

A Constrained Optimization Approach to Virtual Fixtures

Ming Li, Ankur Kapoor and Russell H. Taylor

Department of Computer Science

NSF Engineering Research Center for Computer-Integrated Surgical Systems and Technology

The Johns Hopkins University, Baltimore, MD 21218, USA

{liming,kapoor,rht}@cs.jhu.edu

Abstract—We describe a new method to generate virtual fixtures for surgical robot control which provide sophisticated ways to assist the surgeon. Different spatial motion constraints for human machine collaborative systems can be implemented by using this method if we know the required geometric constraints and the instantaneous kinematics of the robot. It is independent of manipulator types: teleoperative or cooperative controlled; admittance or impedance type. Our method uses weighted, linearized, multi-objective optimization framework to formalize a library of virtual fixtures for task primitives. We set the cost function based on the user’s inputs, and set linearized subject function based on a combination of five basic geometric constraints. In this paper, we also illustrate the implementation for two sample tasks, which are useful for surgical applications, and provide the experimental results for these tasks.

I. INTRODUCTION

Surgical robotic assistant systems work interactively with surgeons to extend human’s capabilities in carrying out a variety of surgical tasks. They usually are human-machine collaborative systems (HMCS) that are operated directly by the surgeon and augment the surgeon’s ability to manipulate surgical instruments in surgery. Virtual fixtures (VF), which are discussed previously in both telemanipulation and cooperative manipulation contexts, provide sophisticated ways to employ the robot’s capabilities to assist the surgeon.

The goal of VF is to provide anisotropic motion behavior to the surgeon’s motion command besides filtering out tremor to enhance precision, stability and safety. Rosenberg [1] used virtual fixtures, which was implemented using impedance planes and auditory feedback, for a peg-in-hole task in a tele-operated environment. Cobots [2] are robotic systems that create virtual fixtures for collaborative manipulation using specialized hardware.

Davies, *et al.* [3] set active constraints to constrain the robot to cut the femur within a permitted region for prosthetic knee surgery. Park, *et al.* [4] developed sensor-mediated virtual fixtures that constrain the robot’s motion or create haptic feedback directing the surgeon to move the surgical instruments in a desired direction. They applied a virtual wall based on the location of the internal mammary artery obtained from a preoperative CT scan to guide a surgeon’s instrument during tele-operated coronary bypass.

Based on JHU Steady-Hand Robot (SHR) system [5], Bettini, *et al.* [6], [7] has focused on “guidance virtual fixtures” to assist the surgeon to move the surgical instruments

in a desired direction. These work were focused on 2D geometric guidance motion of the tool tip or camera and assumed that the tool or camera itself did not have any other environmental constraints. Marayong, *et al.* [8] described and demonstrated motion constraints with varying compliance that were described for the general spatial case. These work generated virtual fixtures mainly based on admittance control law. Li, *et al.* [9], [10] extended Funda’s work [11] to generate virtual fixtures to assist the surgeon to manipulate surgical tools in a complicated working volume, in which 3D anatomical constraints are automatically generated from 3D medical images. Kapoor, *et al.* [13] used the framework of Funda for constrained motion control of a hybrid 8-DoF robot.

In this paper, we present a new method to implement virtual fixture for HMCS. Our approach is based on the optimized constrained control. Theoretically, different virtual fixtures for HMCS can be implemented by using this method if we know the instantaneous kinematics of the manipulator and the geometric constraints. We first outline our constrained control algorithm and linearized constraints setups of the optimization problem for basic geometric constraints. We then describe the implementation and experiments for two sample tasks. Finally, we present conclusions and future works.

II. OPTIMIZATION CONTROL FOR VIRTUAL FIXTURE

A. Optimization algorithm for constrained control

For a given geometric constraints, we can generate virtual fixtures in the form of a quadratic optimization problem with linear constraints. Our assumption is that the geometric constraints are known in the robot coordinate frame. The basic control loop has been presented in [9], [10]. For abbreviation, we just write the key equations.

$$\begin{aligned} \arg \min_{\Delta \vec{q}/\Delta t} & \|W(\Delta \vec{x}/\Delta t - \Delta \vec{x}_d/\Delta t)\|, \\ \text{s.t.} & \quad H\Delta \vec{x}/\Delta t \geq \vec{h}, \\ & \quad \Delta \vec{x}/\Delta t = J\Delta \vec{q}/\Delta t \end{aligned} \quad (1)$$

where $\Delta \vec{q}$ is the desired incremental motions of the joint variables, $\Delta \vec{x}_d$, $\Delta \vec{x}$ are the desired and the computed incremental motions of the task variables in Cartesian space, respectively. J is the Jacobean matrix relating task space to joint space. W is a diagonal matrix for weights. We must ensure proper scaling of weights corresponding to different components. Otherwise, the result of the optimization $\Delta \vec{q}$ will be skewed, causing incorrect control behavior. Δt is the small time interval of control loop.

We should mention that there is an assumption here that for each iteration loop, the incremental motions are sufficiently small and $\Delta\vec{x}/\Delta t = J\Delta\vec{q}/\Delta t$ represents a good approximation to the relationship between $\Delta\vec{x}/\Delta t$ and $\Delta\vec{q}/\Delta t$. The output of our algorithm $\Delta\vec{q}/\Delta t$ is used as set points for low-level position/velocity control loop which guarantees stability.

