2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

An Online RCM Adjusting System for Robot-Assisted Retinal Surgeries

Jun Xia[a], Ting Wang[b], Huanqi Ni[a], Yanlin Li[a], Ruoxi Chen[a],
M. Ali Nasseri[c], Haotian Lin[d], Kai Huang[a]
[a]School of Data and Computer Science, Sun Yat-sen University; [b]Department of Ophthalmology,
The First Affiliated Hospital of Fujian Medical University; [c]Technical University of Munich;
[d] Zhongshan Ophthalmic Center, Sun Yat-sen University

***Abstract*—In robot-assisted retinal surgery, a Remote Center of Motion (RCM) allows the surgical instrument to rotate around a distal fixed point without any lateral translations. The RCM point should be perfectly aligned inside the trocar. Otherwise, unexpected tool translations at the expected remote center will enlarge the force applied to the trocar and consequently result in post-operative complications. Due to the narrow size of the trocar and the lack of real-time detection equipment, the RCM point is hard to be perfectly located inside the trocar. Even if the RCM is perfectly aligned, the movement of the tissue around the eyeball could make it inappropriate again. In this paper, inspired by the control strategy of surgeons, an online RCM adjusting strategy is proposed. Instead of only using one fixed RCM point, to restrict the force between the surgical tool and the trocar, the proposed strategy adjusts the position of the RCM point during the motion. The results show our approach significantly reduces the force between the robot end-effector and surgical port by 64.2%. In addition, the results also demonstrate that our approach complies the RCM trajectories without deforming or spoiling the working space, which is significantly important for obeying surgeon's instructions in practice.**

I. INTRODUCTION

A qualified retinal surgeon should be capable of precision intraocular manipulation while taking into account RCM control to reduce the trauma on the sclera as well as minimize the intraoperative target movement. RCM is a property of the mechanism in the case of a hardware-based/mechanical RCM or the control approach in the case of a software-based RCM, which allows the surgical instrument to rotate around a distal fixed point and restricts any translations except for the depth of penetration [1]. For the hardware-based RCM, the specific location of the RCM point is fixed by its mechanical structure[2], [3], [4], [5], [6], [7]. To align the fixed point, extra actuators should be used to move the entire RCM mechanism, which is not discussed in this paper. For the software-based RCM, it uses control strategy to restrict instrument trajectory at the designated RCM point by tracking desired joint paths[8], [9], [10], [11], [12], [13]. The position of this point can be adjusted relative to the base. The initial RCM point is usually set at the instrument tip or the marker on the instrument. Subsequently, the robot moves the instrument to position the initial RCM point inside the incision, which is the trocar for the retinal surgery, and relocated a new RCM point at the incision or the trocar relative to the base.

It is important for a robot-assisted retinal surgery that the initial RCM point should be perfectly aligned inside the trocar. Otherwise, unexpected tool translations at the expected remote center will enlarge the force applied to the trocar and consequently result in post-operative complications. However, locating this point precisely is hard. To provide a safe surgical tunnel at the sclera, the trocar usually has a very small size, which has a length of $2 \sim 3mm$ with only $300 \sim 600\mu m$ diameter. There is a lack of effective detection equipment to observe the interior of this narrow space in real-time. Thus, there is a high possibility that the initial point could be inappropriately set inside the trocar (e.g., too close to the inner wall or the entrance of the trocar). Besides, even if the point is set appropriately, the tissue around the trocar and the eyeball is elastic, which means a small force could move the trocar and make the RCM point inappropriate again. In retinal surgeries, keeping the trocar fixed throughout the whole operation is almost impossible. It also happens that the surgeon intentionally rotates the eyeball to observe the previously invisible area, and an experienced surgeon would also not deliberately keep the trocar fixed. Instead of that, the surgeon adjusts the control strategy of his or her hands by the feeling of the force between his or her hand-held instrument and the trocar while turning the tools. Therefore, it could be better if the robot has a human-like adjusting algorithm to adjust the position of the RCM point instead of only using one fixed RCM point relative to its base.

In this paper, inspired by the control strategy of surgeons, an online RCM adjusting strategy is proposed to explore new RCM points that reduce the force applied by the surgical tool. The strategy divides the causes of increasing force into two situations: robot-caused and human-caused force. For different situations, different rotation velocities are given to make the robot quickly or constantly adjust to appropriate RCM points that have smaller forces applied to the trocar. To make the robot comply the RCM motion when adjusting the RCM point, a Radial Basis Function (RBF) NN approximator is inserted in the Jacobian matrix of the robot to approximate the disparities between the kinematics with the RCM point in use and the kinematics with the RCM point to be adjusted to. The experimental results show our approach significantly reduces the force between the robot end-effector and surgical port while following the instructions from the user. The comparison results also show that our approach obtain the force reduction of 64.2% and maintain the angle RCM range of 134%.

979-8-3503-7769-9/24/$31.00 ©2024 IEEE

II. System Overview

A Hybrid Parallel-Serial Micro-manipulator [14] (HPSM) is used as our starting point, which allows adjustable RCM control strategy that can change the RCM point during the motion. Therefore, our system mainly consists of three parts: the force sensor attached to the tool holder, the HPSM which can change the RCM point during the motion, and RCM adjusting strategy based on online learning.

As Fig. 1 shows, the surgical tool is installed on a 3D force sensor with force resolution of $1mN$ in τ_a, τ_b, and τ_c axis. The RCM point in the trocar is denoted by γ. The HPSM used as our start point has five DOFs including three translations and two angles on its end-effector with five joints ($q = [q_1, q_2, \cdots, q_5]^T$). In Fig. 1, it should be noted that when the robot works in RCM control mode, only two rotational DOFs, which are θ_1 and θ_2 marked by blue and green, and one translation along the penetration direction, which is marked by red, are allowed. It should also be noted that the translation along the penetration direction causes the frictions more than the force applied to the trocar. Besides, the translation along the penetration direction of the HPSM is only related to joint q_5. Thus, we ignore the influence of this direction in following analysis. Robotic kinematics and dynamics analysis of the HPSM can be found in [15].

Our system adjusts the RCM point by the feedback of the force between the surgical tool and the trocar. Once $|\tau| = \sqrt{\tau_a^2 + \tau_b^2 + \tau_c^2}$ exceeds safety threshold β, the adjusting strategy will be activated. The RCM adjusting strategy is divided into two situations: robot-caused and human-caused force. If a force exceeding β happens with any rotation command input detected, it can be considered that the force is caused by the robot-included reasons (e.g., inappropriate RCM point). On the other way, if the force exceeding $\beta + c$ (c is a small positive constant) happens without any rotation command input detected, it can be considered that a relatively larger force is caused by human-included reasons (e.g., the surgeon intentionally adjusting the eye position to observe other locations of fundus), which the robot plays no role in. In this situation, extra angular velocities are calculated and given to do the RCM adjusting.

