Online Trajectory Planning in Dynamic Environments for Surgical Task Automation

Takayuki Osa, Naohiko Sugita, and Mitsuishi Mamoru
Department of Mechanical Engineering, The University of Tokyo, Japan
Email: {osa, sugi, mamoru}@nml.t.u-tokyo.ac.jp

***Abstract*—Automation of robotic surgery has the potential to improve the performance of surgeons and the quality of the life of patients. However, the automation of surgical tasks has challenging problems that must be resolved. One such problem is adaptive online trajectory planning based on the state of the surrounding dynamic environment. This study presents a framework for online trajectory planning in a dynamic environment for automatic assistance in robotic surgery. In the proposed system, a demonstration under various states of the environment is used for learning. The distribution of the demonstrated trajectory over the environmental conditions is modeled using a statistical model. The trajectory, under given environmental conditions, is computed as a conditional expectation using the learned model. Because of its low computational cost, the proposed scheme is able to generalize and plan a trajectory online in a dynamic environment. To design the motion of the system to track the planned trajectory in a stable and smooth manner, the concept of a sliding mode control was employed; its stability was proved theoretically. The proposed scheme was implemented on a robotic surgical system and the performance was verified through experiments and simulations. These experiments and simulations verified that the developed system successfully planned and updated the trajectories of the learned tasks in response to the changes in the dynamic environment.**

I. Introduction

As the clinical performance of robotic surgery has demonstrated, it will become a common procedure. For instance, the da Vinci system (Intuitive Surgical Inc., CA, US) has proved its performance in many clinical studies and is installed in thousands of hospitals worldwide [5]. However, robotic surgery still has significant potential to improve its clinical performance. Many studies have been conducted to develop more intelligent and sophisticated robotic surgical systems to improve the quality of robotic surgery.

One of the research topics in this area is surgical task automation. It is believed that automatic assistance by a robotic system in a surgical operation would reduce a surgeon's fatigue and operation time. However, the automation of surgical tasks has challenging problems that need to be resolved. One of these is the adaptation of the trajectory according to the state of the dynamic environment.

In surgical task automation, the robotic surgical system is expected to cooperate with surgeons and work with flexible and movable objects such as threads, needles, and soft tissues. Therefore, the robotic system must adapt its motion according to the motion of the surgeon, surgical instruments, and target organ.

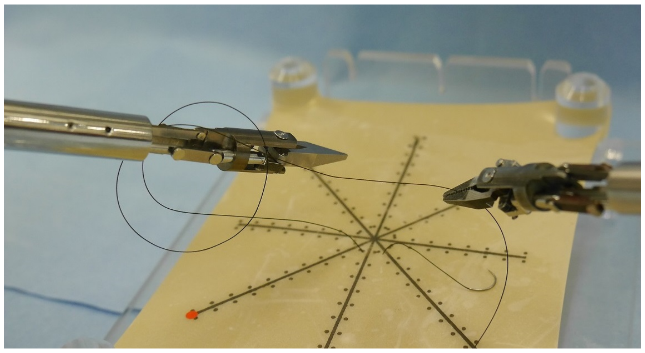


Fig. 1. Surgical knot tying involves making loops around a surgical instrument. If the surgical instrument moves during the process, the trajectory must be adapted online.

For example, surgical knot tying involves making loops around a surgical instrument using a thread held by another instrument. In this looping task, the trajectories for making the loop must be adapted according to the position of the surgical instrument, maintaining the topological feature of the trajectory (Fig. 1). If the surgical instrument to be looped is moving during the task, the trajectory must be recalculated and adapted online according to the motion. This kind of adaptive trajectory planning is required for automatic assistance in many situations. However, this online planning and adaptation of the trajectory is very challenging, and the solution for this problem has not yet been established.

In this paper, we present a method for learning time-and space-dependent trajectories from demonstrations, and planning and updating trajectories online adapting to changes in the dynamic environment. In the proposed scheme, demonstration trajectories, under various environmental conditions, are used for learning the task trajectory process. The demonstrated trajectories are normalized in a time domain, and their distribution over the environmental condition is modeled using a statistical method. Using the learned models, the learned tasks are generalized to the new state of the environment. Because of its low computational cost, this method is able to plan and update trajectories online adapting to changes in the environment. On the other hand, the update of the planned trajectories can also cause a discontinuous change of the planned trajectory. If the planned trajectory is input to the system directly, the discontinuous change could cause unstable system behavior. Therefore, we present a scheme to design the motion of the slave manipulator to track the planned trajectory in a smooth and stable fashion using the concept of a sliding mode control. The proposed scheme was implemented on a robotic surgical system and the performance of the developed system was verified through experiments and simulations.

This paper is structured as follows. The next section describes the previous studies related to trajectory planning by learning from demonstrations. Section III describes the details of the proposed method. Section IV provides the experiments and simulations to evaluate the developed system. Section V discusses the results presented in this study. The conclusions and outline of future work can be found in Section VI.

II. Related Studies

Many studies related to the automation of surgical tasks have been published [19, 9, 10, 11]. However, to the best of our knowledge, no existing surgical robotic system plans and updates a trajectory online to adapt to changes in the environment. Van der Berg et al. developed a system that learns surgical knot tying from demonstrations and executes the learned task faster than the demonstrations [19]. However, the system cannot adapt the planned trajectory if the initial conditions change. Mayer et al. proposed approaches to learn surgical tasks from demonstrations [9, 10, 11]. Although a scheme for generalizing the learned task to a new situation is presented in [10], the scheme cannot be applied to online trajectory planning because of its computational costs and limitations in modeling the situation for simulations. Schulman et al. [16, 17] presented the most related works. They proposed a scheme to generalize demonstrated trajectories to a new situation, and they achieved automatic suturing in new situations in a simplified setup [16]. However, it is expected that the computational cost will be considerably high for online trajectory planning in a dynamic environment.

