Qiao, Fellow, IEEE

Abstract—Tracking-and-inserting a moving peg using a robot manipulator is a challenging task in manufacturing. In the past decades, various visual-based methods have been proposed for robotic manipulating static targets, which usually ignore the time delay in robot command transmission and image processing. However, for tracking and inserting a moving peg, time delays cannot be overlooked because they can reduce the tracking performance and even cause manipulations to fail. In this paper, a robot visual-based control with a time-delay compensator is presented to solve the problem of inserting a moving peg. The time-delay compensator was designed using RBFNNs, a feedback compensator aimed at eliminating the tracking errors caused by the time delays. Thus, we could manipulate a moving object using a commercial industry robot, even with the time-variant delays in the control loop. Furthermore, the visual-based controller with the pseudo-inverse image Jacobian matrix was designed using a linearization model. Thus, the matrix could be efficiently updated using the model. In the experiment, we inserted a peg into a moving hole using an eye-in-hand robot with precision.

Index Terms—Tracking-and-inserting, Visual-based control, Time-delay compensator.

I. Introduction

Tracking-and-inserting a moving peg is an important task in manufacturing. In some cases, such as robot assembly tasks, a robot manipulator is required to follow a moving hole and then insert a peg into it. Visual-based robot controllers focus on controlling the motion of a robot with visual feedback and had been used in robot tracking-and-manipulating tasks. The visual-based robot control is generally classified into two categories [1], i.e., position-based visual servo (PBVS) and image-based visual servo (IBVS). IBVS methods are less sensitive to calibration errors and thus have advanced in recent years. This work would employ IBVS as the robot visual-based controller.

A. Tracking-and-manipulating a static target

A typical robot visual-based task is to track-and-manipulate a static object or follow a predefined trajectory, and the performance of the visual-based control mainly depends on the estimated speed and precision of the robot Jacobian matrix. Thus, online image Jacobian matrix identification is an important problem to be addressed in manipulating a static object. Kosmopoulos [2] presented a method for the robust estimation of the feature Jacobian matrix by training the feature Jacobian for visual-based robot manipulating systems. With uncertain robot kinematics, dynamic models, and camera depth, Cheah et al. [3] used an adaptive Jacobian SP-ID setpoint controller, where the depth parameters were updated online. Wang [4] proposed two adaptive controllers based on image-space observers that realized the image-space tracking objective without relying on the image-space velocity measurement and requiring the inversion of the estimated camera depth. Lizarralde [5] formulated the visual tracking problem as a relative degree two multiple-input multiple-output adaptive controller problem, where the robot kinematics and camera parameters of the Jacobian matrix were indirectly updated in the adaptive scheme. Khan et al. [6] proposed a linear matrix inequality approach to estimate the camera parameters and inverse kinematics, thus circumventing the calibration of camera parameters for pseudo-inverse or inverse Jacobian matrices. Zhang and Li [7] proposed an inversion-free IBVS method for an eye-in-hand camera configuration; they designed the robot controller using a recurrent neural network in joint space for tracking a static object. Hu et al. [8] used a quaternion formulation to represent the rotation tracking error and then developed an adaptive homography-based visual tracking controller. Liu et al. [9] adopted an adaptive algorithm to calibrate the camera parameters online and employed fuzzy logic systems to approximate the unmodeled nonlinear robot dynamics and external disturbances. Hwang et al. [25] used the reinforcement learning method to resolve system noise and uncertainties in the estimation of the robotic Jacobian and interaction matrices and implemented the method in tracking a target using an eye-in-hand configuration. However, they only tested the method in the simulation environment.

The aforementioned methods exhibited competent tracking precision in visual-based manipulations, despite the uncertain camera calibration parameters, robot kinematics, and dynamics. However, most of them focused mainly on tracking a predefined trajectory or a static object. A few of them...
considered the tracking of a moving object due to the limited field of view of the camera.

B. Tracking-and-manipulating a moving target

On the other hand, some researchers aim to tracking-and-manipulating a moving object, where the trajectory is first predicted, following which the object is manipulated at the estimated intercept point. To address the problem of tracking and grasping a moving object using the robot eye-to-hand configuration, Allen et al. [10] first computed the motion parameter of the object from vision, then moved the robotic arm to track the intercept and grasp the object through predictive control. However, the accuracy of the manipulation was not examined in their work. Gong et al. [23] used a geometric particle filter to track a moving object and updated the robot’s motion using dynamic motor primitives until the robot could grasp a moving object. Salehian et al. [11] focused on catching a flying object by estimating its trajectory, then intercepting it at the estimated intercept point. For hitting a ball with a table tennis robot, Zhang et al. [12] predicted the trajectories of the ball before and after being hit based on its visual measurement and motion model and then estimated the hitting point based on the ball’s trajectory. Chen and Lu [13] employed a Soft-Actor-Critic deep learning algorithm to learn the robot’s action directly from the image for grasping a moving object.