Since the force sensor is attached to the end-effector, the force $\tau = [\tau_a, \tau_b, \tau_c]^T$ can be aligned with the robotic base coordinate system by a rotation matrix R,

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = R^{-1} \begin{bmatrix} \tau_a \\ \tau_b \\ \tau_c \end{bmatrix} \tag{1}$$

$$R = \begin{bmatrix} cos(\theta_1) & sin(\theta_1)sin(\theta_2) & sin(\theta_1)cos(\theta_2) \\ 0 & cos(\theta_2) & -sin(\theta_2) \\ -sin(\theta_1) & sin(\theta_2)cos(\theta_1) & cos(\theta_1)cos(\theta_2) \end{bmatrix} \tag{2}$$

where τ_x, τ_y, τ_z are the components of the force in x, y, and z axis, respectively. Thus, the angular velocities can be set as

$$\begin{cases} \dot{\theta}_1 = sign(\tau_x)\dot{\theta}_{max,1}\frac{\sqrt{\tau_x^2+\tau_z^2}}{\sqrt{\tau_x^2+\tau_y^2+\tau_z^2}} \\ \dot{\theta}_2 = sign(\tau_y)\dot{\theta}_{max,2}\frac{\sqrt{\tau_y^2+\tau_z^2}}{\sqrt{\tau_x^2+\tau_y^2+\tau_z^2}} \end{cases} \tag{3}$$

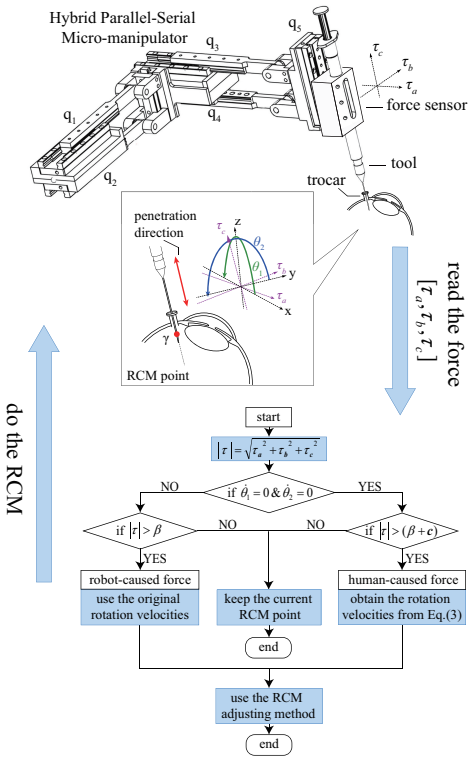


Fig. 1. System Overview

where $sign(\tau_x)$ and $sign(\tau_y)$ indicate that the directions of the given angular velocities $\dot{\theta}_1$ and $\dot{\theta}_2$ are as the same as the directions of the corresponding lateral forces τ_x and τ_y. $\dot{\theta}_{max,1}$ and $\dot{\theta}_{max,2}$ denote the maximum angular velocities of θ_1 and θ_2, which means the robot is expected to adjust itself as fast as possible while large human-caused forces are detected. Finally, each angular velocity is determined by the ratio of its corresponding combined force ($\sqrt{\tau_x^2 + \tau_z^2}$ and $\sqrt{\tau_y^2 + \tau_z^2}$) to the total combined force $\sqrt{\tau_x^2 + \tau_y^2 + \tau_z^2}$. The detailed algorithm about the part of RCM adjusting is introduced in Section. III.

III. RCM Adjusting

Section.II introduces our system scheme. This section further discusses how the RCM adjusting strategy is designed. Our proposed RCM adjusting approach makes the robot adjust its kinematics with two conditions complied: 1) complying the RCM motion; 2) decreasing the force. Thus, in Section.III-A we introduce the first condition by demonstrating how the proposed system can restrict the translation errors while complying RCM motion. Then, in Section.III-B, to satisfy the second condition, an online learning method is proposed to decrease the force.

A. *Complying* RCM *Motion*

In Fig. 1 we denote the force, RCM point, and the joint motors' trajectories as $\tau = [\tau_a, \tau_b, \tau_c]^T$, γ, and $q = [q_1, q_2, \cdots, q_5]^T$, respectively. It should be noted that the HPSM has no singularities or redundant solutions in its working space and robotic kinematics analysis of it can

also be found in [15]. Thus, we directly use J to denote its Jacobian matrix in following deductions. The coordinate system of the force sensor is converted to the robotic base coordinate system by the rotation matrix R in Section.II.

To ensure the robot comply the RCM control, instead of decreasing the velocities or displacements of the joints, following control strategy should be complied during changing RCM points,

$$\begin{cases} \dot{q} = \widehat{J}(\widehat{\gamma}_t, q)^{\dagger}[0,0,0,\dot{\widehat{\theta}}]^T \\ [\dot{l},\dot{\bar{\theta}}]^T = \bar{J}(\bar{\gamma},q)\dot{q} \\ \tau = R\tau_l = Rf(l) \end{cases} \tag{4}$$

where, we use $\widehat{\bullet}$, $\bar{\bullet}$, and $\widetilde{\bullet}$ to denote the real ones being used, the ideal ones to adjust to, and the errors between them, respectively, so $\widehat{\bullet} + \widetilde{\bullet} = \bar{\bullet}$. q is joint motors' position read by the encoders. $\dot{q}$ is joint motors' velocities. $\bullet^{\dagger}$ denotes the generalized inverse operation, which actually is the inverse operation in our case. Here, we define the point $\widehat{\gamma}$ as the real RCM point that is in use, and $\bar{\gamma}$ as the ideal RCM point that we want the robot to adjust to. Correspondingly, $\widehat{J}$ and $\bar{J}$ denote the real and ideal Jacobian matrices that are two 5×5 matrices. $\dot{\widehat{\theta}} = [\dot{\widehat{\theta}}_1, \dot{\widehat{\theta}}_2]$ represents the rotation velocities set by the user at the real RCM point, which have no translations in the x, y and z axis with $\dot{\widehat{\theta}}_1$ and $\dot{\widehat{\theta}}_2$ rotation speed for each angle. Correspondingly, $\dot{\bar{\theta}} = [\dot{\bar{\theta}}_1, \dot{\bar{\theta}}_2]$ represents the rotation velocities happening at the ideal RCM point $\bar{\gamma}$. $\dot{l} = [\dot{x}, \dot{y}, \dot{z}]$ represents the translation errors at the ideal RCM point $\bar{\gamma}$. $\tau_l = [\tau_x, \tau_y, \tau_z]^T$ denotes the force in the robotic base coordinate system, which follows Hooke's law represented by $f(l)$. It should be noted that we execute the joint trajectory tracking by directly using the Proportion Integral Differential (PID) controller. Thus, the PID tracking errors are ignored in our deductions because we are proposing an online RCM adjusting method while PID tracking errors can cause translation errors even the ideal RCM point is perfectly adjusted to.

