Adaptive Neural Network-based Visual Servoing with Integral Sliding Mode Control for Manipulator

Haibin Zeng[1], Zhihui LU[2], Yueyong Lv[1], Jiaming Qi[1]

1. Harbin Institute of Technology, Harbin 150001, China
E-mail: zenghaibin_hit@163.com

2. Aerospace System Engineering Shanghai, Shanghai 201109, China
E-mail: 25110690@qq.com

Abstract: The Calibration-free visual servoing control is challenging, since it is difficult to estimate the relationship between the motion of joint and the motion of image features. Previous studies often approximate the relation in purely online or offline ways. A scheme for robot arm manipulation with both online and offline learning is proposed in this paper. The relation is formulated in a local linear format with Jacobian matrix, which is approximated by radial-basis function network (RBFN). Primitively, the RBFN is trained offline to form a relatively appropriate estimation of the matrix, which is the beginning of the online step. Then, an online modification of the RBFN is executed to compensate the error caused by changes of camera's position and pose or insufficient training. The simulation experiments show that the proposed scheme can provide a reliable offline trained model and can adapt well to the changes of camera's position and pose due to the online update law.

Key Words: RBFN, uncalibrated visual servoing, online update, end-effector positioning, Jacobian matrix

1 Introduction

In traditional applications, the robotic arm visual servoing system relies on a lot of calibration work, including: camera internal parameter calibration, robot kinematics and dynamic model parameter calibration [1-3]. Calibration has some obvious shortcomings, such as: it cannot be calibrated under high temperature and strong radiation; the system structure changes, the calibration parameters will change, and it needs to be re-calibrated regularly; the calibration process is complicated, requiring professionals and equipment, and the calibration cost is high [4,5]. These calibration tasks limit the development of intelligent robots, so researchers have begun to delve into uncalibrated visual servoing, such as estimating Jacobian matrices purely online using Kalman filtering, using neural networks to fit hand-eye relationships offline, etc.

The uncalibrated visual servoing control for robot arm refers to a technology that uses visual feedback signals to construct a closed-loop control system guiding the robot to complete tasks related to end-effector's position or pose without hand-eye relationship or pre-calibrating [2]. The demand for Stability and Practicality of robot arm end-effector position manipulation arouses a significant amount of research efforts. Robot visual servoing without calibration is a highly coupled and an uncertain complex nonlinear system [6]. In order to overcome the errors in camera and robot model caused by environmental change, the robustness of control system is highly concerned.

Uncalibrated Visual Servoing system can be divided into image-based uncalibrated visual servoing (IBUVS) and position-based uncalibrated visual servoing (PBUVS) according to whether the feedback signal is pure image information or estimated position information. Compared with PBUVS, IBUVS does not require parameters of camera and robot arm, making the hand-eye system more flexible and intelligent [7,8].

As for image-based uncalibrated visual servoing, the estimation of the relationship between joint space motion and change of image characteristic is the key factors to ensure stability and robustness. Some classical methods estimate the relationship in purely online or offline way, which cannot find a better balance between system performance and system robustness [9]. These online estimated models are less accurate because only a small amount of local data can be utilized [10,11], and those offline models are formed based on data collected in specific situations, which does not necessarily apply in other situations [12].

In this paper, a scheme with both online and offline method for hand-eye relationship estimation is proposed. First, the hand-eye relationship is formulated in a linear format by Jacobian matrix. Then, an offline trained of radial-basis function network fitting the Jacobian matrix is carried out with data collected from UR5 joint motion based on the premise that the camera is at a specific position and pose. In order to improve the robustness of the system against camera pose changes and the insufficient performance of offline models, we propose an online update algorithm for the RBFN model trained offline. Using the Lyapunov function, we can prove the stability of the system. Finally, a series of simulation experiments are carried out to verify the superiority of the offline model used in this paper compared with the purely online Kalman filter estimation and the improvement of the system performance by the online update law for RBFN.