In the field of motion planning, the programming by demonstration (PbD) approach has been investigated by many researchers [1]. One notable scheme is the movement primitive approach proposed by Schaal et al. [15, 12]. They achieved generalizing task trajectories to various start and end-points by learning from demonstrations. However, this is not applicable to tasks where the topological features of the trajectory must be adapted according to the state of the environment. Another important previous work is presented by Billard et al. [7, 6, 4]. They developed a scheme of learning tasks as a time-invariant dynamic system and adapting the motion trajectory in real-time for obstacle avoidance [6]. However, this scheme is not applicable to time-dependent trajectories that are often necessary for surgical task automation.

In this paper, we present a system that learns time-and space-dependent trajectories from demonstrations in various environmental conditions and plans a trajectory online in the dynamic environment. In addition, this study presents a control scheme to allow the system to track the planned and updated trajectories online, in a stable manner. The combination of the proposed trajectory-planning and trajectory-tracking schemes enables us to develop a system that can adapt its motion online according to a change in the environment.

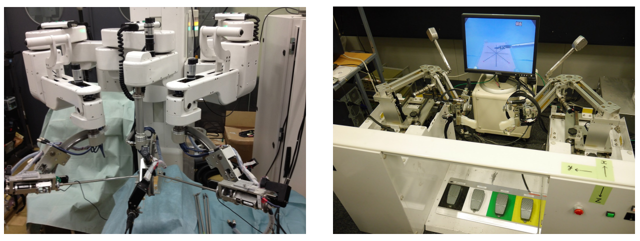

(a) (b)

Fig. 2. Robotic telesurgery system: (a) slave and (b) master sites

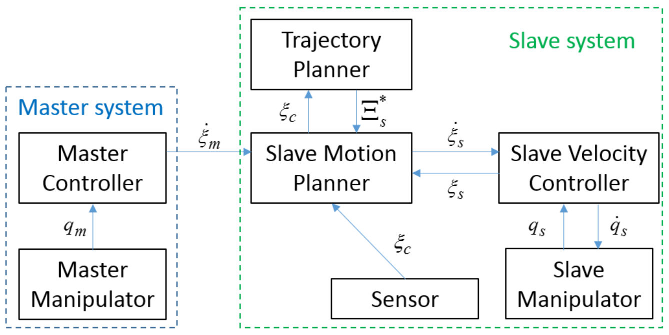


Fig. 3. System architecture.

III. Method

A. Overview of the proposed method

We designed a method for the automatic execution of learned tasks in a robotic tele-surgical system. The configuration, shown in Fig. 2, consists of a master and slave manipulator. This is a standard robotic surgical system setup. The designed system architecture is summarized in Fig. 3.

Let us define ξ as the state of the system. Here, ξ can be task space variables or configuration space variables. To avoid the loss of generality, we do not specify a clear definition of ξ. When a human operator at the master site controls the slave manipulator, a motion at the master manipulator,$\dot{\xi}_m$, is transmitted to the slave system using socket communication. The slave motion planner scales the received motion signal and converts it into a motion of the slave manipulator, $\dot{\xi}_s$ as follows:

$$\dot{\xi}_s = K_{ms}\dot{\xi}_m \tag{1}$$

where K_{ms} is a positive diagonal matrix that represents the scaling relationship between the master manipulator and the slave manipulator. Then, $\dot{\xi}_s$ is sent to the slave velocity controller. This resolves the inverse kinematics and controls the servomotors of the slave manipulator. The slave velocity controller controls the individual joint velocity of the slave manipulator, $\dot{q}_s$, to track the motion input $\dot{\xi}_s$.

When the robotic system executes the learned automated task, the trajectory planner plans trajectories for the automated tasks, $\Xi_s^* = [\xi_s^*(0), \ldots, \xi_s^*(T)]$ where T is the length of the

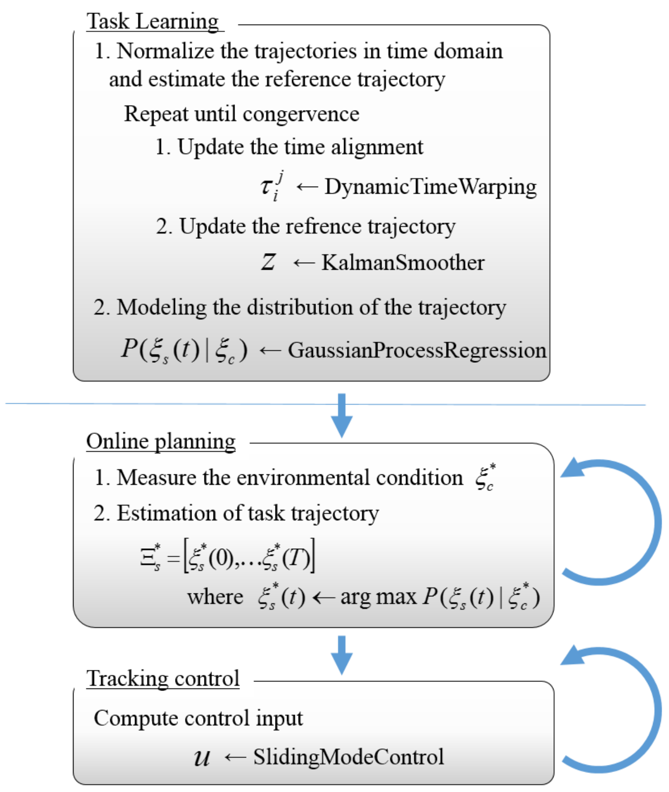

Task Learning
1. Normalize the trajectories in time domain and estimate the reference trajectory
Repeat until congervence
1. Update the time alignment
$\tau_i^j \leftarrow$ DynamicTimeWarping
2. Update the refrence trajectory
$z \leftarrow$ KalmanSmoother
2. Modeling the distribution of the trajectory
$P(\xi_s(t) \mid \xi_c) \leftarrow$ GaussianProcessRegression
Online planning
1. Measure the environmental condition ξ_c^*
2. Estimation of task trajectory
$\Xi_s^* = [\xi_s^*(0),\ldots\xi_s^*(T)]$
where $\xi_s^*(t) \leftarrow \arg\max P(\xi_s(t) \mid \xi_c^*)$
Tracking control
Compute control input
$u \leftarrow$ SlidingModeControl

Fig. 4. Overview of the proposed scheme.

planned task. The values measured at the slave site, ξ_c, are constantly sent to the trajectory planner as variables representing the environmental condition. The trajectory planner plans the trajectory for the automated tasks online according to ξ_c. The motion planner determines the motion input to the slave velocity controller, $\dot{\xi}_s$, such that the system tracks the planned trajectory Ξ_s^* in stable manner.