The control strategy shown in Equ. (4) can be rewritten as,

$$\begin{cases} \dot{q} = (\bar{J}^{\dagger}_{(:,3:4)} - \widetilde{J}^{\dagger}_{(:,3:4)})\dot{\widehat{\theta}}^T \\ [\dot{l},\dot{\bar{\theta}}]^T = \bar{J}\dot{q} \\ \tau = Rf(l) \end{cases} \tag{5}$$

where $J^{\dagger}_{(:,3:4)}$ denotes the third and fourth columns of $J^{\dagger}$, which corresponds to $\dot{\widehat{\theta}}_1$ and $\dot{\widehat{\theta}}_2$. It can be seen that if $\widetilde{J}^{\dagger}_{(:,3:4)}$ is zero, which means the RCM point in use is perfectly the ideal one,

$$\begin{aligned} [\dot{x},\dot{y},\dot{z},\dot{\bar{\theta}}]^T =& \bar{J}\bar{J}^{\dagger}_{(:,3:4)}\dot{\widehat{\theta}}^T \\ =& [0,0,0,\dot{\bar{\theta}}]^T \end{aligned} \tag{6}$$

However, it is only an ideal circumstance. Instead, we use a RBF NN approximator $\kappa = \widetilde{J}^{\dagger}_{(:,3:4)} + \eta$ to approximate the error matrix $\widetilde{J}^{\dagger}_{(:,3:4)}$. We use RBF NN in this paper, since RBF NN has been proven to be easier to train, simpler than other NN structures, resilient to input noise, and has an especially good online learning ability [16].

*Lemma 1 (*RBF *approximation):* If there are sufficient nodes, under suitable width σ and node centers $\boldsymbol{\delta}$, RBF NN can approximate any smooth function $\boldsymbol{F_a(x)}$ over a compact set $\boldsymbol{x} \in \boldsymbol{\Omega_x}$ with convergent errors: $\boldsymbol{F_a(x)} = \boldsymbol{W^* S(x) + \eta(x)}$ where $\boldsymbol{W^*}$ is the ideal weight matrix, and $\boldsymbol{\eta(x)}$ is the convergent errors, satisfying $||\boldsymbol{\eta(x)}||_{\boldsymbol{2}} \leq \boldsymbol{\eta_o}$, $\boldsymbol{\eta_o}$ is a constant[17].

Adding κ in the inverse kinematics of Equ. (5), we obtain

$$[\dot{x},\dot{y},\dot{z}]^T = \bar{J}_{(1:3,:)}\dot{\widehat{\theta}}^T \tag{7}$$

$$\dot{\bar{\theta}}^T = \bar{J}_{(4:5,:)}\dot{\widehat{\theta}}^T \tag{8}$$

where $\bar{J}_{(1:3,:)}$ denotes the first three rows of $\bar{J}$ and $\bar{J}_{(4:5,:)}$ denotes the last two rows of $\bar{J}$. Equ. (7) is the velocity error of the translations relative to the translation constrains of $[0,0,0]^T$ and Equ. (8) is the rotation velocity error of the angles of θ_1 and θ_2. Considering Lemma.1 and taking the 2-norm on both sides of Equ. (7) yields

$$\begin{aligned} ||[\dot{x},\dot{y},\dot{z}]^T||_2 &= ||\bar{J}_{(1:3,:)}\dot{\widehat{\theta}}^T||_2 \\ &\leq ||\bar{J}_{(1:3,:)}||_2||\dot{\widehat{\theta}}^T||_2 \\ &\leq ||\bar{J}_{(1:3,:)}||_2\eta_o||\dot{\widehat{\theta}}||_2 \end{aligned} \tag{9}$$

The Jacobian matrix $\bar{J}_{(1:3,:)}$ is bounded and can be represented as $||\bar{J}_{(1:3,:)}||_2 \leq \psi_o$, where ψ_o is a constant. Considering $||[\dot{x},\dot{y},\dot{z}]^T||_2 = ||[\dot{x},\dot{y},\dot{z}]||_2$, the translation error is shown below.

$$||[\dot{x},\dot{y},\dot{z}]||_2 \leq \psi_o\eta_o\dot{\widehat{\theta}}_o \tag{10}$$

where $\dot{\widehat{\theta}}_o$ is the 2-norm of the rotation velocities set by the user. Thus, the translation error $\dot{x}$, $\dot{y}$, and $\dot{z}$ can be restricted within a certain and constant boundary $\psi_o\eta_o\widehat{\theta}_o$ while doing RCM moving.

It should be noted the translation error has a linear correlation with the convergent error η_o and decreasing the convergent error can compress the translation error close to zero. Similar to Equ. (9). the rotation errors can also be restricted as,

$$||[\dot{\bar{\theta}}]||_2 \leq \psi_1\eta_o\dot{\widehat{\theta}}_o \tag{11}$$

where ψ_1 is a constant and $||\bar{J}_{(4:5,:)}||_2 \leq \psi_1$. It can be seen that while the learning error converges to 0, the translation and rotation errors also converge to 0.

B. Decreasing Force

Section.III-A proves that by adding a well-trained RBF NN in the inverse kinematics during the RCM process, the translations on the ideal RCM point can be restricted in a constant boundary. In other words, by adding κ, $\widehat{J}^{\dagger}_{(:,3:4)} + \kappa$ can get closer to the ideal inverse Jacobian matrix $\bar{J}^{\dagger}_{(:,3:4)}$ with bounded errors. This section explains how the κ is trained online to decrease the force τ_x and τ_y.

We use RBF NN in the NN approximator κ shown in Fig. 2. Since we try to approximate the Jacobian matrix $\widetilde{J}(\bar{\gamma},q)^{\dagger}_{(:,3:4)}$, we use q as the NN input vector. It should be noted that, q is the motor trajectory which can be obtained by the encoder. To keep the size of the RBF output coincident

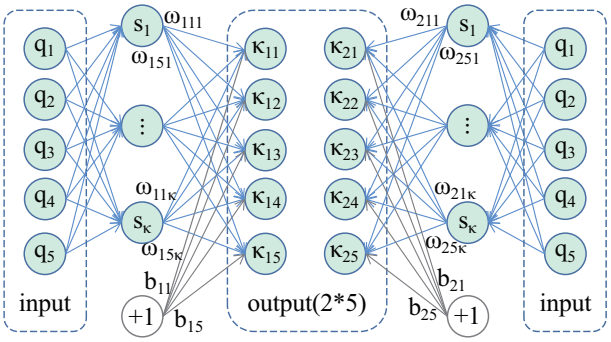


Fig. 2. NN Structure

with $\widetilde{J}(\bar{\gamma},q)^{\dagger}_{(:,3:4)}$, a RBF NN approximator κ with a 5×2 output vector is designed in Fig.2. The RBF NN approximator κ can be designed as,

$$\kappa(q)_{ij} = W_{ij}S(q) + b_{ij} \tag{12}$$

for $i = 1, 2$ and $j = 1, 2, 3, 4, 5$. $\kappa(q)$ is a 5×2 output matrix. $W_{ij} = [W_{ij1}, W_{ij2}, \cdots, W_{ijk}]$ is the weight matrix, where k is the number of neurons. $S(q) = [S_1(q), S_2(q), \cdots, S_k(q)]^T$ is the neuron matrix, where $S_k(q) = e^{-\|q_t-\delta_k\|^2/\sigma^2}$. δ_k denotes the kth center while σ donates the bandwidth shared by all neurons. b is a 5×2 bias matrix. It should be noted that, we initialize the weight matrix W and the bias matrix b with zeros.

