22

Telerobotic Control by Virtual Fixtures for Surgical Applications

Ming Li, Ankur Kapoor, and Russell H. Taylor

Johns Hopkins University
Department of Computer Science
3400 N. Charles St.
Baltimore, MD 21218, USA
{liming,kapoor,rht}@cs.jhu.edu

Summary. We present a new method to generate spatial motion constraints for surgical robots that provide sophisticated ways to assist the surgeon. Surgical robotic assistant systems are human-machine collaborative systems (HMCS) that work interactively with surgeons by augmenting their ability to manipulate surgical instruments in carrying out a variety of surgical tasks. The goal of "virtual fixtures" (VF) is to provide anisotropic motion behavior to the surgeon's motion command and to filter out tremor to enhance precision and stability. Our method uses a weighted, linearized, multi-objective optimization framework to formalize a library of virtual fixtures for task primitives. We set the objective function based on user input that can be obtained through a force sensor, joystick or a master robot. We set the linearized subject function based on five basic geometric constraints. The strength of this approach is that it is extensible to include additional constraints such as collision avoidance, anatomy-based constraints and joint limits, by using an instantaneous kinematic relationship between the task variables and robot joints. We illustrate our approach using three surgical tasks: percutaneous needle insertion, femur cutting for prosthetic implant and suturing. For the percutaneous procedures we provide a remote center of motion (RCM) point that provides an isocentric motion that is fundamental to these types of procedures. For femur cutting procedures we provide assistance by maintaining proper tool orientation and position. For the suturing task we address the problem of stitching in endoscopic surgery using a circular needle. We show that with help of VF, suturing can be performed at awkward angles without multiple trials, thus avoiding damage to tissue.

22.1 Introduction

Robotic surgical assistance is an emerging technology of human-computer cooperation to accomplish delicate and difficult surgical tasks. Examples of surgical assistant systems can be seen in laparoscopic surgery [1, 2, 3, 4, 5], microsurgery [6, 7, 8], orthopedic surgery [9] and sinus surgery [10]. The Intuitive daVinci robotic surgical system [3] and the Computer Motion Zeus robotic surgical system [4] are two commercialized surgical telemanipulators which are capable of performing remote telerobotic laparoscopic surgery. Although most of the teleoperated systems are admittance-controlled microsurgical robots ([6, 11, 12, 13, 14]), based on

M. Ferre et al. (Eds.): Advances in Telerobotics, STAR 31, pp. 381–401, 2007.
springerlink.com

force–reflecting master–slave configurations, some of the teleoperated robotic augmentation systems employ a passive input device for operator control [7], while others are joystick controlled [10].

Most surgical procedures in which robots are called for assistance are characterised by restricted access to the workspace as well as constrained manipulation of the surgical tool. In such cases, the surgeons' ability can be augmented by techniques such as virtual fixtures (VF). Virtual fixtures, which have been discussed previously in the literature for both telerobotic and cooperative robots [15, 16, 17, 18, 19, 20], are algorithms which provide anisotropic behavior to surgeons motion command besides filtering out tremor to provide safety and precision.

An important case of virtual fixtures are forbidden regions, where the surgical tool is restricted to a certain region in the workspace. Davies *et al.* [15] set active constraints to constrain the robot to cut the femur and tibia within a permitted region for prosthetic knee surgery. Park *et al.* [16] 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 teleoperated coronary bypass. The recent work at JHU by Okamura, Hager *et al.* on virtual fixtures [17, 18, 19] used admittance control laws to implement vision-based guidance virtual fixtures for retinal vein cannulation. These works are based either on a specific robot type or on a specific task.

This chapter presents a new method to implement virtual fixtures for surgical assistant robots. We extend the work of Funda *et al.* [21] by applying the method to generate complicated virtual fixtures based on the human's input for surgical assistant robots. Funda presented an optimal motion control method to control both redundant and deficient robotic systems based on motion constraints. Our approach [22] uses a weighted, linearized multi-objective optimization framework to formalize a library of virtual fixtures for task primitives. Our paradigm covers the implementation of guidance virtual fixtures, forbidden region virtual fixtures and combinations of both for generating spatial motion constraints to control the robotic assistant. It is independent of manipulator characteristic and can be used for admittance or impedance type.