B. Geometric constraints

In our approach the required virtual fixtures are analyzed and broken into a combination of one or more of the five basic geometric constraints. In this section, we discuss these five basic geometric constraints and present linearized subject function corresponding to each of these constraints. I.e., we provide the method to set H and \vec{h} in inequality subject function of (1) for each basic constraint. We model the robot task frame as a purely kinematic Cartesian device with the tool position $\vec{x}_p \in \mathbf{R}^3$ and the tool orientation given by unit vector $\hat{l}_t \in \mathbf{R}^3$. Given a reference target, we define the signed distance error $\vec{\delta} = [\vec{\delta}_p^t, \vec{\delta}_r^t]^t \in \mathbf{R}^6$ from the reference target to the tool. The incremental motion is defined as $\Delta\vec{x} = [\Delta\vec{x}_p^t, \Delta\vec{x}_r^t]^t \in \mathbf{R}^6$. We denote translational components by subscript p , and rotational components, which are expressed in Rodriguez vector, by subscript r . We assume that for small angle, $\Delta\vec{x}_r$ approximates Euler Angles.

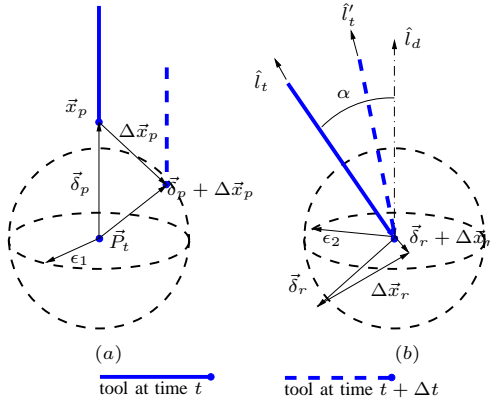


Fig. 1. Geometric relation for (a) Move towards a point and (b) Align tool to a direction

1) *Stay at a point* : The first basic constraint we describe, is to maintain the tool at a given target point $\vec{P}_t \in \mathbf{R}^3$. The signed errors are then set as $\vec{\delta}_p = \vec{x}_p - \vec{P}_t$, $\vec{\delta}_r = \vec{0}$. We require that after the incremental motion, the tool position, $\vec{x}_p + \Delta\vec{x}_p$ to be as close as possible to the target point \vec{P}_t . As shown in Fig. 1(a), we can express this as :

$$\|\vec{\delta}_p + \Delta\vec{x}_p\| \leq \epsilon_1 \quad (2)$$

where ϵ_1 is a small positive number that defines the size of the range that can be considered as the target. It implies that the projections of $\vec{\delta}_p + \Delta\vec{x}_p$ on the pencil through \vec{P}_t be less than ϵ_1 . We approximate the sphere of radius ϵ_1 by polytope having $n \times m$ vertices, and rewrite (2) by a group of linear

inequalities

$$\begin{bmatrix} c_{\alpha i} c_{\beta j}, c_{\alpha i} s_{\beta j}, s_{\alpha i}, 0, 0, 0 \end{bmatrix}^t (\vec{\delta} + \Delta\vec{x}) \leq \epsilon_1; \quad i=1, \dots, n; j=1, \dots, m. \quad (3)$$

$$c_{\alpha i} = \cos \frac{2\pi i}{n}; c_{\beta j} = \cos \frac{2\pi j}{m}; s_{\alpha i} = \sin \frac{2\pi i}{n}; s_{\beta j} = \sin \frac{2\pi j}{m}$$

Then we set H and \vec{h} as

$$H = \begin{bmatrix} -c_{\alpha 1} c_{\beta 1}, & -c_{\alpha 1} s_{\beta 1}, & -s_{\alpha 1}, & 0, & 0, & 0 \\ & \dots & & & & \\ -c_{\alpha 1} c_{\beta m}, & -c_{\alpha 1} s_{\beta m}, & -s_{\alpha 1}, & 0, & 0, & 0 \\ & \dots & & & & \\ -c_{\alpha n} c_{\beta 1}, & -c_{\alpha n} s_{\beta 1}, & -s_{\alpha n}, & 0, & 0, & 0 \\ & \dots & & & & \\ -c_{\alpha n} c_{\beta m}, & -c_{\alpha n} s_{\beta m}, & -s_{\alpha n}, & 0, & 0, & 0 \end{bmatrix}, \quad \vec{h} = \begin{bmatrix} -\epsilon_1 \\ \vdots \\ -\epsilon_1 \end{bmatrix} + H\vec{\delta} \quad (4)$$

As the value of $n \times m$ increases, the volume of polyhedron reduces and the polyhedron approaches the inscribed sphere with radius ϵ_1 . Therefore, the linearized conditions of (4) are a better approximation to (2) for larger values of $n \times m$. However, more constraints require more time to solve the optimization problem. From equation (3), the minimum value for $n \times m$ to obtain a symmetrical polyhedron is 4×4 , though 3×3 gives a bounded polyhedron with least value of $n \times m$.