However, the eye-to-hand configuration (the camera observed the robot within its workspace) had a less precise but global sight of the scene, whereas eye-in-hand configuration (the camera was mounted on the robot end-effector) has a partial but precise sight of the scene [30]. In order to perform tracking-and-inserting tasks, this work employed eye-in-hand configuration to achieve precise target localization. The limited field of view of eye-in-hand configuration required the robot to move as soon as it has seen the target, but the time delays would significantly reduce the tracking performance [29]. Recently, researchers developed methods for grasping moving objects with eye-in-hand configuration. For example, Peng et al. [31] estimated the position of the moving target with an improved kernel correlation filter algorithm and then controlled an eye-in-hand robot arm to reach the position. The grasping was performed when the pose error between the target and the end of the robot arm was lower than a predefined threshold. Chen et al. [32] predicted the motion of the target by the combination of Kalman filter and interpolation and then employed a refined PID controller for adjusting the error between the desired and actual pose of the robot end-effector. Note that both methods were position-based visual servo controllers, where the estimation of the moving target position was important for manipulating the target. But we directly computed the desired position of the robot arm according to the image tracking error and then performed the robot insertion when the tracking error was less than the threshold. That is, we employed an image-based servoing control approach that did not need to estimate the motion of the target. Wong et al. [33] estimated the moving target using Long Short-Term Memory (LSTM) based network and predicted the grasping point of the moving object with a Convolutional Neural Network (CNN) based on the captured image. Their work was very similar to an open-loop method that generated the robot motion without feedback. But our method was a close-loop control method using the feedback. Note that all three works [31-33] also focused on the grasping of a moving object, which allowed greater tracking error than that of the insertion in our work. For example, the average perdition error of reference [33] was about 2 mm, while our average tracking error of ours was about 0.3 mm. Therefore, we developed a leaning-based method to compensate for unmodeled aberrations to achieve precision manipulation.

C. The purpose of this work

The time delay of a robot manipulator is generally the result of image processing and the transmission of the external command as a setpoint to the inner control loop. To neutralize the time delay, some compensation methods have been proposed to improve visual tracking accuracy. For example, Laiacker et al. [27] measured the time delay between the object localization and manipulator control with a target marker and then compensated it based on the predicted motion of the robot manipulator. In the proposed method, the manipulator repeatedly grasped an object with an accuracy higher than 2 cm. However, the time delay in the system was assumed to be a constant value and the target object was static. Fujimoto et al. [26] applied a disturbance observer to compensate for the time delay in image processing. They proposed a multi-rate controller to achieve smaller tracking errors than that single-rate controllers. However, it was necessary to know the time delay $T_{\text{ld}}$ for the design of the observer.

To precisely track and insert a moving target using an eye-in-hand robot under time-variant delays, we propose a radial basis function neural network (RBFNN)-based adaptive visual-servo control method. The method was realized by adjusting the image Jacobian matrix online and compensating for time delays using the RBFNNs. The proposed controller can track and assemble a moving object using a robot eye-in-hand configuration. The major contributions of this work are as follows.

a) To the best of our knowledge, the problem of tracking-and-inserting a moving object based on the robot eye-in-hand configuration is under-researched. To closely follow the target object, we designed a feedback time-delay compensator using RBFNNs to eliminate the tracking errors caused by the time delays in the robot’s visual servo control loop. Some neural networks (NNs), such as RBFNNs, multi-layer feedforward NNs, and recurrent NNs, have been employed to approximate a system inverse for dynamical compensation attributable to the time delays in command transmission [14]. However, the compensator is a feedforward compensator and also needs samples to train the parameters of the NNs. We designed a feedback compensator, where the output of the NNs was computed online according to the tracking performance, to compensate for the time delays in the robot visual servo control loop, as well as in the command transmitting and the image processing.

b) To address the uncertain camera parameters, we derived the visual servoing controller with the pseudo-inverse of the image Jacobian as a linearization model, and then adjusted the matrix by the linear model. The update of the matrix was related
only to the previous state of the visual servoing system; thus, it afforded computational efficiency.

The rest of this paper is arranged as follows. In Section II, the tracking problem is presented and the Jacobian matrix adaptive method is proposed in Section III. The time-delay compensator is discussed in Section VI. Finally, the comparison of several conducted experiments of the proposed method with traditional image-based tracking methods is discussed.

II. PROBLEM STATEMENT

The focus of this work was on tracking-and-inserting a moving object using an eye-in-hand robot manipulator. The robot manipulator was required to closely follow the target object such that it was unable to escape from the view field of the camera. Thus, the peg held by the robot could be precisely inserted into the moving hole, where the clearance between the peg and the hole was 1 cm. The control scheme of the tracking and inserting system, which includes a master controller, a robot controller, and several joints controller, is shown in Fig. 1. The master controller was designed to calculate the trajectory of the robot with visual feedback. The robot controller was used to receive and process the commands from the master controller by communication channels such as TCP/IP.

The command transmission and image processing are often prone to time delay, whose values are often varying and imprecise, rather than deterministic. Generally, the uncertain time delays affect the performance of the robot, especially in tracking-and-inserting a moving peg.

![Fig. 1. Structure of the visual-based robot control.](image)

To solve the problem, a feedback compensator was developed in this study to eliminate time delays arising in the course of command transmission and image processing in the robot’s visual-based control loop. Furthermore, we employed the adaptive image Jacobian matrix to address the uncertainties of hand-eye parameters. We called the proposed method image-based visual servoing control with a time-delay compensator (IBVS-TDC). The objective of the IBVS-TDC was to minimize tracking errors due to the uncertainty concerning the matrix \( e_{tae}(t) \) and time delay in command transmission \( e_{delay}(t) \). In the following sections, the details of the IBVS-TDC method are discussed.

III. IMAGE JACOBIAN MATRIX ADAPTATION

In this section, we briefly describe the traditional visual-based robot control method and present a control with a pseudo-inverse image Jacobian matrix adaptation algorithm.

A. Pseudo-inverse image Jacobian matrix

We employed a pinhole camera model to describe the visual servoing system. Assuming that \( \mathbf{x}_i \in \mathbb{R}^2 \) \((i = 1, ..., m)\) is the coordinate of the projection of the \( i \)th feature point on the image plane and \( \dot{\mathbf{r}} \in \mathbb{R}^6 \) is the spatial velocity of the camera corresponding to the robot base frame. The relationship between the velocity of the feature point on the image plane and the velocity of the camera can be described as follows [3, 4, 15]:

\[
\mathbf{x}_i(t) = \mathbf{J}_{r_i} \dot{\mathbf{r}}_i(t),
\]

where \( \mathbf{J}_{r_i} \) is the image Jacobian matrix.