2 Visual Servoing Control Systems

Compared to position-based visual servoing control systems, image-based visual servoing control systems avoid the need for real-time estimation of the target's position in

[*]This work is supported by National Key Research and Development Program of China, 2020YFB1506700, Funded by Science and Technology on Space Intelligent Control Laboratory, No. HTKJ2020KL502014, and National Natural Science Foundation (NNSF) of China under Grant 12150008, 61973100, 61876050.

the base coordinate system, reducing computational processes and system latency [8]; in addition, its control process does not include the robot and camera model parameters, so it has better robustness and a more comprehensive range of applications [9]. In the image-based visual servoing control scheme proposed in this paper (see Fig. 1), sensors such as depth cameras are used to obtain the current position of the end-effector of the robot arm in the camera coordinate system and use this as the present image feature, which is compared with the desired image feature to obtain the characteristic error. The characteristic error is converted into the value of the robot arm joint motion by correspondence to drive the end-effector of the robot arm to the desired position.

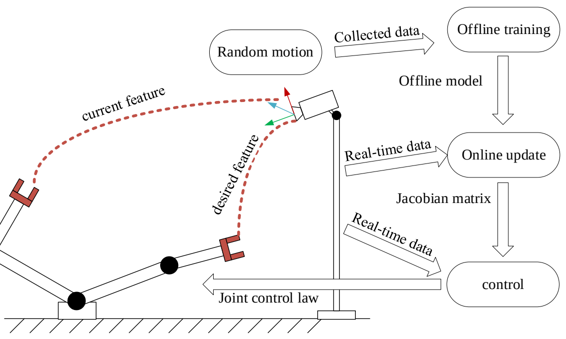

Fig. 1: Overview of the scheme for end-effector position manipulation

As shown in Fig. 1, the image-based uncalibrated visual servoing scheme for end-effector position in this paper, is roughly divided into two parts: offline training and online update. In the first part, random motion in joint space is carried out, so that we get enormous data about joint motion and image feature captured by depth camera. These collected data is used for offline training. The trained RBFN can provide relatively accurate Jacobian matrix estimation. Once the actual operation starts, the system goes to online mode, of which the offline trained model is a start point. While manipulation, real-time data helps the online algorithm update the RBFN, making the RBFN resistant to environmental and model changes. The controller will continue to drive the end-effector to the desired position until the feature error converges to zero. The work of this paper is mainly the design of offline training, online update law and control law shown in Fig. 1.

3 Methodology

The hand-eye relationship can be expressed as a differential formulation, that is, the velocity of joint space and velocity of characteristics captured by camera. The velocity of the image features can be locally linearly related to the velocity vector of the joint space using a Jacobian matrix.

$$\dot{\boldsymbol{x}} = \boldsymbol{J} \cdot \dot{\boldsymbol{r}} \tag{1}$$

$\boldsymbol{x}$ denotes the 3D coordinates measured by the depth camera, in other words, the image features. $\boldsymbol{r}$ denotes the joint space vector of the robot arm. $\boldsymbol{J}$ denotes the Jacobian matrix estimated in real-time.

The main problem of image-based uncalibrated visual servoing control system lies in the real-time estimation of Jacobian matrix and the design of the joint space controller. In this paper, we use a radial-basis-function neural network (RBFN) to map the current joint state to the local linear manipulation model. In the offline phase, the RBFN is trained on the data collected from UR5 model. Also, the module is updated online during operation, which reduces the impact of insufficient training or error of training data, thus increasing the system's robustness.

3.1 RBFN Model Offline Training

The RBFN is a forward network with global approximation capability and strong non-linear mapping ability. The training samples used in this paper are collected based on the UR5 robotic arm model. The design of the RBFN depends on the nature of the fitting target.

There is a significant difference in the impact of six individual joint angles on the end-effector features due to the characteristics of the robotic arm model. The Jacobian matrix can be written as

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial \boldsymbol{x}_1}{\partial \boldsymbol{r}_1} & \cdots & \frac{\partial \boldsymbol{x}_1}{\partial \boldsymbol{r}_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{x}_3}{\partial \boldsymbol{r}_1} & \cdots & \frac{\partial \boldsymbol{x}_3}{\partial \boldsymbol{r}_6} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_1 & \boldsymbol{J}_2 & \boldsymbol{J}_3 & \boldsymbol{J}_4 & \boldsymbol{J}_5 & \boldsymbol{J}_6 \end{bmatrix} \tag{2}$$

It is split into six column vectors, which indicate the effect of the motion of the joints on the image feature values respectively. Taking the UR5 robot arm as an example, the six joints correspond to different axis functions. $\boldsymbol{r}_1$ corresponds to the arm body rotation axis, which carries a significant swing in the horizontal direction of the arm and has almost the most significant effect on the end-effector position. While, $\boldsymbol{r}_6$ corresponds to the end-effector rotation axis, which mainly affects the end-effector pose but has no impact on the end-effector position. From the first axis to the sixth axis, the impact of the rotations of the joints on the end-effector position and on the Jacobian matrix show a decreasing trend, the value of the joint angle of the sixth axis even has no impact on the end-effector position.