2) *Maintain a direction*: This basic constraint is to maintain the tool orientation \hat{l}_t to along a given direction \hat{l}_d . The signed errors are then set as $\vec{\delta}_p = \vec{0}$ and $\vec{\delta}_r = \hat{l}_d \times \hat{l}_t$. We require that after the incremental motion, the angle between the tool orientation \hat{l}'_t and \hat{l}_d are close to zero. As shown in Fig. 1(b), we express this as

$$\|\vec{\delta}_r + \Delta\vec{x}_r\| \leq \epsilon_2 \quad (5)$$

where ϵ_2 is a small positive number that defines the size of the range that can be considered as the desired direction. Similar to the discussion in II-B.1, we set H and \vec{h} as

$$H = \begin{bmatrix} 0, & 0, & 0, & -c_{\alpha 1} c_{\beta 1}, & -c_{\alpha 1} s_{\beta 1}, & -s_{\alpha 1} \\ & \dots & & & & \\ 0, & 0, & 0, & -c_{\alpha 1} c_{\beta m}, & -c_{\alpha 1} s_{\beta m}, & -s_{\alpha 1} \\ & \dots & & & & \\ 0, & 0, & 0, & -c_{\alpha n} c_{\beta 1}, & -c_{\alpha n} s_{\beta 1}, & -s_{\alpha n} \\ & \dots & & & & \\ 0, & 0, & 0, & -c_{\alpha n} c_{\beta m}, & -c_{\alpha n} s_{\beta m}, & -s_{\alpha n} \end{bmatrix}, \quad \vec{h} = \begin{bmatrix} -\epsilon_2 \\ \vdots \\ -\epsilon_2 \end{bmatrix} + H\vec{\delta} \quad (6)$$

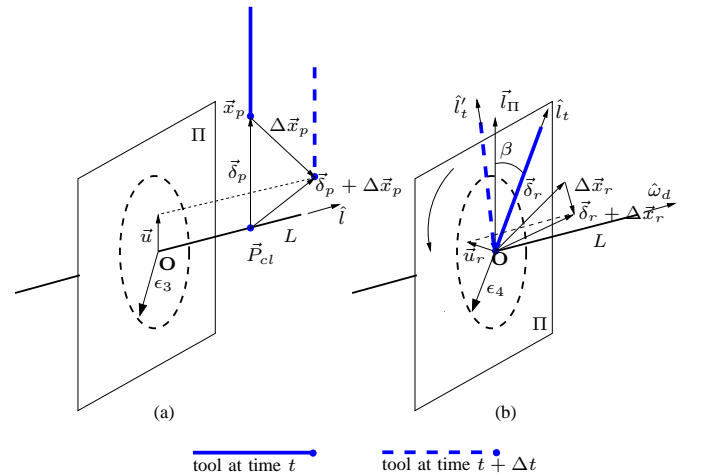


Fig. 2. Geometric relation for (a) Move along a line and (b) Rotate around a line

3) *Move along a line* : The next basic constraint is to guide the tool to move along a reference line in 3D space, given by $L : L(s) = \vec{L}_0 + \hat{l} \cdot s$, where \vec{L}_0 is a point on the line and $\hat{l} = [d_x \ d_y \ d_z]^t$ is the unit vector that indicates the direction of the line. We require that after each incremental motion, the translational component of the tool $\vec{x}_p + \Delta\vec{x}_p$ to be along (or close to) the reference line. If the actual position is off the path because of some external disturbance, the control algorithm should drive the tool back to the line. The geometric constraints should envelop the reference line and absorb the disturbance.

From the given line $L(s)$, we can compute \vec{P}_{cl} which is the closest point on $L(s)$ to \vec{x}_p . The signed errors are then set as $\vec{\delta}_p = \vec{x}_p - \vec{P}_{cl}$ and $\vec{\delta}_r = \vec{0}$. We define a vector \vec{u}_p as the projection of vector $\vec{\delta}_p + \Delta\vec{x}_p$ on the plane Π which is perpendicular to line L . As shown in Fig. 2(a), our requirement is equivalent to $\|\vec{u}_p\|$ being close to zero, which can be written as

$$\|\vec{u}_p\| \leq \epsilon_3 \quad (7)$$

where ϵ_3 is a small positive number that defines the distance error tolerance to the reference line.

To determine \vec{u}_p from $\vec{\delta}_p + \Delta\vec{x}_p$ we need to compute a rotation matrix R_3 , which would transform plane Π to the world (or robot) coordinate frame. To compute R_3 , we first define an arbitrary vector \hat{l}' which is not aligned with \hat{l} , then we generate two unit vectors \hat{v}_1 and \hat{v}_2 which span the plane Π . R_3 is formed by these vectors as shown in (8)

$$R_3 = [\hat{v}_1 \ \hat{v}_2 \ \hat{l}]; \quad \hat{v}_1 = \frac{\hat{l} \times \hat{l}'}{\|\hat{l} \times \hat{l}'\|}; \quad \hat{v}_2 = \frac{\hat{v}_1 \times \hat{l}}{\|\hat{v}_1 \times \hat{l}\|} \quad (8)$$

Any unit vector in the plane Π with \mathbf{O} as origin can be written in world coordinate frame as $R_3 [\cos \alpha_3 \ \sin \alpha_3 \ 0]^t$.