An overview of the proposed scheme for the online trajectory planning that is implemented in the trajectory planner is shown in Fig. 4. The goal of the online trajectory planning in this study is to plan a trajectory $\Xi_s^*(t)$ in the dynamic environment condition ξ_c^*, in real time. The environment condition, ξ_c^*, refers to the state of the environment, such as the position of a surgical instrument, and the position and posture of a surgical needle.

In the proposed method, we model the spatial distribution of the trajectory over the environmental condition at time t as $P(\xi_s(t) \mid \xi_c^*)$. Then, the optimal trajectory under the condition ξ_c^*, Ξ_s^*, can be estimated as a conditional expectation as follows:

$$\Xi_s^* = [\xi_s^*(0), \ldots, \xi_s^*(T)] \tag{2}$$

where

$$\xi_s^*(t) = \mathrm{argmax} P(\xi_s(t) \mid \xi_c^*) \tag{3}$$

In a learning task phase, the system normalizes the demonstrated trajectories in the time domain and estimates the reference trajectory. In this step, Dynamic Time Warping (DTW) is employed to normalize the demonstrated trajectories in the time domain, and the Kalman smoother is employed to estimate the reference trajectory [14, 19]. Then, the spatial distribution of the demonstrated trajectories over the condition ξ_c is modeled using Gaussian Process Regression (GPR) [13].

During the execution of the learned task, the trajectory is planned and updated online according to ξ_c using the learned models. When the planned trajectory is updated, the planned trajectory changes discontinuously because of the computation time even though the computational cost is very small. Because of such discontinuous change of the planned trajectory, the planned trajectory cannot be input to the slave velocity controller directly. This would cause unstable system behavior. Therefore, we employ the concept of a sliding mode control to determine the motion of the slave manipulator $\dot{\xi}_s$ such that the system tracks the planned trajectory in a smooth and stable manner.

B. Learning and Planning a Trajectory

1) Normalization in the Time domain and Estimation of a Reference Trajectory: Before modeling the spatial distribution of the trajectory over the environment condition, we must normalize the demonstrated trajectories in the time domain. We do this because the execution speed of the trajectory varies for each demonstration. To normalize the demonstrated trajectories in the time domain and estimate the reference trajectory, we use the method described in [19]. As in [19], we regard the demonstrated trajectory as a noisy "observation" of the "reference " trajectory.

We express the state of the discrete system at time t as follows:

$$z(t) = \begin{bmatrix} \xi(t) \\ \dot{\xi}(t) \end{bmatrix} \tag{4}$$

where $\xi(t)$ is the state of the robotic system at the t th time step. The system can be expressed as a linear system with process noise and measurement noise, and we assume that the process noise and measurement noise are distributed as a Gaussian distribution. Therefore, we can express the system as follows:

$$z(t+1) = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} z(t) + w(t) \tag{5}$$

where $w(t) \sim N\left(\mathbf{0}, \begin{bmatrix} P & 0 \\ 0 & Q \end{bmatrix}\right)$, and P and Q represent the process noise and measurement noise, respectively.

Then, we can write the relationship between the reference trajectory z and demonstrated trajectories $y^1, \ldots, y^M$ as follows:

$$\begin{bmatrix} y^1(\tau_t^1) \\ \vdots \\ y^M(\tau_t^M) \end{bmatrix} = \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix} z(t) + v(t),\ v(t) \sim N\left(\mathbf{0}, \begin{bmatrix} R^1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & R^M \end{bmatrix}\right) \tag{6}$$

where τ^j is the time mapping from the reference trajectory z to the demonstrated trajectory y^j and R^j is the variance of the j th trajectory from the reference trajectory. Thus, we can use the Kalman smoother to estimate the reference trajectory

z. In (6), we set $R^j = I$ for $t = 0, \ldots, M$ so that all of the demonstrations contribute equally to the estimation of the reference trajectory.

To estimate the reference trajectory at the initial step, we initialized the time alignment τ as follows:

$$\tau_t^j = \frac{t}{N} T^j \tag{7}$$

where τ_t^j is the time alignment of the tth step of the jth demonstration, N is the total number of time steps of the reference trajectory, and T^j is the length of the jth demonstration.

Then, the time alignment τ_t^j is updated for $t = 0, \ldots, N$ and $j = 0, \ldots, M$ with DTW using the reference trajectory as the norm in the time domain [14]. In DTW, we employ the following constraint for the time alignment:

$$\frac{1}{2}\left(\tau_{t+1}^{old} - \tau_t^{old}\right) \le \tau_{t+1}^{update} - \tau_t^{update} \le 2\left(\tau_{t+1}^{old} - \tau_t^{old}\right) \tag{8}$$

where τ_t^{old} is the time alignment before the update and τ_t^{update} is the updated time alignment. This constraint means that the DTW changes the speed of the motion between half-speed and double-speed compared with the speed before the DTW process. This constraint avoids computing an unnatural time alignment.