In order to measure the effect of each joint angle change on the Jacobian matrix, we define $\Delta \boldsymbol{J}_k = \boldsymbol{J}\text{-}\boldsymbol{J}_k$, where $\boldsymbol{J}$ denotes the original Jacobian matrix, and $\boldsymbol{J}_k$ denotes the value of the new Jacobian matrix after the k^{th} joint is increased by 10°. By calculating the Frobenius norm value of the $\Delta \boldsymbol{J}_k$ after the k^{th} joint change, the influence of the k^{th} joint change on the Jacobian matrix can be revealed. In order to measure the effect of each joint angle change on the end-effector position, the 2-norm value of the six columns from $\boldsymbol{J}_1$ to $\boldsymbol{J}_6$ shown in equation(2) is calculated.

Based on the UR5 robotic arm model, a number of states were selected at equal intervals in the 6-dimensional joint space. The mean of the 2-norm value of the six columns from $\boldsymbol{J}_1$ to $\boldsymbol{J}_6$ in these sample states and the mean of the Frobenius norm value of the matrix from $\Delta \boldsymbol{J}_1$ to $\Delta \boldsymbol{J}_6$ in these sample states were found, and the results are shown in Fig. 2 and Fig. 3 respectively, which show a decreasing trend of the impact of the joints rotation on the end-effector position(shown in Fig. 2) and on the Jacobian matrix(shown in Fig. 3).

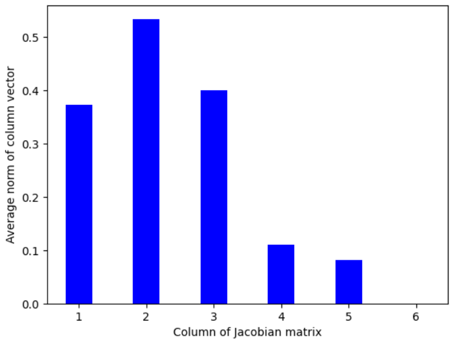

Fig. 2: Mean of 2-norm of $\boldsymbol{J}_1$ to $\boldsymbol{J}_6$

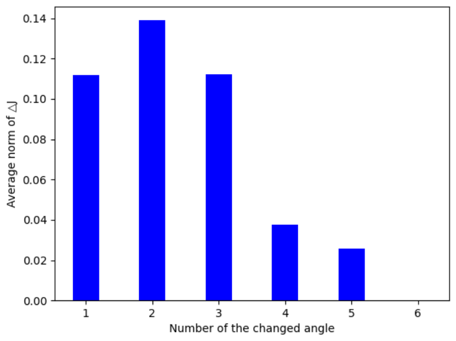

Fig. 3: Mean of Frobenius norm of $\Delta\boldsymbol{J}_1$ to $\Delta\boldsymbol{J}_6$

These two properties are fully taken into account when designing RBFN. Because of the difference in the values of the six columns, six independent RBFN were constructed for the six columns of the Jacobian matrix with different learning rates while training. Since different joints have different effects on the Jacobian matrix, the basis function of RBFN is also modified appropriately as follows.

The RBFN has a three-layer structure: the input layer, the hidden layer, and the output layer [13]. In this paper, the number of nodes of the input layer is 6, consistence with the length of joint space vector; the number of nodes of the hidden layer is 216; the number of nodes of the output layer is 3. The outputs of the six RBF neural networks form the Jacobian matrix.