Form (7) implies that projection of \vec{u}_p on the pencil at \mathbf{O} in the plane Π be less than ϵ_3 . We approximate the circle of radius ϵ_3 by considering a polygon with n vertices centered at origin, and rewrite (7) as

$$\left[R_3 \begin{bmatrix} c_{\alpha i} & s_{\alpha i} & 0 \end{bmatrix}^t, 0, 0, 0 \right] \cdot (\vec{\delta}_p + \Delta\vec{x}_p) \leq \epsilon_3, \quad i=1, \dots, n. \quad (9)$$

$c_{\alpha i} = \cos \frac{2\pi i}{n}; s_{\alpha i} = \sin \frac{2\pi i}{n}$

We can set H and \vec{h} as,

$$H = \begin{bmatrix} -R_3 \begin{bmatrix} c_{\alpha 1} & s_{\alpha 1} & 0 \end{bmatrix}^t, 0, 0, 0 \\ \dots \\ -R_3 \begin{bmatrix} c_{\alpha n} & s_{\alpha n} & 0 \end{bmatrix}^t, 0, 0, 0 \end{bmatrix}, \quad \vec{h} = \begin{bmatrix} -\epsilon_3 \\ \dots \\ -\epsilon_3 \end{bmatrix} - H\vec{\delta}. \quad (10)$$

Again as the value of n increases, the area of polygon reduces and the polygon approaches the inscribed circle with radius ϵ_3 . From equation (9), the minimum value for n to obtain a bounded polygon is 3.

4) *Rotate around a line*: Given a line L , with direction \hat{w}_d , the constraint is to rotate the tool around this line while keeping the orientation of the tool perpendicular to L . Even if external disturbance changes the orientation away from the plane Π , our virtual fixture is required to drive it back.

As shown in Fig. 2(b), \hat{l}_Π is the unit vector of the projection of the orientation of the tool \hat{l}_t in plane Π . The signed errors

are set as $\vec{\delta}_p = \vec{0}$ and $\vec{\delta}_r = \hat{l}_\Pi \times \hat{l}_t$. Then we define a vector \vec{u}_r as the projection of vector $\vec{\delta}_r + \Delta\vec{x}_r$ on the plane Π . Our requirement that after the incremental rotation the tool is on plane Π is equivalent to $\|\vec{u}_r\|$ being close to zero.

$$\|\vec{u}_r\| \leq \epsilon_4 \quad (11)$$

where ϵ_4 is a small positive number that defines the error tolerance. The tolerance specifies how much the direction can shift from \hat{w}_d .

We compute R_4 similar to R_3 in (8), but by using \hat{w}_d instead of \hat{l} . Then we write (11) as

$$\left[0, 0, 0, R_4 \begin{bmatrix} c_{\alpha i} & s_{\alpha i} & 0 \end{bmatrix}^t \right] \cdot (\vec{\delta}_p + \Delta\vec{x}_p) \leq \epsilon_4, \quad i=1, \dots, n. \quad (12)$$

$c_{\alpha i} = \cos \frac{2\pi i}{n}; s_{\alpha i} = \sin \frac{2\pi i}{n}$

Then H and \vec{h} are set as:

$$H = \begin{bmatrix} 0, 0, 0, -R_4 \begin{bmatrix} c_{\alpha 1} & s_{\alpha 1} & 0 \end{bmatrix}^t \\ \dots \\ 0, 0, 0, -R_4 \begin{bmatrix} c_{\alpha n} & s_{\alpha n} & 0 \end{bmatrix}^t \end{bmatrix}, \quad \vec{h} = \begin{bmatrix} -\epsilon_4 \\ \dots \\ -\epsilon_4 \end{bmatrix} - H\vec{\delta}. \quad (13)$$

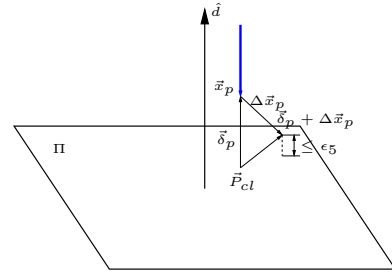


Fig. 3. Geometric relation for plane related case

5) *Plane related case*: In the plane related case, the control algorithm is required either to prevent the tool from penetrating the given plane Π or confine the tool to move on the given plane Π . From the given plane $\Pi(s)$, we can easily compute \vec{P}_{cl} on $\Pi(s)$ which is the closest to \vec{x}_p . The signed errors are $\vec{\delta}_p = \vec{x}_p - \vec{P}_{cl}$ and $\vec{\delta}_r = \vec{0}$. As shown in Fig. 3, to prevent plane penetration, the geometric constraints can be expressed by

$$\hat{d}^t \cdot (\vec{\delta}_p + \Delta\vec{x}_p) \geq 0 \quad (14)$$

where \hat{d} is the unit normal direction of $\Pi(s)$ and points to the free half space. Then H and \vec{h} are set as

$$H = [\hat{d}^t, 0, 0, 0], \quad \vec{h} = -H\vec{\delta}. \quad (15)$$

If we further confine the tool on the plane, we can add constraints

$$\hat{d}^t \cdot (\vec{\delta}_p + \Delta\vec{x}_p) \leq \epsilon_5 \quad (16)$$

where ϵ_5 is a small positive number, which defines the range of error tolerance. Then H and \vec{h} are set as

$$H = \begin{bmatrix} \hat{d}^t & 0 & 0 & 0 \\ -\hat{d}^t & 0 & 0 & 0 \end{bmatrix}, \quad \vec{h} = \begin{bmatrix} 0 \\ \dots \\ -\epsilon_5 \end{bmatrix} - H\vec{\delta}. \quad (17)$$