1) NN *Updating Law:* In this part, we use the chain rule to deduce the weight updating law for the RBF NN κ in such a discrete system that the robot works from step $t = 0$ to step $t = t_o$ and spends ς time for each moving between adjacent steps. The loss function at step t_o is defined as,

$$g(\tau_{t_o}) = \frac{(\tau^2_{a,t_o} + \tau^2_{b,t_o} + \tau^2_{c,t_o})}{2} \tag{13}$$

Restricting the force to zero, however, is impossible for the system, which could also waste too much time and computing resource. Thus, we set a force threshold β in the online learning procedure. κ is trained based on the loss function (13) until $|\tau_{t_o}| \le \beta$ for the situation of robot-caused force or $|\tau_{t_o}| \le \beta + c$ for the situation of human-caused force.

Taking the derivative of the loss function (13) to the weight matrix W and using the chain rule, we have

$$\frac{\partial g(\tau_{to})}{\partial W_{ij,t}} = L^g_\tau L^\tau_l L^l_W \tag{14}$$

for $i = 1, 2$ and $j = 1, 2, 3, 4, 5$, where

$$\begin{cases} L^g_\tau = \begin{bmatrix} \frac{\partial g(\tau_{to})}{\partial \tau_{a,t_o}} & \frac{\partial g(\tau_{to})}{\partial \tau_{b,t_o}} & \frac{\partial g(\tau_{to})}{\partial \tau_{c,t_o}} \end{bmatrix} \\ L^\tau_l = \begin{bmatrix} \frac{\partial \tau_{a,t_o}}{\partial \dot{x}_{t_o}} & \frac{\partial \tau_{a,t_o}}{\partial \dot{y}_{t_o}} & \frac{\partial \tau_{a,t_o}}{\partial \dot{z}_{t_o}} \\ \frac{\partial \tau_{b,t_o}}{\partial \dot{x}_{t_o}} & \frac{\partial \tau_{b,t_o}}{\partial \dot{y}_{t_o}} & \frac{\partial \tau_{b,t_o}}{\partial \dot{z}_{t_o}} \\ \frac{\partial \tau_{c,t_o}}{\partial \dot{x}_{t_o}} & \frac{\partial \tau_{c,t_o}}{\partial \dot{y}_{t_o}} & \frac{\partial \tau_{c,t_o}}{\partial \dot{z}_{t_o}} \end{bmatrix} \\ L^l_W = \begin{bmatrix} \frac{\partial \dot{x}_{t_o}}{\partial W_{ij,t_o}} \\ \frac{\partial \dot{y}_{t_o}}{\partial W_{ij,t_o}} \\ \frac{\partial \dot{z}_{t_o}}{\partial W_{ij,t_o}} \end{bmatrix} \end{cases} \tag{15}$$

For the term L^g_τ, we have

$$\begin{cases} \partial g(\tau_{to})/\partial \tau_{a,t_o} = \tau_{a,t_o} \\ \partial g(\tau_{to})/\partial \tau_{b,t_o} = \tau_{b,t_o} \\ \partial g(\tau_{to})/\partial \tau_{c,t_o} = \tau_{c,t_o} \end{cases} \tag{16}$$

For the term L^τ_l, similar to Equ. (1), τ_a, τ_b and τ_c can be expressed by their components on x, y, and z axes, respectively. Considering $f(l)$ in Equ. (4) follows Hooke's law, we have

$$\begin{bmatrix} \tau_{a,t_o} \\ \tau_{b,t_o} \\ \tau_{c,t_o} \end{bmatrix} = Rf(l) = R\sum_{t=0}^{t=t_o}\left(\begin{bmatrix} \lambda_x \dot{x}_t \\ \lambda_y \dot{y}_t \\ \lambda_z \dot{z}_t \end{bmatrix} \varsigma \right) \tag{17}$$

where R is described in Equ. (2). λ_x, λ_y, and λ_z are the stiffness coefficient in the x, y, and z axis, respectively.

For the term L^l_W, considering Equ. (12),

$$\begin{aligned} \frac{\partial \dot{x}_{t_o}}{\partial W_{ij,t_o}} =& \partial \bar{J}_{(1,:),t_o} \kappa \dot{\widehat{\theta}}^T_{t_o} / \partial W_{ij,t_o} \\ =& \partial \bar{J}_{(1,j),t_o} W_{ij,t_o} S_{t_o} \dot{\widehat{\theta}}_{i,t_o} / \partial W_{ij,t_o} \\ =& \bar{J}_{(1,j),t_o} S_{t_o} \dot{\widehat{\theta}}_{i,t_o} \end{aligned} \tag{18}$$

$$\begin{aligned} \frac{\partial \dot{y}_{t_o}}{\partial W_{ij,t_o}} =& \partial \bar{J}_{(2,:),t_o} \kappa \dot{\widehat{\theta}}^T_{t_o} / \partial W_{ij,t_o} \\ =& \partial \bar{J}_{(2,j),t_o} W_{ij,t_o} S_{t_o} \dot{\widehat{\theta}}_{i,t_o} / \partial W_{ij,t_o} \\ =& \bar{J}_{(2,j),t_o} S_{t_o} \dot{\widehat{\theta}}_{i,t_o} \end{aligned} \tag{19}$$

$$\begin{aligned} \frac{\partial \dot{z}_{t_o}}{\partial W_{ij,t_o}} =& \partial \bar{J}_{(3,:),t_o} \kappa \dot{\widehat{\theta}}^T_{t_o} / \partial W_{ij,t_o} \\ =& \partial \bar{J}_{(3,j),t_o} W_{ij,t_o} S_{t_o} \dot{\widehat{\theta}}_{i,t_o} / \partial W_{ij,t_o} \\ =& \bar{J}_{(3,j),t_o} S_{t_o} \dot{\widehat{\theta}}_{i,t_o} \end{aligned} \tag{20}$$

where $\bar{J}_{(i,:,t_o)}$ denotes the ith row of $\bar{J}$ at step t_o and $\bar{J}_{(i,j,t_o)}$ donates the item of $\bar{J}$ on the ith row and the jth column at step t_o. Since $\bar{J}$ is not known, so we instead use $\widehat{J}$ in the weight updating law.