The mathematical model of the usually used RBFN is:

$$\theta_{it} = e^{-\left(\frac{\|\boldsymbol{r}-\boldsymbol{u}_{it}\|}{\delta_{it}}\right)^2} = e^{-\left(\frac{\sqrt{\sum_{j=1}^{l}(\boldsymbol{r}_j-\boldsymbol{u}_{itj})^2}}{\delta_{it}}\right)^2} = e^{-\left(\sqrt{\sum_{j=1}^{l}\left(\frac{\boldsymbol{r}_j-\boldsymbol{u}_{itj}}{\delta_{it}}\right)^2}\right)^2} \qquad (3)$$

$$\boldsymbol{J}_i = \boldsymbol{W}_i\theta_i(\boldsymbol{r}), \quad i=1,\dots,6 \qquad (4)$$

Where θ_{it} represents the basis function value, and u_{it} represents the center of the basis function, and δ_{it} represents the standard deviation. The center and standard deviation of the basis function can cluster the input samples. When calculating the distance between the sample and the cluster center, the Euclidean distance is usually used [14], as in equation (3). But this practice hides the fact that different joints have different effects on the Jacobian matrix. Assume that the input is a two-dimensional vector $(\boldsymbol{r}_1, \boldsymbol{r}_6)$, and that the $\boldsymbol{r}_1$ term is much more important than the $\boldsymbol{r}_6$ term. If $\boldsymbol{r}_1$ and $\boldsymbol{r}_6$ correspond to the same standard deviation δ_{it}, consistent with the circular curve $\boldsymbol{L}$ in Fig. 4. The points on the circle are considered as equal distances from the center, which is not consistent with the nature of the different importance of the two indicators. The ideal case corresponds to the elliptical curve $\boldsymbol{M}$ in Fig. 4, where the indicator corresponding to the x-axis is more critical than the one corresponding to the y-axis. The issue of curve M corresponds to equation (5), where each dimension of the input sample corresponds to a different standard deviation. The points on the ellipse line are considered to have the same distance from the center.

$$\theta_{it} = e^{-\left(\sqrt{\sum_{j=1}^{l}\left(\frac{\boldsymbol{r}_j-\boldsymbol{u}_{itj}}{\delta_{itj}}\right)^2}\right)^2} \qquad (5)$$

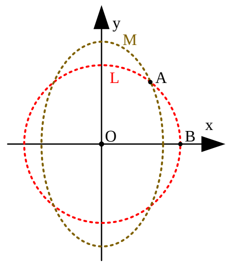

Fig. 4: Consider Weighted Dual Metric Clustering

In the neural network training, we use the PyTorch framework to train the RBFN used to fit the $\boldsymbol{J}_1$ values. The two RBFN models used corresponding to (3) and (5). The mean square error (MSE) loss function value, see Fig. 5, indicates that the weighted RBF neural network basis functions are more consistent with the characteristic that the six joints have different weights on the end-effector position.

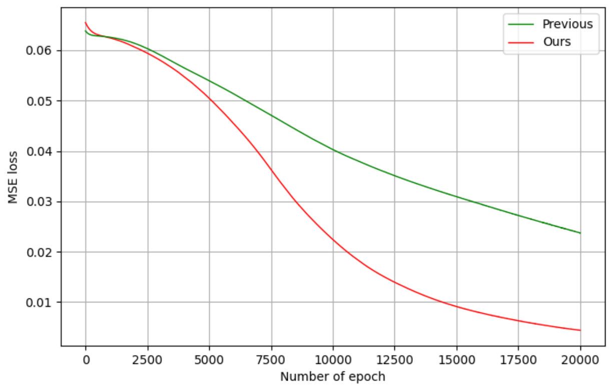

Fig. 5: Training speed of two RBFN model

3.2 Design of Sliding Mode Control with Real-Time Update Law and Stability Proof

The RBFN is trained offline based on the premise that the camera is at a specific position and pose. While in practice, the trained model is not always suitable due to wrong training data or inadequate training. In addition, when using the trained model, the camera model and the transfer matrix from camera to base coordinates have certain errors [15]. To address these issues, this section describes how to calculate the joint angle control law and the neural network model update value in real-time based on the feature error and gives stability proof.