The values of ϵ_i , $i = 1, \dots, 5$ define the range of the error tolerance. They specify how much the robot can drift away from the reference constraints. For smaller values the user can move the manipulator along the path as specified by constraints. For larger values the user has more free motion.

C. Position rate control in HMCS

To make the HMCS intuitive, the incremental motion should be proportional to the user's input. We can modify the objective function in (1) to (18) to implement such a behavior.

$$\arg \min_{\Delta \vec{q}} \|W(\Delta \vec{x} - k\vec{\tau})\| \quad (18)$$

where $\vec{\tau}$ - the user's input is either force or torque in hands-on cooperative system or joystick readout in remote control. k is a scalar for tuning the ratio between the incremental motion and the desired input. To ensure safety we can also define an upper bound for the incremental motion magnitude.

III. EXPERIMENTS AND RESULTS

We use JHU Steady-Hand Robot [5] (SHR) to implement and demonstrate our work. The robot has 7-DOF with a remote-center-of-motion (RCM) kinematic structure. The robot has $10\mu\text{m}$ position resolution, and (relatively) high stiffness. A 6-DOF force sensor (ATI Nano43 F/T transducer) is integrated at the end-effector. The operator holds a tool mounted at the end-effector. The robot responds to the applied force, allowing the operator to have direct control over the robot motion. We illustrate our virtual fixture implementation using two sample tasks and report the results.

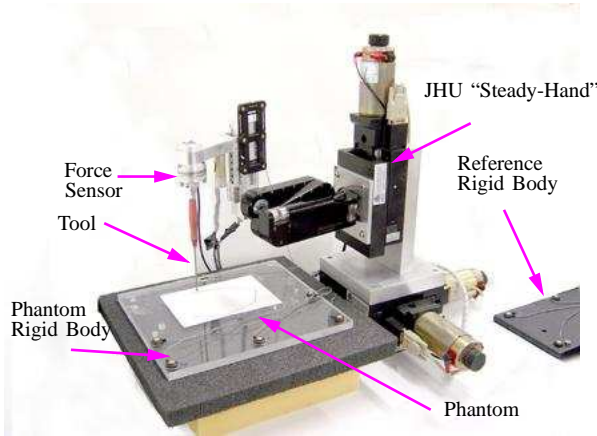


Fig. 4. Experimental setup

A. Follow a curve with a fixed tool orientation with respect to the curve.

The first sample task is to guide the tool tip following a 2D b-spline curve in a plane while keeping the tool perpendicular to the plane. The experimental setup is shown in Fig. 4. We drew a set of line segments on a flat plastic plate, assumed to be a plane, and attached OPTOTRAK[®] (Northern Digital Inc., Waterloo, Ontario, Canada) LEDs to the plate. We defined a *target* coordinate frame for the plane with origin, x and y axis on the plane and z axis pointing along the normal direction of the plane. We used a digitizer to gather sample points on the line segments in the *target* coordinate frame. Then we generated a 5th degree b-spline curve in the *target* coordinate frame by interpolating these sample points. After

robot calibration and registration with a standard method, we transformed the plane and b-spline curve into the robot coordinate frame. We used the OPTOTRAK[®] to track the motion of the plane and the b-spline curve.

We implemented a virtual fixture for this task by generating constraints on two task frames: the tool tip frame and the tool shaft frame. The “*tool tip frame*” refers to a coordinate frame whose origin is at the tip of the tool and whose orientation is parallel to the tool holder of the robot. The “*tool shaft frame*” refers to a coordinate frame whose origin corresponds to the point on the tool that is 100mm away from the tool tip and whose orientation is again parallel to the tool holder.

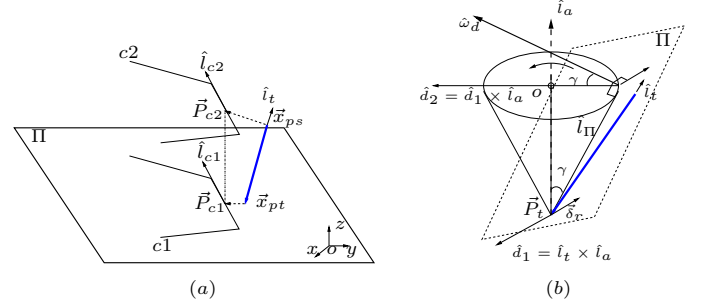


Fig. 5. Geometric relation for two sample tasks (a) Follow a curve with a fixed tool orientation with respect to the curve (b) Rotate around an axis with a fixed angle.