Hence, the weight updating law can be deduced as,

$$\Delta W_{ij,t_o} = l_r \varsigma \tau^T_{t_o} R \begin{bmatrix} \lambda_x \widehat{J}_{(1,j),t_o} \\ \lambda_y \widehat{J}_{(2,j),t_o} \\ \lambda_z \widehat{J}_{(3,j),t_o} \end{bmatrix} S_{t_o} \dot{\widehat{\theta}}_{i,t_o} \tag{21}$$

where l_r is the learning rate. The derivation of the bias updating law is basically the same as that of the weight updating law, so here we give the bias updating law directly.

$$\Delta b_{ij,t_o} = l_r \varsigma \tau^T_{t_o} R \begin{bmatrix} \lambda_x \widehat{J}_{(1,j),t_o} \\ \lambda_y \widehat{J}_{(2,j),t_o} \\ \lambda_z \widehat{J}_{(3,j),t_o} \end{bmatrix} \dot{\widehat{\theta}}_{i,t_o} \tag{22}$$

where the stiffness coefficient λ is set as a constant in the learning rate l_r. Considering the tissues around the trocar, the stiffness coefficients λ_x, λ_y, and λ_z of the trocar are hard to know, so we directly treat them as the same and merge them to the learning rate. Thus, the final weight and bias updating law can be expressed as,

$$\begin{cases} \Delta W_{ij,t_o} = l_r \varsigma \tau^T_{t_o} R \begin{bmatrix} \widehat{J}_{(1,j),t_o} \\ \widehat{J}_{(2,j),t_o} \\ \widehat{J}_{(3,j),t_o} \end{bmatrix} S_{t_o} \dot{\widehat{\theta}}_{i,t_o} \\ \Delta b_{ij,t_o} = l_r \varsigma \tau^T_{t_o} R \begin{bmatrix} \widehat{J}_{(1,j),t_o} \\ \widehat{J}_{(2,j),t_o} \\ \widehat{J}_{(3,j),t_o} \end{bmatrix} \dot{\widehat{\theta}}_{i,t_o} \end{cases} \tag{23}$$

2) Center and Bandwidth Selecting: For the center and bandwidth selecting of the Gaussian kernel function. The centers δ are used to measure the Euclidean distance of the input samples, which is the motor trajectories q in this paper. To obtain a satisfied approximation, we want a

compact and uniform set δ, which distributes in the set of q. Hence, the centers are then uniformly selected from the area of $[q_{1,min}, q_{1,max}] \times [q_{2,min}, q_{2,max}] \times \cdots [q_{5,min}, q_{5,max}]]$, where min, max denote the minimum and the maximum position value for the joint. Equ. (24) shows how we set δ, it can be seen that if we select p numbers for each joint, we will have $k = p^5$ centers.

$$\delta_i = [\delta_{i_1,1}, \delta_{i_2,2}, \cdots, \delta_{i_j,j} \cdots, \delta_{i_5,5}] \quad i_j \in (1,p), i \in (1,k)$$
$$\delta_{i_j,j} = q_{j,min} + i_j(q_{j,max} - q_{j,min})/(p-1) \tag{24}$$

In [18], it has been proved that when the value of bandwidth σ is small enough: 1) the RBF kernel can correctly classify all training samples; 2) when σ is too small, it can cause over fitting. In our case, too small σ also requires more centers and neurons for κ, which costs much more time and computing resource for the online learning. Since the bandwidth σ indicates the working range of the Gaussian kernel, we set σ as

$$\sigma = (\sum_{i=1}^{n}((q_{i,max} - q_{i,min})/p)^2)^{(1/2)} \tag{25}$$

Equ. (25) indicates that we want $\sigma \times p$ to cover the distribution of set q_i for $i = 1, 2, \cdots, 5$.

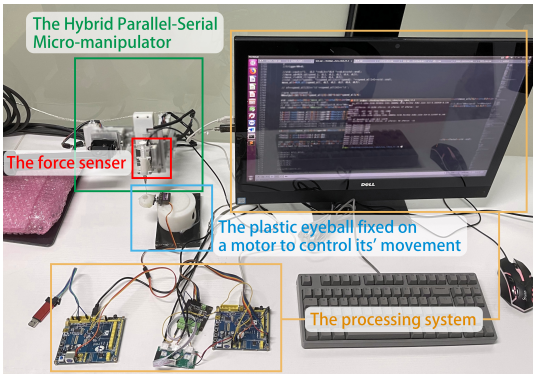
IV. Experiment Result

A. Experiment Setup

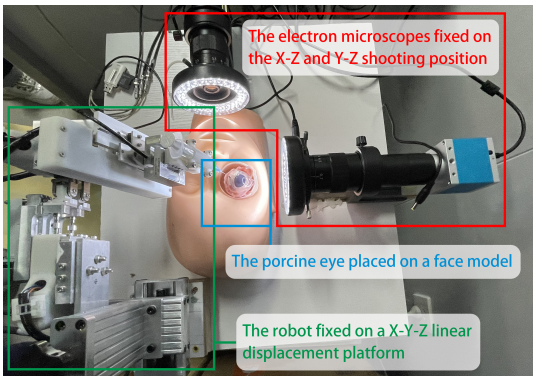
The following experiments utilizes the HPSM. The robot uses a PID controller to track the prescribed trajectory. The moving time ς for each step is set to be 0.1s. Each linear positioner is implemented with a piezoelectric ceramic motor. A 30 Gauge needle is installed on the K3D40 force sensor (ATI Industrial Automation, Apex, NC) with force resolution of $10mN$. The control and learning program run on a Gigabyte GB-BKi7HA-7500 PC with an Intel Core i7-7500U processor and 16GB of memory. For NN parameters, we set $p = 5$ for all following experiments, which produces 5^3 centers and neurons for κ. According to Equ. (24) and (25), we have centers δ distributed in $[-12, -6, 0, 6, 12] \times [-12, -6, 0, 6, 12] \times [-12, -6, 0, 6, 12] \times [-12, -6, 0, 6, 12] \times [-12, -6, 0, 6, 12]$ and bandwidth $\sigma = 15.2$.

To follow the principle of control variables and make the result comparable, a phantom eyeball as shown in Fig. 3(a) is utilized to conduct the experiments in Section. IV-B. The eyeball is fixed on a micro servo motor that can hold the eyeball stably. The tracking error of the micro servo motor is $0.01rads$. The motor rotates the eyeball with an angle of $\pi/18rads$ to simulate the situation of human-caused force. The position of the robot and the eyeball is fixed during the experiment. The initial positions of each joint of the robot and the initial angle of the motor holding the eyeball are the same for each experiment (all zeroes). The phantom eyeball has a small hole as the trocar and the initial RCM point is set at $1mm$ vertically above it. In Section. IV-B, each test was repeated for 10 times.

In Section. IV-C, an ex-vivo porcine eye is utilized to simulate the surgical environment. This part of the experiment is aimed to further show how the online adjusting system react to a scene more like a surgical environment, where the tissue around the eyeball is soft and elastic, and the trocar can be moved by a small force. As shown in Fig.3(b), to capture the trajectory of the trocar and the surgical tool, two electron microscopes are fixed at x-z and y-z shooting position, which deliver a resolution of 1.2 μm. The robot is fixed on an x-y-z linear displacement platform, which is used to adjust the position of the robot relative to the microscopes so that both microscopes can capture clear images. The porcine eye is placed on the face model without any fixation and only depends on the tissue around it to keep stable. The experiments in this part are hard to be repeated under the same controlling variables. Thus, the experimental figures in Section. IV-C only show the result of one test.