After bringing equation (4) into equation (2), we get the velocity of characteristic.

$$\dot{\boldsymbol{x}} = \boldsymbol{J} \cdot \dot{\boldsymbol{r}} = \begin{bmatrix} \boldsymbol{J}_1 & \cdots & \boldsymbol{J}_6 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{r}}_1 \\ \vdots \\ \dot{\boldsymbol{r}}_6 \end{bmatrix} \tag{6}$$

$$= \sum_{i=1}^{6} \boldsymbol{J}_i \cdot \dot{\boldsymbol{r}}_i = \sum_{i=1}^{6} \boldsymbol{W}_i \cdot \theta_i \cdot \dot{\boldsymbol{r}}_i$$

Here we define the velocity error as

$$\boldsymbol{e} = \dot{\boldsymbol{x}} - \hat{\boldsymbol{J}}\dot{\boldsymbol{r}} = \sum_{i=1}^{6} \boldsymbol{W}_i \theta_i \dot{\boldsymbol{r}}_i - \sum_{i=1}^{6} \hat{\boldsymbol{W}}_i \theta_i \dot{\boldsymbol{r}}_i = \sum_{i=1}^{6} \Delta \boldsymbol{W}_i \theta_i \dot{\boldsymbol{r}}_i \tag{7}$$

Where $\hat{\boldsymbol{W}}_i$ represents the estimated fully connected layer. In addition, $\Delta \boldsymbol{W}_i = \boldsymbol{W}_i - \hat{\boldsymbol{W}}_i$. The RBFN are updated online with the aim, among others, of $\Delta \boldsymbol{W}$ and velocity error convergence to zero.

In this paper, sliding mode control is used to make the image characteristic error converge to zero, and the integral sliding surface is designed as

$$\boldsymbol{s} = c_1 \triangle \boldsymbol{x} + c_2 \int \triangle \boldsymbol{x} dt \tag{8}$$

The initial value of the integral term of sliding surface is set as $\int \triangle \boldsymbol{x} dt = -c_1 / c_2 \triangle \boldsymbol{x}$, the term $\triangle \boldsymbol{x}$ is determined by the initial state of the system, while the initial value of the integral term can be self-designed. By setting the initial value of the integral term, we can ensure that it is on the sliding mode surface at the very beginning, thus enhancing the convergence speed of the system. The joint control law is designed as

$$\dot{\boldsymbol{r}} = -\frac{c_2}{c_1} (\hat{\boldsymbol{J}})^{+} \triangle \boldsymbol{x} \tag{9}$$

where $(\hat{\boldsymbol{J}})^{+}$ is the Moore-Penrose pseudo-inverse of the estimated Jacobian matrix. In addition, $\triangle \boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_d$ where $\boldsymbol{x}_d$ is the desired position vector of the end-effector, and the number of c_1 and c_2 are positive. The online update law of the j^{th} row of $\hat{\boldsymbol{W}}_i$ in the i^{th} RBFN is designed as follows.

$$\dot{\hat{\boldsymbol{W}}}_{ij}^{\mathrm{T}} = \dot{\boldsymbol{r}}_i \theta_i (c_1^2 \triangle \boldsymbol{x}_j + c_1 c_2 \int \triangle \boldsymbol{x}_i dt + n_1 \boldsymbol{e}_j) \tag{10}$$

Where $\triangle \boldsymbol{x}_j$ is the j^{th} element of the characteristic error $\triangle \boldsymbol{x}$, and $\boldsymbol{e}_j$ is the j^{th} element of the velocity error $\boldsymbol{e}$. The n_1 is positive scalar. It can be seen that the information of the sliding surface is not included in the control law but is reflected in the update law of RBFN. The proposed scheme by (9) and (10) allows controlling the position of end-effector to a desired position, and compensating the error of the offline trained model.

The stability of the system is analyzed as follows. By multiplying both sides of (9) by $\hat{\boldsymbol{J}}$,we have

$$\hat{\boldsymbol{J}}\dot{\boldsymbol{r}} = -\frac{c_2}{c_1} \hat{\boldsymbol{J}} (\hat{\boldsymbol{J}})^{+} \triangle \boldsymbol{x} \tag{11}$$

Note that from (11) and (7), it can be obtained that

$$\hat{\boldsymbol{J}}\dot{\boldsymbol{r}} = \hat{\boldsymbol{J}}\dot{\boldsymbol{r}} - \boldsymbol{J}\dot{\boldsymbol{r}} + \boldsymbol{J}\dot{\boldsymbol{r}} = -\boldsymbol{e} + \dot{\boldsymbol{x}} \tag{12}$$

Substituting (11) into (12), we have

$$\boldsymbol{e} = \dot{\boldsymbol{x}} + \frac{c_2}{c_1} \hat{\boldsymbol{J}} (\hat{\boldsymbol{J}})^{+} \triangle \boldsymbol{x} \tag{13}$$