Assuming that \( m \) feature points \( \mathbf{x}_i(i = 1, ..., m) \) are selected by the feature extraction algorithm and the positions of the desired feature points and the corresponding velocities are \( \mathbf{x}^*_i(i = 1, ..., m) \) and \( \dot{\mathbf{r}}^*_i(i = 1, ..., m) \), respectively. Thus, the tracking error is defined by the following:

\[
\mathbf{E} = \begin{bmatrix}
\mathbf{x}_1^* - \mathbf{x}_1 \\
\vdots \\
\mathbf{x}_m^* - \mathbf{x}_m
\end{bmatrix},
\mathbf{\dot{E}} = \begin{bmatrix}
\dot{\mathbf{r}}_1^* - \dot{\mathbf{r}}_1 \\
\vdots \\
\dot{\mathbf{r}}_m^* - \dot{\mathbf{r}}_m
\end{bmatrix} \in \mathbb{R}^{2m \times 1}
\]

The trajectory tracking model employed in this work is the sliding-model control method, a typical method used in robot tracking control [16]:

\[
s(t) = \mathbf{\dot{E}} + K_t \int \mathbf{E} dt + K_p \mathbf{E}, \quad K_t, K_p > 0.
\]

Note that \( s(t) = 0 \) defines a stable sliding surface. The purpose of designing a sliding-mode controller is to push the system onto the sliding surface by making \( s(t) \) small.

Setting \( s(t) = 0 \) in (3) yields:

\[
\mathbf{\dot{E}} + K_t \int \mathbf{E} dt + K_p \mathbf{E} = 0.
\]

Substituting (1) and (3) into (4) yields:

\[
\mathbf{J}_t (\dot{\mathbf{r}}^* - \dot{\mathbf{r}}) + K_t \int \mathbf{E} dt + K_p \mathbf{E} = 0,
\]

with

\[
\mathbf{J}_t = \text{diag}(\mathbf{J}_{r_1}, ..., \mathbf{J}_{r_m}) \in \mathbb{R}^{2m \times 6m},
\]

\[
\dot{\mathbf{r}} = [\dot{\mathbf{r}}_1, ..., \dot{\mathbf{r}}_m]^T \in \mathbb{R}^{6m \times 1}, \quad \dot{\mathbf{r}}^* = [\dot{\mathbf{r}}^*_1, ..., \dot{\mathbf{r}}^*_m]^T \in \mathbb{R}^{6m \times 1}.
\]

Then, multiplying both sides of the equation by \( \mathbf{J}^T_t \in \mathbb{R}^{6m \times 2m} \) gives

\[
\mathbf{r} = \mathbf{J}^T_t (\dot{\mathbf{r}}^* + K_p \mathbf{E} + K_t \int \mathbf{E} dt),
\]

where \( \dot{\mathbf{r}}^* = [\dot{\mathbf{x}}_1^*, ..., \dot{\mathbf{x}}_m^*]^T \in \mathbb{R}^{2m \times 1} \), and \( \mathbf{J}^T_t \in \mathbb{R}^{6m \times 2m} \) is the pseudo-inverse of the image Jacobian.

B. Pseudo-inverse image Jacobian Matrix Adaptation

By utilizing the traditional linear controller design formula, we derived the control equation using the pseudo-inverse of the image Jacobian as:

\[
\mathbf{u}(t) = \dot{\mathbf{r}}(t)
\]

\[
= \mathbf{J}^T_t (\dot{\mathbf{r}}^*(t - 1) + K_p \mathbf{E}(t - 1) + K_t \int E(t - 1) dt).
\]

The tracking control problem in the task space consists of the tracking error of the desired feature point trajectories \( \mathbf{x}_i^*(i = 1, ..., m) \) and the desired velocity of the feature points...
\( \mathbf{X}^*_i \) \((i = 1, \ldots, m)\) in the image plane. We rewrote (7) as follows:

\[
u(t) = J^*_i(t) \mathbf{M}(t - 1),\]

with \( \mathbf{M}(t - 1) = \mathbf{F}'(t - 1) + K_p \mathbf{E}(t - 1) + K_i \int \mathbf{E}(t - 1) dt \).

Although the kinematics and dynamics of the commercial industry robot are constant and precise, the hand-eye parameters vary due to the long operation duration. Thus, a pseudo-inverse Jacobian adaptation method was proposed.

Inspired by work [17], the adaptation of the pseudo-inverse image Jacobian matrix \( \mathbf{J}^*_i \) was designed as follows:

\[
\mathbf{J}^*_i(t) = \mathbf{J}^*_{i0}(t) + \frac{\alpha \mathbf{u}(t - 1) \mathbf{M}^*(t - 1)}{c^2 + \mathbf{M}(t - 1) \mathbf{M}^*(t - 1)},
\]

where \( \alpha, c \in (0, 1) \), \( t \) denotes the current state, and \( (t - 1) \) denotes the previous state, \( \mathbf{I} \) is a unit diagonal matrix, \( \mathbf{J}^*_{i0} \) is the initial pseudo-inverse image Jacobian matrix.

Once the velocity of the robot end-effector was obtained using Eq. 7, the desired position of the robot was computed as:

\[
r(t) = \mathbf{r}(t - 1) + \mathbf{r}(t) T,
\]

where \( T \) is the command output cycle of the robot controller.