(a) The phantom eye experiment for Section.IV-B

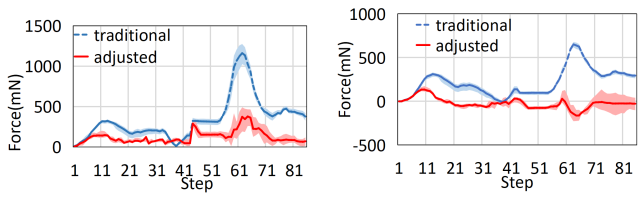


(b) The ex-vivo porcine eye experiment for Section.IV-C

Fig. 3. Experiment environment

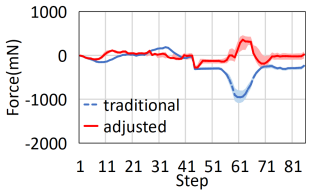
B. The performance of the online RCM adjusting

1) Overall performance of the online RCM adjusting: First, the performances in master-slave mode of the online RCM adjusting system against the traditional RCM that has no RCM adjusting is investigated. The RCM point of the robot is set to the same position, which is at $1mm$ vertically above the entry point on the eye. We set force threshold $\beta = 100mN$, $\underline{c} = 100mN$, learning rate $l_r = 1$, and rotation speed $\dot{\widehat{\theta}} = [0.005rads/s, 0rads/s]$. To make the results comparable, a master signal set is utilized to replace the real master manipulator, which is shown in Fig. 5(a). The black dotted line is the signal of translation velocity along the penetration direction. The blue dotted line is the rotation velocity $\dot{\theta}_1$ and the red line is the rotation velocity $\dot{\theta}_2$. This master signal set is utilized for each test in this experiment.

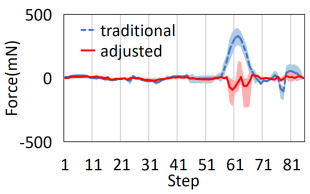


(a) Force value of $|\tau|$

(b) Force value of τ_a

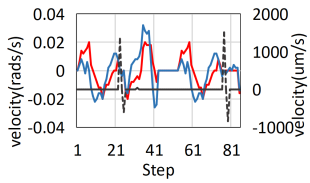


(c) Force value of τ_b

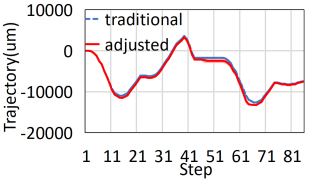


(d) Force value of τ_c

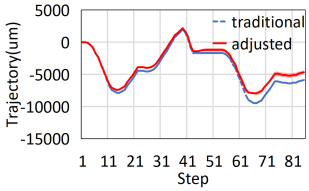
Fig. 4. The force values of traditional RCM versus the online RCM adjusting in Master-Slave control



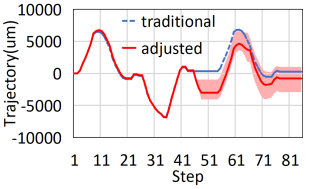
(a) Input master signals



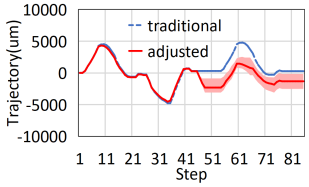
(b) Trajectory of joint q_1



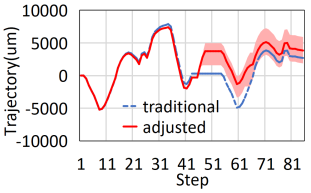
(c) Trajectory of joint q_2



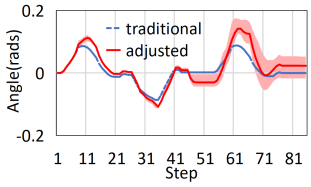
(d) Trajectory of joint q_3



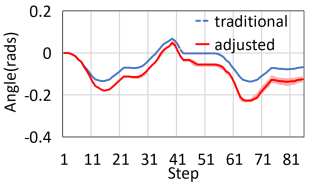
(e) Trajectory of joint q_4



(f) Trajectory of joint q_5

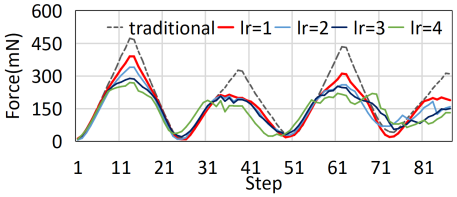


(g) Trajectory of angle θ_1

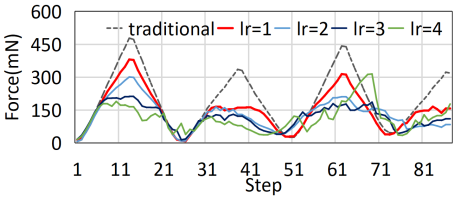


(h) Trajectory of angle θ_2

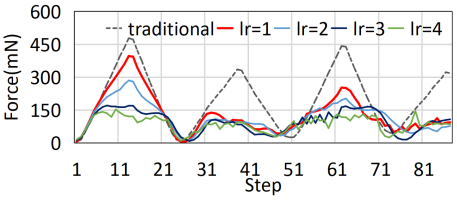
Fig. 5. The details of traditional RCM versus the online RCM adjusting in Master-Slave control



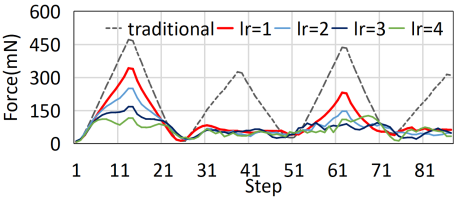
(a) $|\tau|$ with $\beta = 200$with $l_r = 1\ 4$



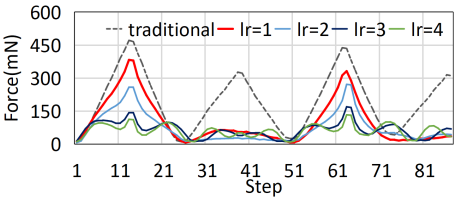
(b) $|\tau|$ with $\beta = 150$with $l_r = 1\ 4$



(c) $|\tau|$ with $\beta = 100$with $l_r = 1\ 4$

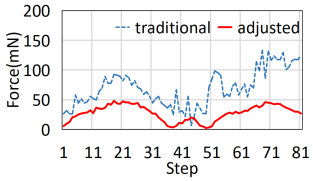


(d) $|\tau|$ with $\beta = 50$with $l_r = 1\ 4$

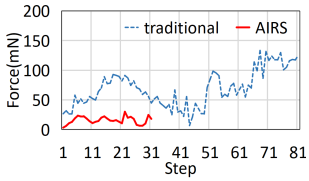


(e) $|\tau|$ with $\beta = 0$with $l_r = 1\ 4$

Fig. 6. The learning performances of the online adjusting with different l_r and β