After the time alignment is updated, the reference trajectory is also updated using the updated time alignment. In this processing loop, the updating of the time alignment and reference trajectory is repeated until the reference trajectory converges. In our cases, this loop was repeated once or twice until convergence.

2) Modeling the distribution of the trajectories over the environmental condition: Using the trajectories normalized in the time domain, we model the spatial distribution of the trajectory over the environment condition. For this purpose, we employ GPR [13]. GPR is a non-parametric method for regression. It models a joint distribution of the data without directly modeling a regression function. Although there are other options for regression, such as Gaussian Mixture Regression [2, 7] and Locally Weighted Regression [3], we chose GPR because it can regress the nonlinear relationship globally using relatively little training data. We express the dataset of the demonstrations and the given environmental conditions as follows:

$$X = \begin{bmatrix} \left(\xi_c^1\right)^T \\ \vdots \\ \left(\xi_c^M\right)^T \end{bmatrix},\ Y = \begin{bmatrix} \left(\xi_s^1(\tau_i^1) - z(i)\right)^T \\ \vdots \\ \left(\xi_s^M(\tau_i^M) - z(i)\right)^T \end{bmatrix},$$
$$X^* = \left[\left(\xi_c^*\right)^T\right], Y^* = \left[\left(\xi_s^*(i) - z(i)\right)^T\right] \tag{9}$$

where X is the set of the environmental conditions in the demonstrations, Y is the set of deviations of the demonstrated trajectories from the reference trajectory at the tth step, X^* is the new environmental condition given for a trajectory estimation, and Y^* is the estimated deviation from the reference trajectory under the environmental condition X^* as the tth step. In GPR, the distribution of Y can be modeled as a multivariate as follows:

$$P\begin{pmatrix} y_j \\ y_j^* \end{pmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X)+\sigma_n^2 I & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*)+\sigma_n^2 I \end{bmatrix}\right) \tag{10}$$

where $Y = [y_1, \ldots, y_n]$ and K is the kernel matrix defined as follows:

$$\begin{aligned} K(X,X) &\in \mathbf{R}^{M\times M},\ (K(X,X))_{i,j} = k(\xi_c^i, \xi_c^j) \\ K(X^*,X) &\in \mathbf{R}^{1\times M},\ (K(X^*,X))_{1,j} = k(\xi_c^*, \xi_c^j) \\ K(X,X^*) &\in \mathbf{R}^{M\times 1},\ (K(X,X^*))_{i,1} = k(\xi_c^i, \xi_c^*) \\ K(X^*,X^*) &\in \mathbf{R},\ K(X^*,X^*) = k(\xi_c^*, \xi_c^*) \end{aligned} \tag{11}$$

In (11), $k(x_i, x_j)$ represents the kernel function. In the developed system, we employ the squared exponential kernel function defined as [13]:

$$k(x_i, x_j) = \sigma_f \exp\left(-\frac{1}{2l^2}(x_i - x_j)^T(x_i - x_j)\right) \tag{12}$$

The performance of the GPR depends on the selection of the hyperparameters $[\sigma_f, \sigma_n, l]$. These hyperparameters can be obtained by maximizing the marginal likelihood defined as follows:

$$\log p = -\frac{1}{2} y^T K y - \log\det|K| - n\log 2\pi \tag{13}$$

The optimization problem of maximizing the marginal likelihood can be solved using gradient-based optimization methods. This optimization problem often falls into local optima, and therefore, we perform the optimization from randomly chosen initial conditions to find the optimal hyperparameters. The hyperparameters are dependent on the training dataset, namely, X and Y, and are not dependent on the new environmental condition X^*. Thus, the optimization of the hyperparameters is performed only before the execution of the task.

The joint distribution of the trajectory under the given environmental condition can be modeled as a Gaussian distribution as follows:

$$y_j^* | y_j, X, X^* \sim \mathcal{N}(\mu^*, \Sigma^*) \tag{14}$$

where

$$\mu^* = K(X^*,X)(K(X,X) + \sigma_n^2 I)^{-1} y_j \tag{15}$$
$$\begin{aligned} \Sigma^* &= K(X^*,X^*) + \sigma_n^2 I \\ &- K(X^*,X)(K(X,X) + \sigma_n^2 I)^{-1} K(X,X^*) \end{aligned} \tag{16}$$

Therefore, the trajectory under the given environmental condition can be estimated as follows:

$$\begin{aligned} \xi_s^*(t) &= z(t) + \arg\max P(Y^*|Y,X,X^*) \\ &= z(t) + \left(K(X^*,X)(K(X,X) + \sigma_n^2 I)^{-1} Y\right)^T \end{aligned} \tag{17}$$

where $\xi_s^*(t)$ is the state of the system at the tth step in the estimated trajectory for the given environmental condition. The complete trajectory $\Xi^* = [\xi_s^*(0), \ldots, \xi_s^*(N)]$ can be obtained by computing (17) for $i = 0, \ldots, N$.

In this scheme, once $K(X,X)$ and σ_n are obtained in the offline phase, the only computation required for online trajectory planning is computing (17) for $i = 0, \dots, N$, which is a matrix calculation with relatively small computational cost. This feature enables a trajectory planning in a dynamic environment with sufficiently short computation time.

C. Motion Control Scheme

When the planned trajectory is updated during the automation task, it changes discontinuously. However, if the motion input to the slave manipulator changes discontinuously, it could cause unstable system behavior. Therefore, the motion input to the slave manipulator must be designed to be smooth to ensure that the system tracks the updated trajectory in stable manner.

For this purpose, we employ the concept of sliding mode control [18]. In this case, the control input to the system is the velocity of the system. Therefore, the system can be expressed by a linear system as follows:

$$\dot{\xi}_s = u \tag{18}$$

In the framework of the sliding mode control, the desired system behavior is expressed as a sliding surface as follows:

$$s(t) = \xi_s(t) - \xi_s^*(t) = 0 \tag{19}$$

where $\xi_s(t)$ is the state of the system at time t and $\xi_s^*(t)$ is the desired state at time t obtained from the updated planned trajectory.