IV. TIME-DELAY COMPENSATOR USING RBFNNs

For a visual-based robot control system, the time delays in command transmission via TCP/IP affect the robot’s cycle times. We generally have no direct access to the time delays in the robot system, that is, there is no mathematical model to describe the time delays. To solve the problem, an NN-based compensator was employed to decrease the cycle times for tracking and manipulating a moving object.

As described in the following section, an RBFNN-based time-delay compensator was employed to compensate for the time delays to ensure the tracking of a moving object.

A. RBFNN

NN control methods have received considerable attention for use in robot control and unknown robot dynamic models [18]. RBFNNs have a simple architecture that is mathematically tractable and has the merits of a fast-learning speed and good approximation capabilities [19]. Thus, we selected the RBFNNs for predicting the time delay and other disturbances in visual servo control systems. The input signals of RBFNNs are \( \mathbf{x} = [x_1, x_2, \ldots, x_n] \). In this work, we used the positions of the end-effector in Cartesian coordinate system and image feature in image coordinate system, respectively, as the input to the NNs, that is \( \mathbf{x} = [x_t, y_t, p_x, p_y] \). And the output of the network was the position error of the image feature in image coordinate system at the next moment, that is \( \mathbf{y} = [e_{xt+1}, e_{yt+1}] \).

The vector in the hidden layer of the radial basis function (RBF) is \( \mathbf{h} = [h_j]^T \), where \( h_j \) is generally determined by a Gaussian function:

\[
h_j(x) = \exp\left(-\frac{\|x - c_j\|^2}{2b_j^2}\right),
\]

where \( c_j \) is the center vector of the Gaussian function of neural net \( j \), and \( b_j \) represents the width of the \( j \)th RBF.

The output of the RBFNNs can be rewritten using the following vector:

\[
y_m(x) = \sum_{j=1}^{m} w_j h_j(x),
\]

where \( w_j \) is the weight connecting the \( j \)th hidden node to the output.

The RBFNN-based compensator is represented as [20]:

\[
\mathbf{y}(t) = W^\top \mathbf{h}(x),
\]

where \( \mathbf{y}(t) \in \mathbb{R}^{m \times 1} \) is the time-delay compensation by the RBFNNs.

B. RBFNN-based time-delay compensator

Generally, a delay can be effectively handled by applying the Smith predictor [28], if the information on its delay, at least, is known and constant [24]. However, the time delays in command transmission in a robot system are difficult to determine and apply. We employed RBFNNs to compensate for the time delay of the robot system. As described in the next section, the validity of the time-delay compensator was illustrated by tracking a moving object and comparing the results to those of the traditional image Jacobian matrix method without compensation.

We provided the IBVS-TDC controller, as shown in Fig. 2, where \( X^* \) is the desired position in the image plane, \( X \) is the position measured by the camera, \( e^{-\tau} \) is the time delay between the controller and robot actuator, \( \mathbf{u} \) and \( \mathbf{y} \) are the outputs of the robot controller and compensator, respectively, \( \lambda \) is the weight, \( P \) denotes the position of the robot end-effector, and \( \mathbf{J}^*_i \) is the pseudo-inverse image Jacobian matrix. The control system is continuous. The output of the image Jacobian matrix controller is a delay output, which could induce a disturbance in the robot system. The time-delay compensator estimated the disturbance and added it to the feedback of the robot’s position for compensation. Thus, we obtained the equation of IBVS-TDC as follows:

\[
\hat{u}(t) = J^*_i(t) \left(F^*(t - 1) + K_p E(t - 1) + \lambda y(t - 1)\right) + K_i \int (E(t - 1) + \lambda y(t - 1)) dt,
\]

We employed RBFNNs as a compensator because of its...
simple structure and fast learning speed. We used the positions of the end-effector and image feature in the Cartesian and image spaces, respectively, at the current moment as the input to the NNs and the output of the networks was the position error of the image feature in the image space at the next moment. Thus, the robot’s movement depended on the image error at the current moment as well as the estimated image error at the next moment. The prediction of the image error was considered a time-delay compensator to reduce the tracking error and improve the robot manipulation performance. We trained the network using supervised learning by collecting data and then used the trained model for online compensation. The training of the network was similar to other traditional types of training. Some other network structures could also be used as the compensator, as long as the data fit is optimal. In this study, a feedback compensator for tracking a moving object was used, where the parameters of the network were trained to fit the time-varying delays induced by command transmission, image processing, and motor driving. However, the compensators in [14] are feedforward compensators, which are generally used in the tracking of a predefined trajectory.

C. Stability analysis

**Theorem.** Consider the visual servoing controller described in Eq. 14 with the RBFNN-based time-delay compensator in Eq. 13, adaptive Pseudo-inverse image Jacobian Matrix laws in Eq. 9, the robot manipulator can track the moving object and the closed-loop system can achieve asymptotic stability.

**Proof:** We rewrite the error function as:

$$\dot{E} = -K_p E - \lambda \dot{y} - \lambda K_p y,$$  

(15)

A Lyapunov function $V$ was designed:

$$V = \frac{1}{2} E^T E + \frac{\lambda K_p}{2\eta} \dot{h}^T h (W^T h),$$

(16)

Thus, the time derivative of the Lyapunov function is:

$$\dot{V} = \frac{\dot{E}^T E}{2} + \frac{\dot{E}^T E}{2} = K_p E^T E + \lambda K_p \dot{h}^T W h + \frac{\lambda K_p}{2\eta} \dot{h}^T W h,$$

(17)

The adaptive rate of weight was proposed as:

$$W = \eta h E^T, \quad W^T = \eta E h^T, \quad \eta > 0,$$

(18)

and (18) were substituted into (17)

$$\dot{V} = -K_p E^T E - \lambda \eta \dot{h}^T h E^T E < 0.$$

(19)

Therefore, the Lyapunov function $V$ will continuously decrease until $E = 0$. Then, the visual servoing system with the proposed controller will be stable.