Tool tip frame: For each computational loop, the tool tip position \vec{x}_{pt} and the tool orientation \hat{l}_t are obtained from the robot encoders and its kinematics. We compute the closest point \vec{P}_{c1} to \vec{x}_{pt} on the b-spline $c1$. The tangent direction \hat{l}_{c1} of the b-spline on \vec{P}_{c1} serves as the reference direction. Using the method discussed in II-B.3, we set H_t and \vec{h}_t using $n = 8$ in (10). The constraints for the tool tip frame are

$$H_t J_t(q) \Delta \vec{q} \geq \vec{h}_t \quad (19)$$

where $J_t(q)$ is the Jacobian matrix that maps the tool tip frame to the joint space.

Tool shaft frame: We create a second b-spline curve $c2$ by translating the given b-spline $c1$ by $[0 \ 0 \ 100\text{mm}]^t$ in the *target* coordinate frame. We maintain the tool shaft perpendicular to the plane by constraining the origin of the tool shaft frame to move along $c2$.

As shown in Fig. 5(a), we translate \vec{P}_{c1} by $[0 \ 0 \ 100\text{mm}]^t$ in *target* coordinate frame to obtain \vec{P}_{c2} . The reference direction is set as $\hat{l}_{c2} = \hat{l}_{c1}$. Using the method discussed in II-B.3, we set H_s and \vec{h}_s using $n = 8$ in (10). The constraints for the tool shaft frame are

$$H_s J_s(\vec{q}) \Delta \vec{q} \geq \vec{h}_s \quad (20)$$

where $J_s(\vec{q})$ is the Jacobian matrix that maps the tool shaft frame to the joint space.

We combine two constraints together to generate the virtual fixture for the task. We require our tip motion proportional to the user's force input \vec{f} , then we set our optimization problem

as

$$\begin{aligned} & \arg \min_{\Delta \vec{q}} \left\| W_t J_t(\vec{q}) \Delta \vec{q} - k \vec{f} \right\|, \\ \text{s.t.} \quad & \begin{bmatrix} H_t & 0 \\ 0 & H_s \end{bmatrix} \begin{bmatrix} J_t(\vec{q}) \\ J_s(\vec{q}) \end{bmatrix} \Delta \vec{q} \geq \begin{bmatrix} \vec{h}_t \\ \vec{h}_s \end{bmatrix} \end{aligned} \quad (21)$$

where $W_t = I$. We solve (21) by LSI method in [12] for the set of joint displacements $\Delta \vec{q}$, then move the robot. We set ϵ_3 for both constraints as 0.001 to generate stiff motion constraints. k is set as 0.05.

To validate our experiment, we use the OPTOTRAK[®] to record the tool tip position and the tool orientation. The tip position error is defined as the distance from the tool tip position to the reference b-spline curve. The average tip position error of five trials is $0.32 \pm 0.19 \text{ mm}$. The trajectory of the tool tip with respect to the b-spline curve and the error profile of a trial are shown in Fig 6. The big errors occur at the sharp turnings. The time for each loop is around 150 ms , in which more than 140 ms is for communication between the robot and the OPTOTRAK[®] reading via a local network. The communication delay contributes to the big errors on the sharp turnings where the tangent direction changes dramatically. To evaluate the effect of the communication delay, we compared the tip error using different frequencies for the control loop. We removed the communication between the robot and OptoTrak from the loop, only used robot encoders and kinematics to record the tool tip motion. As shown in Fig. 7, the error at the sharp turning decreased in the case that the time interval of each loop is shorter.

Further, we move and tilt our reference b-spline curve by moving and tilting the phantom plate. Our control algorithm drove the tool tip back to the b-spline. Fig. 8 shows the error absorption profile. Again the communication delay contributes to the time lag.

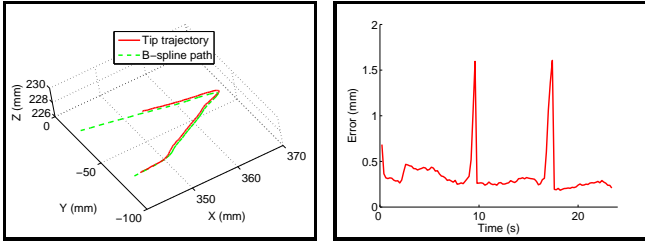


Fig. 6. In following a curve task, The results are recorded in the OPTOTRAK[®] tracking system. (left) The trajectory of the tool tip with respect to the reference b-spline curve. (right) The magnitude of the tool tip position error.

B. Rotate tool around an axis with a fixed angle

This sample task demonstrates a virtual remote center of motion configuration. The aim is to guide the tool tip to pivot on a point other than the SHR mechanical RCM while rotating the tool shaft around a given direction with a fixed angle. In the experiment, we set the given direction as $\vec{l}_a = [0 \ -0.2 \ 1]^t$, the fixed angle as $\gamma = 15 \text{ deg}$, the pivot point \vec{P}_t as $[0 \ 0 \ -10]^t$ with respect to the mechanical RCM.