We must design the motion input, u, so that the system robustly slides on the sliding surface. In this system, we employ the following control law:

$$\begin{aligned} u(t) &= \hat{u} - K \cdot \mathrm{sat}\left(s(t)\right) \\ &= \dot{\xi}_s^* - K \cdot \mathrm{sat}\left(\xi_s(t) - \xi_s^*(t)\right) \end{aligned} \tag{20}$$

where $K = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & k_n \end{bmatrix}$ and $k_i > 0$ for $i = 1, \dots, n$.

The saturation function $\mathrm{sat}(s)$ is expressed as follows:

$$\mathrm{sat}(s) = \begin{bmatrix} \mathrm{sat}(s_1) \\ \dots \\ \mathrm{sat}(s_n) \end{bmatrix} \tag{21}$$

where

$$\mathrm{sat}(s_i) = \begin{cases} 1 & (s_i > c_i) \\ s_i/c_i & (|s| < |c_i|) \\ -1 & (s_i < -c_i) \end{cases} \tag{22}$$

where c_i is a positive constant value. In (20), the first term represents the control input to make the system slide on the sliding surface; the second term represents the control input to make the system converge on the sliding surface. The use of the saturation function $\mathrm{sat}\,(x)$ is a common solution to avoid a “chattering” problem common to the sliding mode control. Furthermore, the control input defined as (20) is upper-bounded; it avoids excessive change of the control inputs that could cause unstable system behavior.

In the framework of the sliding mode control, it is known that the system converges to the sliding surface in a finite time if the following inequality is satisfied for a certain $\eta = [\eta_0, \dots, \eta_n]$:

$$\frac{1}{2}\frac{d}{dt}s_i^2 < -\eta_i\left|s_i\right| \tag{23}$$

where $s = [s_0, \dots, s_n]^T = [\xi_0 - \xi_0^*, \dots, \xi_n - \xi_n^*]^T$ and n is the dimension of the system [18].

For the range of $|s_i| > c_i$, the condition in (23) can be briefly proved as described below. From (19) - (20), we can obtain the following equations.

$$\begin{aligned} \frac{1}{2}\frac{d}{dt}s_i^2 &= \dot{s}_i \cdot s_i \\ &= \left(\dot{\xi}_{s,i}(t) - \dot{\xi}_{s,i}^*(t)\right) s_i \\ &= \left(u(t) - \dot{\xi}_{s,i}^*(t)\right) s_{s,i} \\ &= -k_i\left(\xi_{s,i}(t) - \xi_{s,i}^*(t)\right) s_i \\ &= -k_i\left|s_i\right| \end{aligned} \tag{24}$$

From line 3 to line 5, we use (20). Therefore, using the control law in (20), the condition in (23) is satisfied, and the system converges to the neighbor of the updated trajectory in a finite time.

Meanwhile, in the range of $|s_i| > c_i$, the following condition is satisfied:

$$\begin{aligned} \dot{s}_i &= \dot{\xi}_{s,i} - \dot{\xi}_{s,i}^* \\ &= u - \dot{\xi}_{s,i}^* && (25) \\ &= -k_i \mathrm{sat}(s_i) && (26) \\ &= -k_i s_i && (27) \end{aligned}$$

Thus, the system exponentially converges to the sliding surface in the neighborhood of the sliding surface. Therefore, it converges from any state to the desired trajectory in a finite time and tracks in stable manner using the control law in (20).

IV. Experiments and Simulations

The proposed scheme was implemented on the robotic system for remote surgery shown in Fig. 2. In the robotic system, two robotic surgical instruments are attached to two robotic arms; one laparoscope is attached to another robotic arm. We performed the experiments and simulations to evaluate the proposed scheme. In the experiments and simulations, the system learned two tasks, namely, a DOUBLE LOOP task and a PICK AND PULL task. Seven demonstrations were performed for each of the tasks under various environmental conditions. The environmental conditions were static during a demonstration. In the simulation, the demonstrated trajectories were separated into six demonstrations as a training dataset and one demonstration for the validation data. The demonstration used for the validation data had a different environmental condition from that of the training dataset, and trajectories demonstrated and planned in the same environmental condition were compared. To quantify the correctness and smoothness of the planned trajectories, the RMS error between the demonstrated trajectory and the planned trajectory, $1/T \sum \left\|\xi_s^d - \xi_s^*\right\|$, and the norm of jerk, $1/T \sum_{t=0}^{T} \left\|\dddot{\xi}_s(t)\right\|$, were computed. After

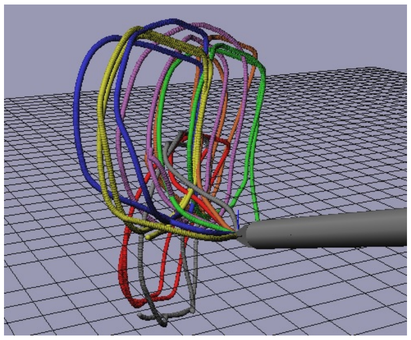


Fig. 5. Dataset of demonstrated trajectory for the DOUBLE LOOP task.

the simulation, experiments were performed to evaluate the developed system's ability to plan trajectories in a dynamic environment, where the environmental condition changed during the automated motions.

A. DOUBLE LOOP task

The DOUBLE LOOP task involves making a loop around the left robotic surgical instrument with a thread held by the right robotic surgical instrument. This task was designed to demonstrate the trajectory generation for a time- and space-dependent task in which the topological shape of the trajectory had to adapt to the environmental conditions.