The Lyapunov function can be designed as

$$V = \frac{1}{2} \boldsymbol{s}^{\mathrm{T}} \boldsymbol{s} + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{l} \Delta \boldsymbol{W}_{ij} \Delta \boldsymbol{W}_{ij}^{\mathrm{T}} \tag{14}$$

Bringing the equation (10) and the equation (8) into (14), we get

$$\begin{aligned} \dot{V} &= \boldsymbol{s}^{\mathrm{T}} \dot{\boldsymbol{s}} - \sum_{i=1}^{n} \sum_{j=1}^{l} \Delta \boldsymbol{W}_{ij} \dot{\hat{\boldsymbol{W}}}_{ij}^{\mathrm{T}} \\ &= (c_1 \triangle \boldsymbol{x}^{\mathrm{T}} + c_2 \int \triangle \boldsymbol{x}^{\mathrm{T}} dt)(c_1 \triangle \dot{\boldsymbol{x}} + c_2 \triangle \boldsymbol{x}) \\ &\quad - \sum_{i=1}^{n} \sum_{j=1}^{l} \Delta \boldsymbol{W}_{ij} \dot{\boldsymbol{r}}_i \theta_i (c_1^2 \triangle \boldsymbol{x}_j + c_1 c_2 \int \triangle \boldsymbol{x}_j dt + \boldsymbol{e}_j) \\ &= -c_2 (c_1 \triangle \boldsymbol{x}^{\mathrm{T}} + c_2 \int \triangle \boldsymbol{x}^{\mathrm{T}} dt) \hat{\boldsymbol{J}} (\hat{\boldsymbol{J}})^{+} \triangle \boldsymbol{x} \\ &\quad + c_1 c_2 \triangle \boldsymbol{x}^{\mathrm{T}} \triangle \boldsymbol{x} + c_2^2 (\int \triangle \boldsymbol{x}^{\mathrm{T}} dt) \triangle \boldsymbol{x} - n_1 \boldsymbol{e}^{\mathrm{T}} \boldsymbol{e} \\ &= -n_1 \boldsymbol{e}^{\mathrm{T}} \boldsymbol{e} \le 0 \end{aligned} \tag{15}$$

As $V > 0$ and $\dot{V} \le 0$,the closed-loop system is stable. When $\dot{V} \equiv 0$, there is $\boldsymbol{e} = \sum_{i=1}^{6} \Delta \boldsymbol{W}_i \theta_i \dot{\boldsymbol{r}}_i \equiv 0$, and since $\dot{\boldsymbol{r}} = -\frac{c_2}{c_1} (\hat{\boldsymbol{J}})^{+} \triangle \boldsymbol{x}$, it follows from Lasalle's invariance principle that when $t \to +\infty$, there is $\Delta x \to 0$.

4 Simulation results

We carry out simulation experiments to validate the proposed method. In order to show the superiority of performance of RBFN in this paper compared to performance of purely online estimation model. The first experiment executed by Kalman Filter method was carried out compared with the second experiment where Jacobian matrix was estimated by offline trained RBFN. The third experimental objective is to highlight the advantages of the integral sliding mode surface compared to the linear sliding mode surface. To verify that the update law can increase the system's robustness, an additional set of experiments is conducted where the interference matrix is introduced. Assume that the original camera-to-end-effector transfer matrix expression is

$${}^{cam}\boldsymbol{T}_{end} = {}^{cam}\boldsymbol{T}_{base} \cdot {}^{base}\boldsymbol{T}_{end} \tag{16}$$

Because of error in the position and pose of the camera each time, there is actually an interference matrix: ${}^{cam1}\boldsymbol{T}_{cam}$ in the transfer matrix from the camera coordinate system to the end-effector of the robot arm, expressed as

$$ {}^{cam1}\boldsymbol{T}_{end} = {}^{cam1}\boldsymbol{T}_{cam} \cdot {}^{cam}\boldsymbol{T}_{base} \cdot {}^{base}\boldsymbol{T}_{end} \tag{17} $$