D. Simulation studies

In this subsection, the influence of the time delay between the master controller and the robot controller was discussed. A time delay will cause instability in the tracking system using the normal image Jacobian controller. We simplified the transfer function of the robot system with a time-variant delay of $G_o(s) = \frac{2^k}{s^2 + 5s + c} e^{-3k \lambda}$, where $k \in (0, 1)$ is a time-variant variable. We set $K_p = 100$ and $K_i = 0.01$, set $\alpha = 0.8$ and $c = 0.35$. These parameters were set identically in the following five tracking ablation experiments. We set $\lambda = 1.2$ in only RBFNN-based time-delay compensator experiment and set $\lambda = 0.4$ in the proposed IBVS-TDC to get the best tracking results respectively. Figure 3 showed the results of the tracking ablation experiments using the traditional visual-based controller (Tra-IBVS), the visual-based controller with the Smith predictor (IBVS-SP), the visual-based controller with pseudo-inverse image Jacobian adaption (IBVS-AD), the visual-based controller with only the RBFNN-based time-delay compensator (IBVS-RBF) and the proposed IBVS-TDC method, respectively. Note that the dashed lines were the desired trajectories and the solid lines were the tracking trajectories in the image coordinate system. The mean tracking errors were shown in Table I. Our proposed method was able to follow the desired trajectory effectively with the smallest mean tracking error.

$$u^* = 0.05 t \cdot \cos (t/200) + 0.05 t \cdot \sin (t/200)$$

$$v^* = 0.05 t \cdot \cos(t/200) - 0.05 t \cdot \sin(t/200)$$

**Fig. 3:** (a) shows the results of the Tra-IBVS; (b) shows the result of the IBVS-SP; (c) shows the result of the IBVS-AD; (d) shows the results of the IBVS-RBF; (e) shows the results of the proposed IBVS-TDC method. Note that the left picture shows the tracking results and the middle
picture shows the tracking errors in x- and y-directions, and the right picture shows the tracking results in the image plane.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Mean(x)</th>
<th>Mean(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tra-IBVS</td>
<td>3.99</td>
<td>5.60</td>
</tr>
<tr>
<td>IBVS-SP</td>
<td>1.64</td>
<td>2.39</td>
</tr>
<tr>
<td>IBVS-AD</td>
<td>0.63</td>
<td>0.84</td>
</tr>
<tr>
<td>IBVS-RBF</td>
<td>0.82</td>
<td>1.02</td>
</tr>
<tr>
<td>IBVS-TDC</td>
<td>0.46</td>
<td>0.68</td>
</tr>
</tbody>
</table>

V. SIMULATIONS AND EXPERIMENTS

We initially conducted two simulation experiments on the manipulation of a moving object to illustrate the validity of the proposed method. We also used a Universal Robot (UR) for tracking a moving hole and then inserting a peg into the hole in a real experiment.

A. Tracking and manipulating simulation

We conducted simulation experiments in V-Rep [21] using Bullet 2.78 for dynamics and V-Rep’s internal inverse kinematics module for robot motion planning. The simulation environment included a UR5 robot, a camera mounted on the robot end-effector, and a moving hole on the ground. The camera identified and transferred the position of the target hole to the UR5 robot. The UR5 robot held the peg and followed the target, guided by the visual information, trying to keep the target in the center of the image. Each test was run for \( n = 80 \) tracked steps in the simulation. We ignored the image noise in the simulations. We set \( K_p = 8 \) and \( K_i = 0.002 \), set \( \alpha = 0.25 \) and \( c = 0.03 \). These parameters were set identically in the following experiments. We set \( \lambda = 0.5 \) in IBVS-RBF and set \( \lambda = 0.34 \) in IBVS-TDC to get the best tracking results respectively. We conducted four sets of experiments using the Tra-IBVS, IBVS-AD, IBVS-RBF, and proposed IBVS-TDC controllers. For the first simulation, the hole moved along a circular trajectory to facilitate the observation of the tracking performance. The execution of the robot in the simulation was different from that of a physical robot, that is, the time required for the robot to execute the control signal was very short. Note that in this study, the input to the RBFNNs was a 4-dimensional vector, where one element was the position \((x, y)\) of the feature point in the image coordinate system and the other was the position \((P_x, P_y)\) of the robot end-effector.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Circular trajectory</th>
<th>Arbitrary trajectory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean(x)</td>
<td>Mean(y)</td>
<td>Mean(x)</td>
</tr>
<tr>
<td>Tra-IBVS</td>
<td>1.90</td>
<td>1.94</td>
</tr>
<tr>
<td>IBVS-AD</td>
<td>1.32</td>
<td>1.36</td>
</tr>
<tr>
<td>IBVS-RBF</td>
<td>1.07</td>
<td>1.25</td>
</tr>
<tr>
<td>IBVS-TDC</td>
<td>0.69</td>
<td>0.73</td>
</tr>
</tbody>
</table>

In the simulations, we described the tracking performance in the world coordinate system as the positions of the target and the robot end-effector can be easily obtained in the world coordinate system from V-Rep. The tracking performances of the IBVS-AD in Fig. 4(b) and IBVS-RBF in Fig. 4(c), outperformed the Tra-IBVS controller in Fig. 4(a). The IBVS-TDC controller in Fig. 4(d) achieved the best performance. The mean tracking errors of different methods for tracking circular trajectories are shown in Table II. Note that the errors in Table II are described in the world coordinate system.