In this task, the environment condition ξ_c was the position of the tip of the left instrument as follows:

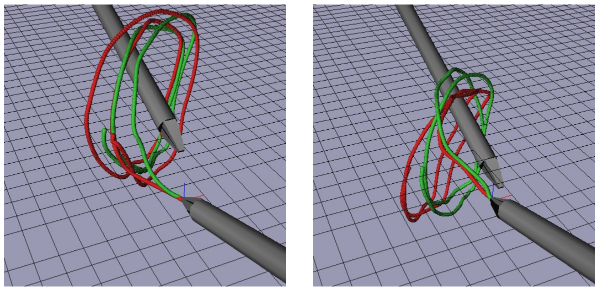
$$\xi_c = \xi_l = [x_l, y_l, z_l]^T \tag{28}$$

The position of the tip of the left instrument ξ_l is expressed in the coordinates of the base of the left robotic arm. The transformation from the coordinates of the left robotic instrument to the coordinates of the right robotic instrument was unknown.

The demonstration trajectory data set in this experiment is shown in Fig. 5. The demonstration trajectories and trajectories planned in the same environmental conditions are shown in Fig. 6. The average of the RMS errors between the demonstrated trajectories and planned trajectories was 27.9 mm. As shown in Fig. 6, the learned task trajectories were successfully generalized to the new environmental conditions. In the simulation, the average of the jerk of the demonstrated trajectories was 1.58, and the average of the planned trajectories was 1.21 [1]. Because of the statistical model of the task, small deviations seen in the demonstrations were not observed in the planned trajectories, and the planned trajectories are smoother than demonstrated trajectories. The computation time for the planning trajectory was 85-90 ms using a 64-bit machine with an Intel Core i7-4600U CPU 2.1 GHz.

Next, the proposed scheme was implemented in the robotic manipulator, and the performance of the developed system was tested by performing the DOUBLE LOOP task in a dynamic environment. In the experiment, the left robotic instrument was controlled by a human operator from the master site. The position of the tip of the left robotic instrument was moved to intentionally disrupt the automatic task execution while the right robotic instrument was moving automatically to perform the task.

[1]The unit of jerk is omitted because it was computed in the normalized time domain.



(a) (b)

Fig. 6. Planned trajectories for DOUBLE LOOP task. (a) and (b) are pictures from the same viewpoint. Red and green trajectories represent the demonstrated trajectories and planned trajectories, respectively. Diameter of the surgical instrument is 10mm.

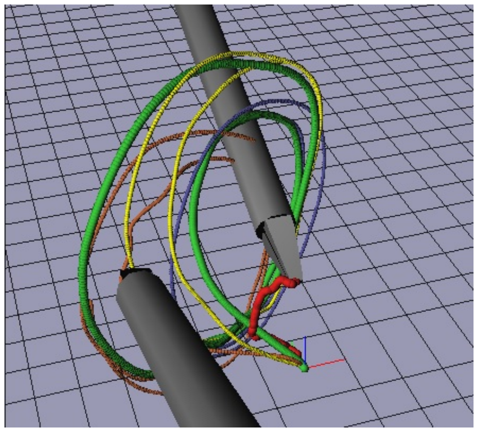


Fig. 8. Visualization of planned trajectories and an executed trajectory of the LOOPING task. Blue, orange, and yellow dots represent the trajectory planned at the beginning of the task, the middle of the task, and the end of the task, respectively. The green and red dots represent the trajectory executed by the right and left hands, respectively.

Figure 7 shows the procedure and results of the experiment. In the experiment, the system adapted the motion of the right robotic instrument and successfully performed the DOUBLE LOOP task. The system planned and updated trajectories 156 times in 18 seconds during the task. The trajectories planned during the task are visualized in Fig. 8. As shown, the trajectory was adapted according to the position of the left surgical instrument. In addition, the system tracked the planned and updated trajectories in a stable manner during the task.

B. PICK AND PULL task

The PICK AND PULL task involves picking and pulling surgical needles held by another surgical instrument. This task was designed to demonstrate the adaptation of a time-dependent trajectory, where the surgical instrument passes the

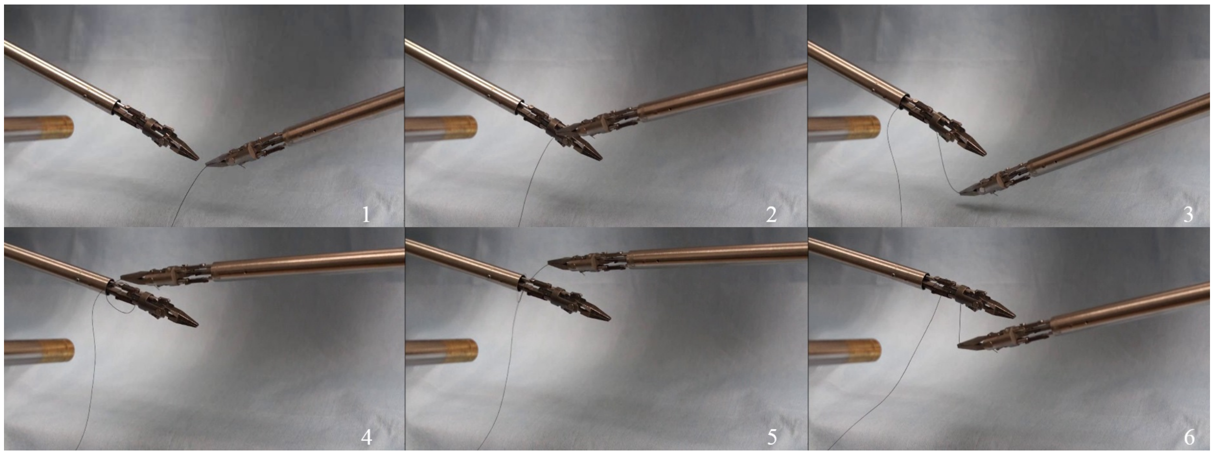


Fig. 7. Execution of DOUBLE LOOP task. The task was executed from the left-hand side to the right-hand side. To disrupt the task, the left robotic instrument was moved to the left at the third figure, and moved upward at the fourth figure. Thereafter, the trajectory was adapted to the new position of the left robotic instrument, and the thread was wound around the left robotic instrument successfully.