For the first experiment, the Jacobian matrix is estimated purely online using Kalman filter; for the second experiment, the Jacobian matrix is estimated directly using an offline trained RBFN; for the third experiment, the sliding surface is designed as $\boldsymbol{s}=c_3 \Delta \boldsymbol{x}$. The robot arm joint control law is designed as $\dot{\boldsymbol{r}} = -k(\hat{\boldsymbol{J}})^{+} \Delta \boldsymbol{x}$, and the model update law is $\dot{\hat{\boldsymbol{W}}}_{ij}^{\mathrm{T}} = \dot{\boldsymbol{r}}_i \theta_i (n_2 \Delta \boldsymbol{x}_j + n_3 \boldsymbol{e}_j)$. When $c_3^2 = n_2$,by deriving the Lyapunov function $V = \frac{1}{2}\boldsymbol{s}^{\mathrm{T}}\boldsymbol{s} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{l}\Delta \boldsymbol{W}_{ij}\Delta \boldsymbol{W}_{ij}^{\mathrm{T}}$, it can be obtained that

$$ \begin{aligned} \dot{V} &= \boldsymbol{s}^{\mathrm{T}}\dot{\boldsymbol{s}} - \sum_{i=1}^{n}\sum_{j=1}^{l}\Delta \boldsymbol{W}_{ij}\dot{\hat{\boldsymbol{W}}}_{ij}^{\mathrm{T}} \\ &= c_3^2 \Delta \boldsymbol{x}^{\mathrm{T}}(\boldsymbol{e} + \hat{\boldsymbol{J}}\dot{\boldsymbol{r}}) \\ &\quad - \sum_{i=1}^{n}\sum_{j=1}^{l}\Delta \boldsymbol{W}_{ij}\dot{\boldsymbol{r}}_i\theta_i(n_2\Delta \boldsymbol{x}_j + n_3\boldsymbol{e}_j) \\ &= -kc_3^2\Delta \boldsymbol{x}^{\mathrm{T}}\Delta \boldsymbol{x} - n_3\boldsymbol{e}^{\mathrm{T}}\boldsymbol{e} \le 0 \end{aligned} \tag{18} $$

Similarly, the stability of the system can be obtained. The fourth experiment is our scheme presented from (9) to (10).

When the interference transfer matrix is not considered, the simulation plots of each experiment are shown in Fig. 6. The method used in this paper converges faster compared to the other three methods. The third method based on linear sliding mode surface is still better than the RBFN without an online update method. Although the camera position and pose are consistence with that of offline training, the offline trained RBFN is not guaranteed to be the optimal solution in all cases. The norm of $\dot{\boldsymbol{r}}$ decreases as the characteristic error $\Delta \boldsymbol{x}$ converges to zero. Because the model update, $\Delta \boldsymbol{W}$ reduces, making the RBFN adaptable to the change of $\dot{\boldsymbol{r}}$, speeding up the convergence of $\boldsymbol{e}$. The Kalman filter purely online update method uses significantly less information than the other methods with RBFN, and therefore its experimental performance is weaker than the other methods. Due to setting a reasonable initial value of the integral term, the method used in this paper is in the initial state on the sliding surface, and the convergence speed is better than that of the third method.

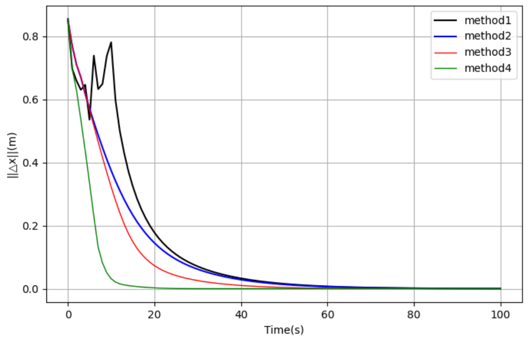

Fig. 6: The simulation plot when there is no interference matrix

When considering the interference matrix due to camera position and pose errors, the simulation plot of each method is shown in Fig. 7, and the interference matrix is numerically set to

$$ {}^{cam1}\boldsymbol{T}_{end} = \begin{bmatrix} \cos(\frac{\pi}{4}) & -\sin(\frac{\pi}{4}) & 0 & 0 \\ \sin(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) & 0 & 0.2 \\ 0 & 0 & 1 & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19} $$

where the unit of length is in meters, and the unit of angle is in radians.