![Tracking results](Image)

Fig. 4. Tracking a target moving along a circular trajectory. The dashed lines and solid lines represent the position of the moving hole and the robot end-effector in the x- and y- directions, respectively. (a) shows the results of the Tra-IBVS; (b) shows the results of the IBVS-AD; (c) shows the results of the IBVS-RBF; (d) shows the results of the proposed IBVS-TDC controller. Note that the solid lines were the desired trajectories and the dashed lines were the tracking trajectories in the world coordinate system.

We used the proposed IBVS-TDC method to track a moving hole with an arbitrary trajectory designed using the PATH module in V-Rep. The tracking position curve of the world coordinate system is shown in Fig. 5(a). The mean tracking errors of various methods for tracking irregular trajectories were shown in Table II.

![Tracking results](Image)

Fig. 5. Tracking a target moving along an arbitrary trajectory. (a) shows the results of the Tra-IBVS; (b) shows the results of the IBVS-AD; (c) shows the results of the IBVS-RBF; (d) shows the results of the proposed IBVS-TDC controller.

The process of tracking and inserting an object into a moving hole was shown in Fig. 6. The robot manipulator followed the
moving hole and inserted the peg into it when the position error between it and the end-effector was less than a pre-defined threshold value.

![Image](image_url)

(a) (b)

**Table III Mean Tracking Errors at Various Moving Speeds**

<table>
<thead>
<tr>
<th>Speed (m/s)</th>
<th>Method</th>
<th>Tra-IBVS</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean</td>
<td>Max</td>
</tr>
<tr>
<td>0.1</td>
<td></td>
<td>0.7227</td>
<td>0.6110</td>
</tr>
<tr>
<td>0.2</td>
<td></td>
<td>1.0745</td>
<td>1.5424</td>
</tr>
<tr>
<td>0.3</td>
<td></td>
<td>1.8670</td>
<td>2.1970</td>
</tr>
<tr>
<td>0.5</td>
<td></td>
<td>2.9931</td>
<td>3.7970</td>
</tr>
<tr>
<td>0.8</td>
<td></td>
<td>4.6358</td>
<td>6.3741</td>
</tr>
<tr>
<td>1.0</td>
<td></td>
<td>6.0432</td>
<td>8.1041</td>
</tr>
</tbody>
</table>

**B. Tracking and inserting a peg into a moving hole in the real world**

In another experiment, we tracked and inserted a peg into a moving hole using a real UR5 robot. And a UR3 robot was used to hold the hole and propel it. An overview of the system was shown in Fig. 7. The UR5 and UR3 were both 6-DoF Universal robots with position repeatability of 0.03 mm. The Robotiq grippers mounted on the end-effector of UR5 and UR3 were used to pick up the peg and hole respectively. In addition, the digital camera (MER-041-436U3M) with a resolution of 720 x 540 pixels was used to capture the target image. Note that the UR5 robot did not know the trajectory of the hole. Thus, we employed the proposed method to enable the UR5 robot to track the hole, following which we inserted the peg into the moving hole. For the image features, we used the AprilTag [22], which was easily and robustly identified, to represent the target hole position in the image. When the center point of AprilTag appeared in the center of the image, the peg was also aligned with the center of the hole. We set $K_p = 10$ and $K_i = 0.005$, set $\alpha = 0.35$ and $c = 0.1$. These parameters were set identically in the following experiments. We set $\lambda = 0.6$ in IBVS-RBF and set $\lambda = 0.5$ in IBVS-TDC to get the best tracking results respectively.

Here, the diameters of the peg and holes were 40 mm and 50 mm, respectively. Thus, the tolerances between the peg and the hole were approximately 10 mm. As the positions of the peg and the hole in the world coordinate system were not available, it is impossible to compute the position errors between the two parts. Therefore, we determined the tracking performance by calculating the error between the position of the target (AprilTag's center point) in the image and the image center point, which was initialized by (0,0).

**Fig. 6.** Process of tracking a moving hole, (a) shows the initial state, (b) shows the robot successfully tracking the moving hole and inserting the peg into it.

We tested the mean and maximum tracking errors of the Tra-IBVS and the proposed IBVS-TDC method at different speeds of the moving target, as shown in Table III. Note that the errors in Table III were in the world coordinate system. As shown in Table III, the IBVS-TDC method reduced both the average and maximum tracking errors. However, although the tracking speed of 1.0 m/s was achievable in the simulation environment, it was hard to realize in the physical environment, as a hole of that size can easily escape from the small view field of the camera at the initial stage.