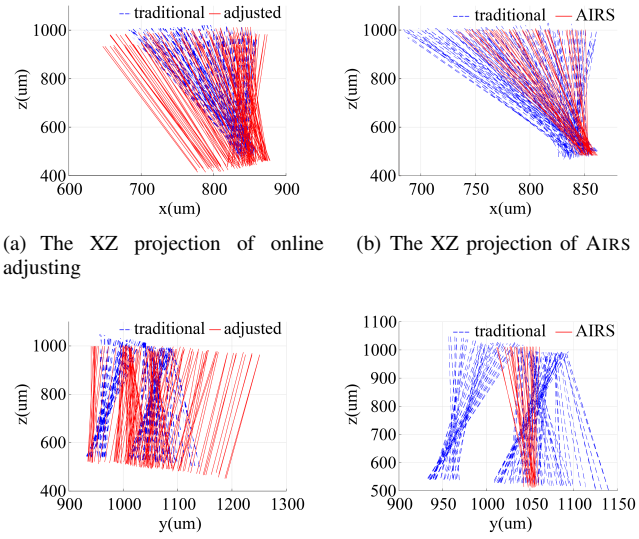
(a) $|\tau|$ of online adjusting



(b) $|\tau|$ of AIRS

Fig. 7. The force performances on the ex-vivo porcine eye

From step 42 to step 46, the input master signals are set to be zero where the plastic eyeball is rotated for $\pi/18rads$ to simulate the situation of human-caused force that surgeon moves the eyeball intentionally. The results are shown in Fig. 4 and 5. The blue dotted lines are of the traditional RCM and the red lines are of the online RCM adjusting system. The stained area is the fluctuation range of 10 test results and the line with the same color as the stained area is the average value of 10 test results. Fig. 4 shows the force $|\tau|$. It can be seen that the online RCM adjusting system can restrict the force smaller than the traditional RCM ($151mN$ vs $427mN$) while doing RCM from step 1 to step 41, corresponding to



(a) The XZ projection of online adjusting

(b) The XZ projection of AIRS

(c) The YZ projection of online adjusting

(d) The YZ projection of AIRS

Fig. 8. The projection of the surgical tool on microscope 1 at XZ plane and microscope 2 at YZ plane plane

the situation of robot-caused force. When the eye is rotated by $\pi/18$ degree, the online RCM adjusting system can also readjust the robot within a safe threshold of the force. In Fig. 4(a), from step 41 to step 47, the red line goes below than $200mN$ rapidly, corresponding to the situation of human-caused force. It should be noted that the readjusting ability of the online RCM adjusting approach also depends on the maximum speed of the motor. After the eyeball has been rotated, the blue line in Fig. 4(a) rise rapidly to $1200mN$ as soon as the robot gets the master signal from step 55 to step 61. It is because the traditional RCM still works on the original RCM point while the eyeball has been moved. In contrast, in Fig. 4(a), the red line has a much lower value ($430mN$) in step 61, because the RCM point has already been adjusted to a more suitable position during step 41 to step 47. The force components τ_a, τ_a, and τ_c in Fig. 4(b) to 4(d) show the online RCM adjusting system can also restrict the force in each direction.

The trajectories of each joint and the RCM rotation angle are shown in Fig. 5(b) to 5(h). From step 42 to step 46, the red lines have relatively large deviations from the blue lines, which means the online adjusting is reacting to the rotation of the eyeball. Except for this part, the red lines have the same form as the blue lines with some deviations from Fig. 5(b) to 5(h), which means the online RCM adjusting system follows the prescribed trajectories joint and obeys the instructions from the user. Overall, the online RCM adjusting system can effectively limit the force by adjusting the RCM online with the RCM instructions and prescribed trajectories followed.

2) The influence of learning parameters: In Equ.23, the rotation speed $\dot{\hat{\theta}}_i$ is given by the master controller. R, S, and J matrices are also determined by robot motor trajectories. Thus, only the learning rate l_r and the force threshold β are the controllable variables in our proposed approach. To make the results comparable, we make the robot do the RCM motion circularly by moving back and forth with constant angular velocities until reaching the maximum or minimum boundary of any joint for 4 times. The angular velocities are set as $\dot{\theta} = [0.0085rads/s, 0.0085rads/s]$. Then in Fig. 6 we set l_r from 1 to 4 and β from $0mN$ to $200mN$ to investigate how they affect the learning performance. It should be noted that each test was also repeated for 10 times and only the average values are shown in this part for clear display.

It can be seen that the force is reduced in each test. Enlarging the learning rate can better reduce the force under the same force threshold β. A too low learning rate would lead to insufficient learning. Decreasing the force threshold β at the same learning rate l_r can also improve the learning performance. However, when β is too low, the performance improvement brought by reducing β is limited. The force performance of $\beta = 50mN$ is close to the one of $\beta = 0mN$. It may be caused by the force being too close to the limit and restricting the force to zero is impossible. In our case, β is also used as a condition to switch the online RCM adjusting approach on and off to prevent learning all the time, which wastes much time and computing resource.

C. Online learning vs AIRS

We also compared our approach with the method AIRS described in [19] in an ex-vivo porcine eye. To make the results comparable, in this part we make the robot do the RCM motion circularly by moving back and forth with constant angular velocities until reaching the maximum or minimum boundary of any joint for 4 times. The AIRS mainly focuses on the prediction part while directly decreasing the rotation velocities instead of adjusting the RCM point when the force is predicted to go large. Thus, we repeat the velocity control strategy in [19] as,

$$\hat{\dot{\theta}}_0 = \hat{\dot{\theta}} + V \tag{26}$$

$$V = [0.005sign(\tau_x)C,\ 0.005sign(\tau_y)C] \tag{27}$$

Equ.(26) is quoted from [19] and has some modifications, where we use $\hat{\dot{\theta}}$ and $\hat{\dot{\theta}}_0$ to represent the original and the modified input rotation velocities of the robot. The original rotation velocities $\hat{\dot{\theta}}$ is set to be $[0.0085rads/s, 0.0085rads/s]$. C is set as a constant value, which acts as a proportion of the deceleration. Force threshold β is used to substitute for the predict part in [19]. Once $|\tau| > \beta$, the velocity control strategy in Equ.(26) is activated.

In Fig. 7, the blue lines donate the force of the tradition RCM, and the red lines donate the force of the online adjusting and the AIRS. The porcine eye is very easy to be moved, resulting in the increasing of the force in the blue line from step 61 to step 81 in Fig. 7. It can be seen that both two methods can restrict the force. In Fig. 7(a), the online adjusting method reduce the force by 49% on the porcine eye, and the force curve of online adjusting is also much smooth compared with the traditional force curve. The red line in Fig. 7(b) has the fewest steps, because AIRS shorten the rotation range of RCM, resulting in fewer steps to complete the same rotation cycle.