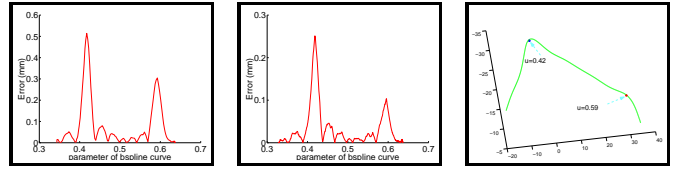


Fig. 7. The magnitude of the tool tip position error measured by the robot encoders and kinematics with different time interval for each control loop in following the curve task. (left) around 150ms each loop, (middle) around 40ms each loop, (right) b-spline curve and the position on which the large errors occur.

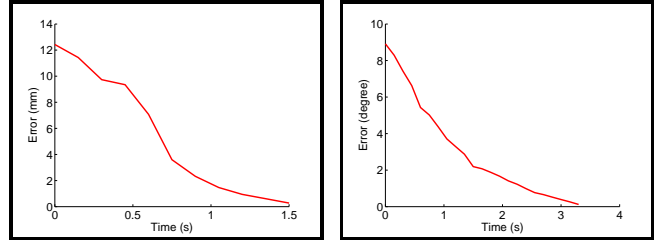


Fig. 8. Error absorption profile. (left) Magnitude of the position error during the error absorption. (right) Magnitude of the orientation error during the error absorption.

Instead of constraining two frames, we only constrain on the tool tip frame in this sample task. For each computational loop, the tool tip position \vec{x}_{pt} and the tool orientation \hat{l}_t are obtained from the robot encoders and its kinematics. For the rotational component, we constrain the tool to rotate along an axis. As shown in Fig. 5(b), the rotational axis $\hat{\omega}_d$ can be calculated as

$$\begin{aligned} \hat{d}_1 &= \frac{\hat{l}_t \times \vec{l}_a}{\|\hat{l}_t \times \vec{l}_a\|}; \quad \hat{d}_2 = \frac{\hat{d}_1 \times \vec{l}_a}{\|\hat{d}_1 \times \vec{l}_a\|}; \\ R &= \begin{bmatrix} \hat{d}_2 \frac{\vec{l}_a}{\|\vec{l}_a\|} & -\hat{d}_1 \end{bmatrix}; \\ \hat{\omega}_d &= R \cdot [\cos \gamma \ \sin \gamma \ 0]^t \end{aligned} \quad (22)$$

The signed error $\vec{\delta}_r$ can be computed as

$$\vec{\delta}_r = \frac{\hat{l}_t \times \hat{l}_a}{\|\hat{l}_t \times \hat{l}_a\|} \cdot \sin \gamma - \hat{l}_t \times \hat{l}_a \quad (23)$$

Following the method described in II-B.4, we set H_r and \vec{h}_r using $n = 8$ in (13).

For the translational component, we constrain the tool tip on \vec{P}_t . Following the method described in II-B.1, we set H_p and \vec{h}_p using $n = 4$ in (4). We require the tool rotational motion proportional to the user's torque input $\vec{\tau}$, then we set the optimization problem as

$$\begin{aligned} & \arg \min_{\Delta \vec{q}} \| W_t (J_t(\vec{q}) \Delta \vec{q} - k \vec{\tau}) \|, \\ \text{s.t.} \quad & \begin{bmatrix} H_p \\ H_r \end{bmatrix} J_t(\vec{q}) \Delta \vec{q} \geq \begin{bmatrix} \vec{h}_p \\ \vec{h}_r \end{bmatrix} \end{aligned} \quad (24)$$

where $W_t = I$. To generate stiff motion constraints, we set the distance error tolerance ϵ_1 for the tip position as 0.001 and the angular error tolerance ϵ_4 for the tool orientation as 0.001. k is set as 0.01.

The time for each loop is 10ms. We use both the robot encoders and the OPTOTRAK[®] to record the tool tip position and the tool orientation. The results are shown in Fig. 9 and Fig. 10. The average pivot point position error for five trials is $0.01 \pm 0.01\text{mm}$ and $0.23 \pm 0.11\text{mm}$ measured by robot encoders and the OPTOTRAK[®] respectively and the average tool orientation angle error for five trials is $0.03 \pm 0.02\text{deg}$ and $0.09 \pm 0.06\text{deg}$ respectively. The RMS accuracy of the OPTOTRAK[®] tracking system is 0.2mm in 3D space.

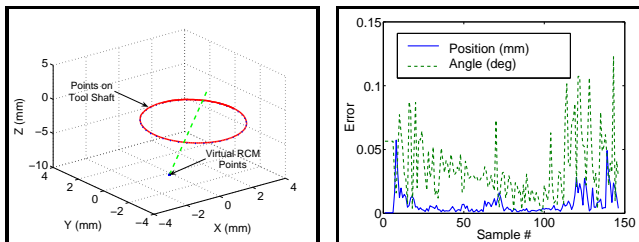


Fig. 9. Experimental results of the second task measured by the robot itself. (left) The trajectory of a point on the tool shaft and the virtual RCM points. The circle shows the actual trajectory of a point on the tool shaft, the dots closed to the circle represent the desired trajectory of the point. (right) The error profile for both the pivot point position error and the tool orientation angle error.