**Fig. 7.** Real-world setup of tracking and insertion of a peg into a moving hole.

In the real experiment, the tracking performance was described in the image coordinate system. In the first experiment, the hole was designed to move along a circular trajectory, with the UR3 robot moving the peg, and tracking the hole with the UR5 robot. Take 630 points evenly on the circumference as the moving position of the hole. The tracking results of Tra-IBVS, IBVS-AD, and IBVS-TDC in the $x$ - direction and $y$ - direction are shown in Fig. 8(a) and Fig. 8(b) respectively. Taking the $x$ direction as an example, the maximum tracking error of the Tra-IBVS was 0.462 mm and the average tracking error was 0.270 mm. The proposed IBVS-TDC method reduced the maximum and average tracking errors (the distance between the feature point position and target position in the image coordinate system) to 0.194 mm and 0.084 mm, respectively, a significant reduction in both cases.
In the second experiment, the hole moved in a sinusoidal curve. The trajectory was formed by splicing two sinusoids. The mathematical expression of the first segment is \( x = 0.7 \cdot \sin(12\pi \cdot y) \); the value range of \( y \) was \( 0 \sim 0.25 \) \( m \). The other segment was symmetrical to it. Take 600 points evenly on one segment as the moving position of the hole. The tracking results are presented in Fig. 9(a-b). In the \( x \) -direction, the maximum and average errors of the Tra-IBVS were 0.642 mm and 0.374 mm, respectively. The proposed IBVS-TDC method reduced the maximum and average tracking errors to 0.271 mm and 0.125 mm, respectively. In the \( y \) -direction, the maximum and average tracking errors of the Tra-IBVS were 0.437 mm and 0.083 mm, respectively. The proposed IBVS-TDC method reduced the maximum tracking error to 0.251 mm and the average tracking error to 0.036 mm.

The steps of the tracking-and-inserting process were presented in Fig. 10. The entire visual servo system consisted of three steps. First, the image was acquired and the feature point positions was calculated (assuming that the time was \( t_1 \)). Next, the feature point position was transmitted to the IBVS-TDC controller and the control law was calculated (assuming that the time was \( t_2 \)). Finally, the robot received the control law by TCP/IP (assuming that the time was \( t_3 \), which is hard to be measured). To enable the robot to track moving targets more smoothly, we performed the three steps simultaneously using three parallel threads, so that the time of one running cycle is \( \max(t_1, t_2) + t_3 \). In the actual system we were running as an example, the time to execute a cycle was about 50 ms + \( t_3 \).

The trajectory of the hole in the \( x \) -direction, the maximum and average tracking errors of the Tra-IBVS were 0.642 mm and 0.374 mm, respectively. The proposed IBVS-TDC method reduced the maximum and average tracking errors to 0.271 mm and 0.125 mm, respectively. In the \( y \) -direction, the maximum and average tracking errors of the Tra-IBVS were 0.437 mm and 0.083 mm, respectively. The proposed IBVS-TDC method reduced the maximum tracking error to 0.251 mm and the average tracking error to 0.036 mm.

The trajectory of the hole in the \( x \) -direction, the maximum and average tracking errors of the Tra-IBVS were 0.642 mm and 0.374 mm, respectively. The proposed IBVS-TDC method reduced the maximum and average tracking errors to 0.271 mm and 0.125 mm, respectively. In the \( y \) -direction, the maximum and average tracking errors of the Tra-IBVS were 0.437 mm and 0.083 mm, respectively. The proposed IBVS-TDC method reduced the maximum tracking error to 0.251 mm and the average tracking error to 0.036 mm.

The steps of the tracking-and-inserting process were presented in Fig. 10. The entire visual servo system consisted of three steps. First, the image was acquired and the feature point positions was calculated (assuming that the time was \( t_1 \)). Next, the feature point position was transmitted to the IBVS-TDC controller and the control law was calculated (assuming that the time was \( t_2 \)). Finally, the robot received the control law by TCP/IP (assuming that the time was \( t_3 \), which is hard to be measured). To enable the robot to track moving targets more smoothly, we performed the three steps simultaneously using three parallel threads, so that the time of one running cycle is \( \max(t_1, t_2) + t_3 \). In the actual system we were running as an example, the time to execute a cycle was about 50 ms + \( t_3 \).

The steps of the tracking-and-inserting process were presented in Fig. 10. The entire visual servo system consisted of three steps. First, the image was acquired and the feature point positions was calculated (assuming that the time was \( t_1 \)). Next, the feature point position was transmitted to the IBVS-TDC controller and the control law was calculated (assuming that the time was \( t_2 \)). Finally, the robot received the control law by TCP/IP (assuming that the time was \( t_3 \), which is hard to be measured). To enable the robot to track moving targets more smoothly, we performed the three steps simultaneously using three parallel threads, so that the time of one running cycle is \( \max(t_1, t_2) + t_3 \). In the actual system we were running as an example, the time to execute a cycle was about 50 ms + \( t_3 \).

The steps of the tracking-and-inserting process were presented in Fig. 10. The entire visual servo system consisted of three steps. First, the image was acquired and the feature point positions was calculated (assuming that the time was \( t_1 \)). Next, the feature point position was transmitted to the IBVS-TDC controller and the control law was calculated (assuming that the time was \( t_2 \)). Finally, the robot received the control law by TCP/IP (assuming that the time was \( t_3 \), which is hard to be measured). To enable the robot to track moving targets more smoothly, we performed the three steps simultaneously using three parallel threads, so that the time of one running cycle is \( \max(t_1, t_2) + t_3 \). In the actual system we were running as an example, the time to execute a cycle was about 50 ms + \( t_3 \).
The process of inserting the peg into the hole in 3D space is shown in Fig. 12, where the speed of the hole was approximately 8 cm/s.

![Fig. 12. Process of tracking and inserting the peg into the hole in 3D space.](image)

**VI. CONCLUSION**

In this paper, we present an image-based visual tracking method for the robot manipulation of a moving object. The method consists of two parts: a pseudo-inverse image Jacobian matrix adaptation and a time-delay compensator. The image Jacobian matrix adaption deals with the uncertainties in the robot image Jacobian matrix. We describe the robot visual servo system using an equivalent linearization model; thus, the linear model yields an adaptation of the image Jacobian matrix. The time-delay compensator solves the delay in the robot, which is useful in tracking a moving object. The image Jacobian adaptation and time-delay compensator are integrated using a sliding-mode controller, which improves the performance of the robot tracking system in terms of precision and speed.