TABLE I
THE PERFORMANCE COMPARISON OF OURS AND AIRS($lr = 3$, $\beta = 100$ AND $C = 100\%$)

$\beta = 100$	Force reduction	Angle range
Ours	64.2%	134
AIRS	73.7%	26.2

Fig. 8 shows the projections of the surgical tool on two microscopes that are fixed at the x-z and y-z shooting angle. Each line in Fig. 8 is sampled from the captured videos at a frequency of every 15 frames. The blue lines are of the traditional RCM and the red lines are of online adjusting and AIRS, respectively. For each line in Fig. 8, the beginning point with larger z is chosen as the point along the needle that is above the trocar and the end point with smaller z is the point at the trocar. It should be noted that the ex-vivo porcine eye has some sliding pushed by the surgical tool in y-z axes. Thus, the blue line from Fig. 8(c) to Fig. 8(d) has been split into two parts.

It can be seen that the online adjusting can adjust the RCM point according to the force. The online adjusting makes the surgical tool move more smoothly and does not spoil the moving range of the surgical tool, performing more like a surgeon control strategy. Although AIRS achieves a larger force reduction, it does sacrifice the rotation ranges in Fig. 8(b) and Fig. 8(d). AIRS avoids moving into the places with large force. These places are usually close to the moving boundaries that are of the larger rotation angles.

Table. I summarizes the comparison results. It can be seen that, at the learning rate $lr = 3$, online RCM adjusting can obtain the force reduction of 64.2% and maintain the angle RCM range of 134%. AIRS has a larger orce reduction of 73.7% with the remaining RCM angle range of only 26.2%.

V. CONCLUSION

In this paper, instead of only using one fixed RCM point, an online RCM adjusting system is proposed. To implement this system, there are three core parts: a 3D force sensor to obtain the force from the instrument, the HPSM to change RCM point, and the RCM adjusting strategy using a special designed RBF NN to adjust the kinematics of the robot. The strategy is designed to decrease the force while complying with the RCM motion. In addition, our strategy has two modules: 1) the module of robot-caused force, where RCM adjusting strategy works by using the master manipulator input; 2) the module of human-caused force, where our strategy uses extra angular velocities calculated by the force. The results show our approach significantly reduces the force between the robot end-effector and surgical port and also complies the RCM constrains by exploring new RCM points instead of simply reducing the displacements of the joints or the rotation angles.

REFERENCES

[1] G. Zong, X. Pei, J. Yu, and S. Bi, "Classification and type synthesis of 1 dof remote center of motion mechanisms," *Mechanism and Machine Theory*, vol. 43, no. 12, pp. 1585–1595, 2008.

[2] H. C. M. Meenink, R. Hendrix, P. C. J. N. Rosielle, M. Steinbuch, and H. Nijmeijer, "A master-slave robot for vitreo-retinal eye surgery," in *Proceedings of the 10th International Conference of European Society for Precision Engineering and Nanotechnology, 31 May-4 June 2010, Delft Netherlands*, 2010, pp. 408–411.

[3] J. Sun, Z. Yan, and Z. Du, "Optimal design of a novel remote center-of-motion mechanism for minimally invasive surgical robot," *IOP Conference Series: Earth and Environmental Science*, vol. 69, p. 012097, 2017.

[4] S. Nisar, T. Endo, and F. Matsuno, "Design and kinematic optimization of a two degrees-of-freedom planar remote center of motion mechanism for minimally invasive surgery manipulators," *Journal of Mechanisms and Robotics: Transactions of the ASME*, vol. 9, no. 3, p. 031013, 2017.

[5] G. Chen, J. Wang, and H. Wang, "A new type of planar 2-dof remote center-of-motion mechanisms inspired by the peaucellier-lipkin straight-line linkage,," *Journal of Mechanical Design*, vol. 141, no. 1, p. 015001, 2018.

[6] N. Sajid, E. Takahiro, and M. Fumitoshi, "Design and optimization of a 2-degree-of-freedom planar remote center of motion mechanism for surgical manipulators with smaller footprint," *Mechanism and Machine Theory*, vol. 129, pp. 148–161, 2018.

[7] J. Smits, D. Reynaerts, and E. V. Poorten, "Setup and Method for Remote Center of Motion Positioning Guidance During Robot-Assisted Surgery," in *5th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, 2019, pp. 1315–1322.

[8] U. Hagn, T. Ortmaier, R. Konietschke, B. Kubler, U. Seibold, A. Tobergte, M. Nickl, S. Jorg, and G. Hirzinger, "Telemanipulator for remote minimally invasive surgery," *IEEE Robotics & Automation Magazine*, vol. 15, no. 4, pp. 28–38, 2009.

[9] C. D. Pham, F. Coutinho, A. C. Leite, F. Lizarralde, P. J. From, and R. Johansson, "Analysis of a moving remote center of motion for robotics-assisted minimally invasive surgery," in *IEEE/RSJ International Conference on Intelligent Robots & Systems*, 2015, pp. 1440–1446.

[10] B. Dahroug, B. Tamadazte, and N. Andreff, "3D Path Following with Remote Center of Motion Constraints," in *International Conference on Informatics in Control, Automation and Robotics*, 2016.

[11] H. Su, C. Yang, G. Ferrigno, and E. D. Momi, "Improved Human–Robot Collaborative Control of Redundant Robot for Teleoperated Minimally Invasive Surgery," *IEEE Robotics & Automation Letters*, vol. 4, no. 2, pp. 1447–1453, 2019.

[12] J. Arévalo, G. Poisson, and P. Vieyres, "A New Kinematic Formulation of the RCM Constraint for Redundant Torque-Controlled Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 4576–4581.

[13] J. Zhao, Y. Zhou, M. Guo, T. Yu, and H. Liu, "A Novel Remote Center of Motion Algorithm Based on 3D Force Dragging for Minimally Invasive Surgical Robot," in *2020 IEEE International Conference on Mechatronics and Automation*, 2020, pp. 1647–1652.

[14] M. A. Nasseri, P. Gschirr, M. Eder, S. Nair, K. Kobuch, M. Maier, D. Zapp, C. Lohmann, and A. Knoll, "Virtual fixture control of a hybrid parallel-serial robot for assisting ophthalmic surgery: An experimental study," in *5th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, 2014, pp. 732–738.

[15] M. A. Nasseri, M. Eder, D. Eberts, S. Nair, M. Maier, D. Zapp, C. P. Lohmann, and A. Knoll, "Kinematics and dynamics analysis of a hybrid parallel-serial micromanipulator designed for biomedical applications," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2013, pp. 293–299.

[16] C. Yang, Y. Jiang, Z. Li, W. He, and C. Su, "Neural Control of Bimanual Robots With Guaranteed Global Stability and Motion Precision," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1162–1171, 2017.

[17] H. Yu, T. Xie, S. Paszczynski, and B. M. Wilamowski, "Advantages of Radial Basis Function Networks for Dynamic System Design," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 12, pp. 5438–5450, 2011.

[18] S. S. Haykin and R. Gwynn, *Neural Networks and Learning Machines*. China Machine Press, 2009.

[19] C. He, N. A. Patel, A. Ebrahimi, M. Kobilarov, and I. Iordachita, "Preliminary study of an RNN-based active interventional robotic system (AIRS) in retinal microsurgery," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 14, no. 6, pp. 945–954, 2019.