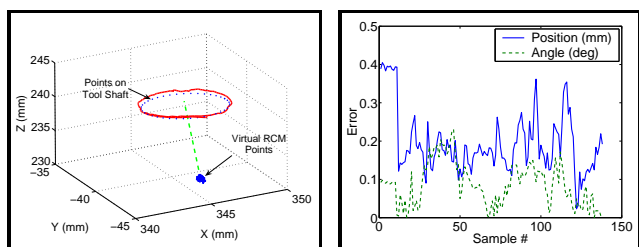


Fig. 10. Experimental results of the second task measured in the OPTOTRAK[®] tracking system. (left) The trajectory of a point on the tool shaft and the virtual RCM points. The circle shows the actual trajectory of a point on the tool shaft, the dots closed to the circle represent the desired trajectory of the point. (right) The error profile for both the pivot point position error and the tool orientation angle error.

IV. CONCLUSIONS AND FUTURE WORK

We describe a new method to generate motion constraints for Human Machine Collaborative Systems.

The presented method applies constrained optimization to generate different types of virtual fixtures for spatial motion constraints. Our method is independent of the manipulator type: teleoperative controlled or hands-on cooperative controlled, admittance type or impedance type. Virtual fixtures for different tasks can be generated by combining one or more basic constraints for one or more task frames.

We show our implementation and the experiments on two sample tasks. The experimental results show that our control

algorithm can generate desired motion constraints and provide assistant to the users in HMCS. Both of the tasks can be easily extended for surgical applications. We can extend the first task to assist the orthopedic surgeon cutting femur for prosthetic implant. In the second sample task, we generate a virtual remote center of motion (RCM), which is useful for percutaneous procedures.

We are now using our method to explore virtual fixtures for more complicated surgical procedures, such as suturing. Currently we maintain safety by limiting the maximum velocity, we would investigate more sophisticated strategies in the future.

ACKNOWLEDGMENT

Partial funding of this research was provided by the National Science Foundation under grants EEC9731748 (CISST ERC), IIS0205318, and JHU internal funds.

REFERENCES

- [1] L. Rosenberg, "Virtual Fixtures: Perceptual Tools for Telerobotic Manipulation," Proceedings of the IEEE Virtual Reality International Symposium, pp. 76-82, 1993.
- [2] M. A. Peshkin, J. E. Colgate, W. Wannasupphrasit, C. A. Moore, R. B. Gillespie, and P. Akella, "Cobot Architecture," IEEE Transactions on Robotics and Automation, pp. 377-390. 17(4),2001
- [3] B. L. Davies, S.J. Harris, W.J. Lin, R.D. Hibberd, R. Middleton, J.C. Cobb, "Active Compliance in robotic surgery - the use of force control as a dynamic constraint," Proceedings of the I MECH E Part H Journal of Engineering in Medicine, vol. 211, pp. 285-292, 1997.
- [4] S. Park, R. Howe, and D. Torchiana, "Virtual fixtures for Robotic Cardiac Surgery," 4th International Conference on Medical Image Computing and Computer-Assisted Intervention, 2001.
- [5] R. H. Taylor, J. Patrick, L.L. Whitcomb, A. Barnes, R. Kumar, D. Stoianovici, P. Gupta, Z. Wang, E. deJuan, L. Kavoussi, "A Steady-Hand Robotic System for Microsurgical Augmentation," International Journal of Robotics Research, vol. 18(12), pp. 1201-1210, 1999.
- [6] A. Bettini, S. Lang, A. Okamura, G. Hager, "Vision Assisted Control for Manipulation Using Virtual Fixtures," IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1171-1176, 2001.
- [7] A. Bettini, S. Lang, A.M. Okamura, and G. Hager., "Vision assisted control for manipulation using virtual fixtures: Experiments at macro and micro scales.," Proc. IEEE International Conference on Robotics and Automation, pp. 3354-3361, 2002.
- [8] P. Marayong, M. Li, A.M. Allison, G.D. Hager, "Spatial Motion Constraints: Theory and Demonstrations for Robot Guidance Using Virtual Fixtures," Proceedings of the 2003 IEEE International Conference on Robotics and Automation, pp. 1954-1959, 2003.
- [9] M. Li, and R.H. Taylor, "Spatial Motion Constraints in Medical Robot Using Virtual Fixtures Generated by Anatomy," Proceedings of the 2004 IEEE International Conference on Robotics and Automation, pp. 1270-1275, 2004.
- [10] M. Li, and R.H. Taylor, "Optimum Robot Control for 3D Virtual Fixture in Constrained ENT Surgery," Proceeding of Medical Image Computing and Computer Assisted Intervention (MICCAI), pp. 165-172, 2003.
- [11] J. Funda, R.H. Taylor, B. Eldridge, S. Gomory, K.G. Gruben, "Constrained Cartesian motion control for teleoperated surgical robots," IEEE Transactions on Robotics and Automation, vol. 1996, 12(3), pp. 453-465, 1996.
- [12] C. Lawson, and R. Hanson, "Solving Least Squares Problems," Englewood Cliffs, NJ: Prentice-Hall, 1974.
- [13] A. Kapoor, and N. Simaan, and R. H. Taylor, "Suturing in Confinned Spaces: Constrained Motion Control of a Hybrid 8-DoF Robot," International Conference on Advanced Robotics, Seattle, WA, 2005