We conducted experiments using a virtual robot in V-Rep and a UR robot in a real environment. We found that in the image coordinate system, the proposed IBVS-TDC reduced the maximum and average tracking errors from 0.642 mm and 0.374 mm of the traditional visual-based tracking method to 0.271 mm and 0.125 mm, respectively. We tracked and inserted a peg into a moving hole using an eye-in-hand robot (the distance between the peg and the hole was 1 cm). In our future work, we try to track a moving object even mitting a significant delay in the robot control loop, where we will employ a wide-angle camera to keep the object within the camera view. And, we will use a telephoto camera to precisely localize the target for robot manipulation. Furthermore, we will decrease the cycle time by using a high frame rate camera, and by accelerating the image processing algorithm.

**APPENDIX**

We define an objective function as equation (A1) and estimate the pseudo-inverse image Jacobian matrix $\hat{J}_i^T$ by its optimal solution

$$f = \frac{1}{2} \| \hat{\mathbf{u}}(t) - \alpha \mathbf{u}(t-1) - \mathbf{u}_0(t) \|^2 + \frac{c}{2} \| \hat{J}_i^T(t) - I_{\mathbf{r}0}(t) \|^2,$$  (A1)

where $\alpha, c \in (0,1)$ are weighting constants.

$$\hat{\mathbf{u}}(t) = \hat{J}_i^T(t)M(t-1) \quad \text{and} \quad \mathbf{u}_0(t) = \hat{J}_i^T(t_0)M(t-1).$$  (A2)

We take the partial derivative with respect to $\hat{J}_i^T$, yields

$$\frac{\partial f}{\partial \hat{J}_i^T} = \frac{1}{2} \frac{\partial \| \hat{\mathbf{u}}(t) - \alpha \mathbf{u}(t-1) - \mathbf{u}_0(t) \|^2}{\partial \hat{J}_i^T} + \frac{c}{2} \frac{\partial \| \hat{J}_i^T(t) - I_{\mathbf{r}0}(t) \|^2}{\partial \hat{J}_i^T}$$

i.e.,

$$\frac{\partial f}{\partial \hat{J}_i^T} = (\hat{\mathbf{u}}(t) - \alpha \mathbf{u}(t-1) - \mathbf{u}_0(t)) \frac{\partial \hat{\mathbf{u}}(t)}{\partial \hat{J}_i^T} + c \left( \hat{J}_i^T(t) - I_{\mathbf{r}0}(t) \right) \frac{\partial \hat{J}_i^T(t)}{\partial \hat{J}_i^T}.$$

Let the partial derivative $\frac{\partial f}{\partial \hat{J}_i^T}$ equal to 0, gives

$$\left( \hat{\mathbf{u}}(t) - \alpha \mathbf{u}(t-1) - \mathbf{u}_0(t) \right)M^T(t-1) + c \left( \hat{J}_i^T(t) - I_{\mathbf{r}0}(t) \right) = 0.$$  (A3)

Substitute (A2) into (A3), gives

$$\left( \hat{J}_i^T(t) - I_{\mathbf{r}0}(t) \right) \left( cl + M(t-1)M^T(t-1) \right) = 0$$

Thus,

$$\hat{J}_i^T(t) = I_{\mathbf{r}0}(t) + \frac{\alpha \mathbf{u}(t-1)M^T(t-1)}{cl + M(t-1)M^T(t-1)}.$$  (9)

where $t$ denotes the current state, $(t-1)$ denotes the previous state, $I$ is a unit diagonal matrix, $J_{\mathbf{r}0}$ is the initial pseudo-inverse image Jacobian matrix.

**REFERENCES**


J.H. Su received the B.Eng. degree in electronic and information engineering from Beijing Jiaotong University, Beijing, China, in 1999, the M.Eng. degree in electronic and information engineering from the Beijing Jiaotong University, Beijing, in 2004, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2009. He is currently a Professor with the Institute of Automation, Chinese Academy of Sciences. He is the author of a book, more than 60 articles, and 28 inventions. His current research interests include robot manipulation, robot learning, and control.

L.L. Wang received a bachelor’s degree in engineering from the College of Communication Engineering of Jilin University, Changchun, China, in 2020. She is currently studying for a master’s degree at the Institute of Automation, University of Chinese Academy of Sciences.
C.K. Liu received a B.Sc. degree in Measurement, Control Technology, and Instrumentations from the University of Electronic Science and Technology of China, Chengdu, China, in 2006, and a Ph.D. degree in robotics and artificial intelligence from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2011. He is a Senior Engineer, and the director of the Space Teleoperation Innovation Team, with the Beijing Aerospace Control Center, Beijing, China. His current research interests include manipulation planning, vision-guided robotics, multi-agent systems, and intelligence fusion.

Hong Qiao (Fellow, IEEE) received the B.Eng. degree in hydraulics and control and the M.Eng. degree in robotics from Xi’an Jiaotong University, Xi’an, China, in 1986 and 1989, respectively, and the Ph.D. degree in robotics control from De Montfort University, Leicester, U.K., in 1995. She was an Assistant Professor with the City University of Hong Kong, Hong Kong, and a Lecturer with the University of Manchester, Manchester, U.K., from 1997 to 2004. She is currently a Professor with the Institute of Automation, Chinese Academy of Sciences (CAS), Beijing, China. She is also a Professor with the CAS Center for Excellence in Brain Science and Intelligence Technology, Shanghai, China. Her current research interests include robotics, machine learning, and pattern recognition. Prof. Qiao is a member of the Administrative Committee of the IEEE Robotics and Automation Society. She is the Editor-in-Chief of Assembly Automation (SCI), an Associate Editor for IEEE Transactions on Cybernetics.