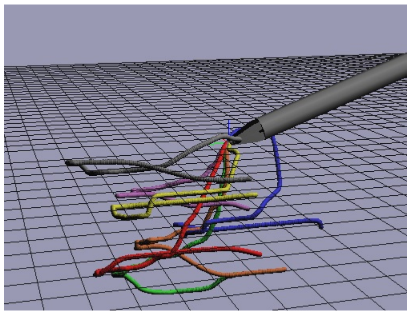


Fig. 9. Dataset of demonstrated trajectory for the PICK AND PULL task.

same area in different directions. In the PICK AND PULL task, the position of the tip of the surgical instrument that held the surgical needle was measured using 3D reconstruction with a stereo web camera. The environment condition ξ_c was the position of the tip of the instrument in the image coordinates as follows:

$$\xi_c = p_{ins}^{image} = [x_{ins}^{image}, y_{ins}^{image}, z_{ins}^{image}]^T \tag{29}$$

The tip of the surgical instrument that held the surgical needle was tracked with a KLT tracker implemented using OpenCV [2][8]. Although stereo calibration was performed for the stereo web camera, the transformation from the image coordinates to the coordinates of the robotic instrument was unknown.

The demonstrated trajectory dataset is shown in Fig. 9. The demonstration trajectories and trajectories planned in the same environmental conditions are shown in Fig. 10. The average of the RMS errors between the demonstrated trajectories and planned trajectories was 11.4 mm. As shown in Fig. 10, the learned task trajectories were generalized successfully to the new environmental condition. In the simulation, the average of the norm of jerk of the demonstrated trajectories was 2.01, and the average of the norm of jerk of the planned trajectories was 1.80 [3]. The computation time for the planning trajectory was 50-80 ms when using a 64-bit machine with an Intel Core i7-4600U CPU 2.1 GHz.

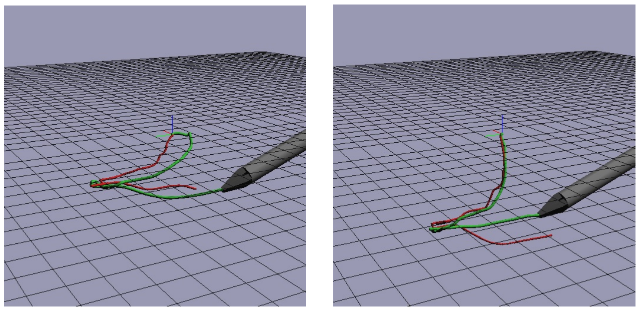


Fig. 10. Trajectories planned for PICK AND PULL task. (a) and (b) are pictures from the same view point. Red and green trajectories represent demonstrated trajectories and planned trajectories, respectively.

Next, the proposed scheme was implemented in the robotic manipulator, and the performance of the developed system was tested by performing the PICK AND PULL task in a dynamic environment. To disrupt the task, the surgical instrument was bent by hand, and the position of the surgical needle was moved intentionally during the task.

Figure 11 shows the procedure and results of the experiment. As shown, although the position of the surgical instrument was moved immediately before picking the surgical needle, the motion was adapted to the new position of the surgical instrument, and the robotic surgical instrument successfully picked the surgical needle. The system planned and updated trajectories 168 times in 15 seconds during the task. The trajectories planned in the experiment are visualized in Fig. 12 along with the trajectory that was actually executed by the robotic manipulator in the experiment. Figure 12 shows

[2]http://docs.opencv.org/

[3]The unit of jerk is omitted because it was computed in the normalized time domain.

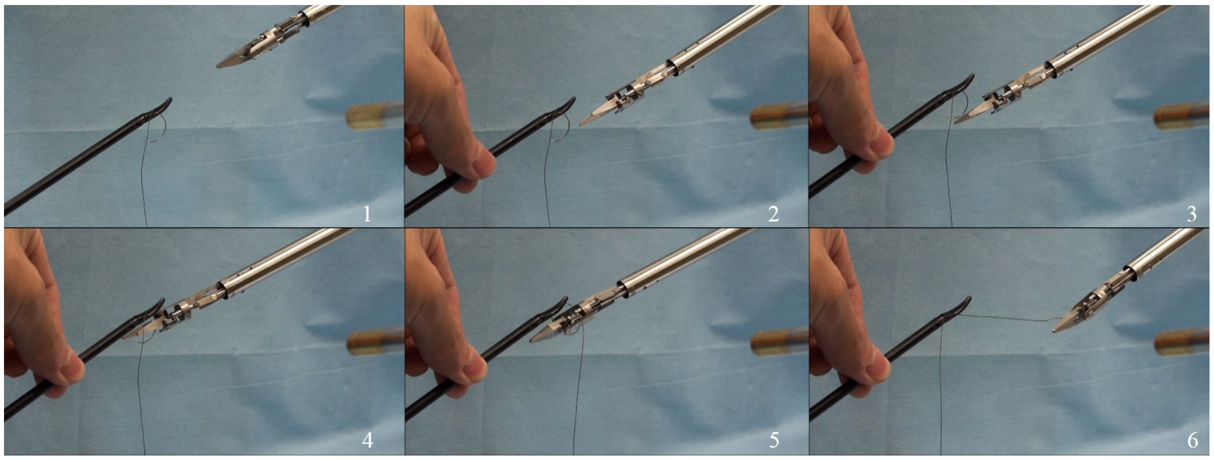


Fig. 11. Execution of PICK AND PULL task. The task was executed from left to right. At the third figure, the surgical instrument was bent and the position of the surgical instrument and the needle was moved. Thereafter, the trajectory was adapted to the new position of the needle, and the robotic instrument successfully picked the needle.