The online update method with Kalman filter generates larger errors at the beginning, because the initial estimation of Jacobian matrix is given without interference; as for thr second method, the RBFN trained without considering the interference matrix can still make the feature error converge to zero after the introduction of the interference matrix; and the third linear sliding surface-based method converges better than the method without the online update law. The performance of the algorithm used in this paper is still the best.

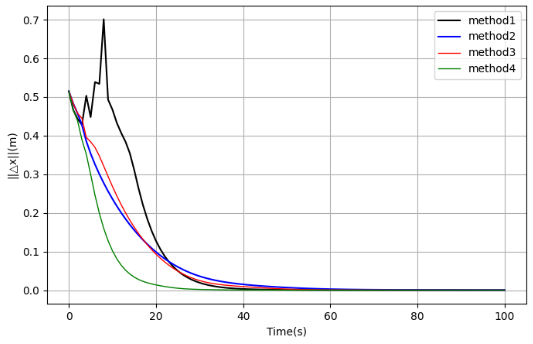

Fig. 7: The simulation diagram when introducing the interference matrix

5 Conclusion

To address the complexity and importance of real-time estimation of Jacobian matrix in the Calibration-free visual servoing system, this paper proposes an online updated RBF neural network and designs the corresponding integral sliding mode control law; some changes are made to the basis function of RBFN to make it more consistent with the characteristics of the robotic arm model, which improves the learning speed during training. The simulation results show that the sliding mode surface and the model update law used in this paper can accelerate the convergence of the system state, which is simple and effective.

References

[1] Dang Hongshe, Hou Jinliang, Qiang Hua, SCARA automatic assembly system based on vision guided [J], *Application of Electronic Technique*, 2017, 43(5): 21-24.

[2] Robotics_Knowledgebase.Visual_Servoing[EB/OL],(2019-11-13)[2022-01-20],https://roboticsknowledgebase.com/wiki/state-estimation/visual-servoing/.

[3] Fang Yongchun, A survey of robot visual servoing [J], *Transactions on Intelligent Systems*, 2008, 3(2): 109-114.

[4] Jia Bing-Xi, Liu Shan, Zhang Kai-Xiang, Chen Jian, Survey on robot visual servo control: vision system and control strategies [J], *Acta Automatica Sinica*, 2015, 41(5): 861−873.

[5] Tao Bo, Gong Zeyu, Ding Han, Survey on uncalibrated robot visual servoing control [J], *Chinese Journal of Theoretical and Applied Mechanics*, 2016, 48(4): 767-783.

[6] Fei Li, Sliding Mode Variable Structure Control for Visual Servoing System [J], *International Journal of Automation and Computing*, 2010, 7(3): 317-323.

[7] ZHAO Y, XIE W F, Liu S, Image-based visual servoing using improved image moments in 6-DOF robot systems[J], *International Journal of Control Automation & Systems*, 2013, 11(3): 586-596.

[8] WU D, ZHONG X, ZHANG X, Uncalibrated image-based visual servoing based on joint space and image moment[C], *Chinese Control Conference*, 2018: 5391-5397.

[9] Peng Yeyuguang, Wu Dongjie, Design and implementation of robot uncalibrated visual servoing system[J], *Application of Electronic Technique*, 2020, 46(6): 77-81.

[10] X Lin, Y Wang, J Olkin, and D, Held, “Softgym: Benchmarking deep reinforcement learning for deformable object manipulation,” *in 4th Conference on Robot Learning*, 2020.

[11] L. Rita and K. Yiannis, “Learning shape control of elastoplastic deformable linear objects,” *in 2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[12] YUE Xiaofeng, YANG Jianfeng, MA Guoyuan, CAO Bin, Robot Uncalibrated Visual Servo Control Based on Genetic Optimized RBF Neural Network[J], *Computer & Digital Engineering*, 2020,48(11): 2617-2621.

[13] DONG Zhonghua, CHEN Pengfei, LI Yalong, Picking Manipulator Control Based on PSO-RBF Neural Network[J], *Electrical Automation*, 2022,51(01):181-183+233.

[14] GUO Xiaoyan, ZHANG Ming, Method of personal credit evaluation of bank based on RBF neural network with weight, *Computer Engineering and Applications*, 2013, 49(5)：258-262.

[15] Zou S, Lv Y, Man Y, et al, Design and implement of shape detection for the soft manipulator[C], *in 39th Chinese Control Conference*, 2020: 3972-3977.