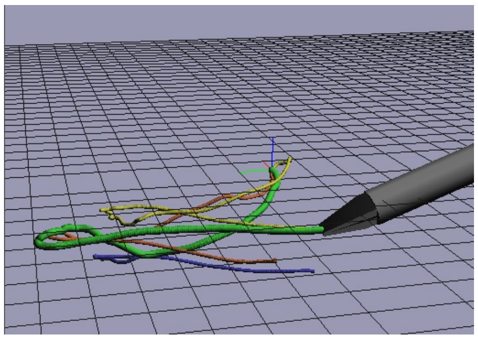


Fig. 12. Visualization of planned trajectories and an executed trajectory of the PICK AND PULL task. Blue, orange, and yellow dots represent the trajectory planned at the beginning of the task, the middle of the task, and the end of the task, respectively. The green dots represent the trajectory executed by the left hands.

that the trajectories are updated online and that the system tracked the updated trajectory in a stable manner during the execution of the learned task.

V. Discussion

In the proposed scheme, the trajectory is planned based on the modeled distribution of the demonstrated trajectory. Therefore, the planned trajectory is valid only in the range where the modeled distribution is appropriate. Thus, to obtain a satisfactory performance in a given area, the demonstrations have to be performed under sufficiently various conditions.

In the experiments, the dimension of environmental condition ξ_c was set to three. Owing to this low dimensionality, the distribution over the environmental condition could be modeled using a relatively small number of demonstrations. However, if the dimension of the environmental condition is high, the number of demonstrations that the system requires for learning the task will increase. In addition, since the environmental condition is defined in Cartesian space, the relationship between the environmental condition and the deviation of the trajectory is expected to be almost linear in this study. If the relationship between the environmental condition and the deviation of the trajectory is nonlinear, for example, in the case where the environmental condition is defined in configuration space, more demonstrations will be required for learning the task.

VI. Conclusion

This study presented a framework for online trajectory planning in a dynamic environment and showed the performance of the proposed scheme through preliminary experiments and simulations. In the proposed scheme, demonstrations in various environmental conditions were used for learning task trajectories. The demonstrated trajectories were normalized in the time domain using DTW, and the distribution of the trajectory over the environmental condition was modeled statistically using GPR. The trajectory under the given environmental condition was estimated as a conditional expectation using the trained GPR model. Because of its low computational cost, the proposed scheme of planning trajectories was able to plan and update the trajectory in a dynamic environment. To ensure that the system tracked the updated trajectories in a stable manner, we presented a scheme for designing the motion of the slave manipulator using a concept of the sliding mode control. The proposed scheme was implemented in a robotic surgical system, and the performance of the developed system was tested through experiments and simulations. These experiments and simulations verified that the system planned and adapted the task trajectory online in response to a change in the dynamic environment.

Acknowledgments

This work was supported by JSPS Grant-in-Aid for Scientific Research(S) 23226006 and Grant-in-Aid for JSPS fellows Grant Number 25-7106.

References

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57:469–483, 2009. ISSN 0921-8890.

[2] S. Calinon, F. D'halluin, E.L. Sauser, D.G. Caldwell, and A.G. Billard. Learning and reproduction of gestures by imitation. *Robotics Automation Magazine, IEEE*, 17:44–54, 2010. ISSN 1070-9932.

[3] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[4] Elena Gribovskaya, Seyed Mohammad Khansari-Zadeh, and Aude Billard. Learning non-linear multivariate dynamics of motion in robotic manipulators. *The International Journal of Robotics Research*, 30(1):80–117, 2011.

[5] G.S. Guthart and Jr. Salisbury, J.K. The IntuitiveTM telesurgery system: overview and application. In *Robotics and Automation (ICRA), 2000 IEEE International Conference on*, volume 1, pages 618–621 vol.1, 2000.

[6] S.-M. Khansari-Zadeh and A. Billard. A dynamical system approach to realtime obstacle avoidance. *Autonomous Robots*, 32:433–454, 2012. ISSN 0929-5593.

[7] S.M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *Robotics, IEEE Transactions on*, 27(5):943–957, 2011. ISSN 1552-3098.

[8] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[9] H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll, and J. Schmidhuber. A system for robotic heart surgery that learns to tie knots using recurrent neural networks. In *Intelligent Robots and Systems (IROS), 2006 IEEE/RSJ International Conference on*, pages 543–548, 2006.

[10] H. Mayer, I. Nagy, A. Knoll, E.U. Braun, R. Lange, and R. Bauernschmitt. Adaptive control for human-robot skilltransfer: trajectory planning based on fluid dynamics. In *Robotics and Automation (ICRA), 2007 IEEE International Conference on*, pages 1800–1807, 2007.

[11] H. Mayer, I. Nagy, D. Burschka, A. Knoll, E.U. Braun, R. Lange, and R. Bauernschmitt. Automation of manual tasks for minimally invasive surgery. In *Autonomic and Autonomous Systems (ICAS), Fourth International Conference on*, pages 260–265, 2008.

[12] Peter Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation (ICRA), 2009 IEEE International Conference on*, pages 763–768, 2009.

[13] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[14] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm for spoken word recognition. In *Acoustics, Speech and Signal Processing, IEEE Transactions on*, pages 159–165. 1978. ISBN 1-55860-124-4.

[15] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *Proc. of the International Symposium on Robotics Research (ISRR)*. Springer, 2004.

[16] John Schulman, Ankush Gupta, Sibi Venkatesan, Mallory Tayson-Frederick, and Pieter Abbeel. A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4111–4117, 2013.

[17] John Schulman, Jonathan Ho, Cameron Lee, and Pieter Abbeel. Learning from demonstrations through the use of non-rigid registration. In *In Proc. of the 16th International Symposium on Robotics Research (ISRR).*, 2013.

[18] Jean Jacques E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice Hall, 1991.

[19] J. van den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, Xiao-Yu Fu, K. Goldberg, and P. Abbeel. Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2074–2081, 2010.