22.2 Constrained Motion Control for Virtual Fixtures

Virtual fixtures are task-dependent computer-generated constraints that limit the robot's movement into restricted regions and/or influence its movement along desired paths. The goal of the virtual fixture algorithm is to generate a sequence of incremental motion commands for the robot such that certain task-specific constraints are satisfied. To keep the system intuitive, the incremental motion should be proportional to the user input. For a surgical assistant robot, it is very important to be able to place absolute bounds on the spatial motion of the different parts of the instrument.

Often, surgical robots are designed to be kinematically redundant for providing dexterous assistance. At the same time, certain tasks such as passing a tool through a cavity place certain requirements and constraints on the robot motion and restrict dexterity. Indeed, some special purpose designs for minimally invasive surgery, such as the IBM LARS [23] and the JHU Steady Hand robot [24] provide such constraints through mechanism design. Other robots such as the Intuitive daVinci [3] and Endorobotics [25] combine a kinematically constrained remote center of motion (RCM) mechanism with a kinematically redundant wrist. Thus, it is important for the robot control algorithm to be able to accommodate unique, special purpose mechanical designs (such as kinematically redundant or deficient mechanisms). In this section we present an overview of constrained optimization approach, followed by certain approximations that allow us to execute the algorithm in real-time.

22.2.1 Constrained Optimization Approach

We begin by defining different task frames that specify a point and/or direction of interest associated with different parts of the instrument. For each of the task frames, we define actual state variables $\mathbf{x}$ and desired state variables $\mathbf{x}^d$. The state, $\mathbf{x} = \mathbf{x}(\mathbf{q} + \Delta\mathbf{q})$ is a function of joint variables $\mathbf{q}$ and joint incremental motion $\Delta\mathbf{q}$. The desired state, $\mathbf{x}^d = \mathbf{x}^d(\boldsymbol{\tau},\ \mathbf{q})$ is a function of human's input $\boldsymbol{\tau}$, joint variables $\mathbf{q}$.

We can formulate a constrained optimization problem to generate the constrained motion for a certain task frame. The most general formulation for this problem is:

$$\begin{aligned} &\Delta\mathbf{q}_{cmd} = \arg\min_{\Delta\mathbf{q}} C(\mathbf{x}(\mathbf{q} + \Delta\mathbf{q}), \mathbf{x}^d) \\ &s.t. \quad A(\mathbf{x}(\mathbf{q} + \Delta\mathbf{q})) \leq \mathbf{b} \end{aligned} \tag{22.1}$$

where $C(\mathbf{x}(\mathbf{q} + \Delta\mathbf{q}), \mathbf{x}^d)$ is the objective function associated with the difference between the actual state variables $\mathbf{x}$ and the desired state variables $\mathbf{x}^d$. The inequality, $A(\mathbf{x}(\mathbf{q} + \Delta\mathbf{q})) \leq \mathbf{b}$ represents the constraint conditions. These constraints are used to force the solution vector $\Delta\mathbf{q}_{cmd}$ to satisfy certain critical requirements, such as restricting the motion of a part of the instrument within a strict motion envelope.

We can combine the constrained motions on different task frames for generating complicated constrained motions. For an example, assume the virtual fixture for task frame $\{i\}$ is

$$\begin{aligned} &\Delta\mathbf{q}_{cmd} = \arg\min_{\Delta\mathbf{q}} C_i(\mathbf{x}_i(\mathbf{q} + \Delta\mathbf{q}), \mathbf{x}_i^d) \\ &s.t. \quad A_i(\mathbf{x}_i(\mathbf{q} + \Delta\mathbf{q})) \leq \mathbf{b}_i \end{aligned} \tag{22.2}$$

Then the complicated virtual fixtures generated by constraining on task frames $\{i,\ (i = 1,\ ...,\ N)\}$ can be formulated as

$$\begin{aligned} &\Delta\mathbf{q}_{cmd} = \arg\min_{\Delta\mathbf{q}} \sum_{i=1}^{N} w_i C_i(\mathbf{x}_i(\mathbf{q} + \Delta\mathbf{q}), \mathbf{x}_i^d) \\ &s.t. \quad A_i(\mathbf{x}_i(\mathbf{q} + \Delta\mathbf{q})) \leq \mathbf{b}_i \\ &\qquad\qquad i = 1, \cdots, N. \end{aligned} \tag{22.3}$$

where w_i specifies the relative importance of minimizing the objective function error for different task frames. The combination of a weighted objective function and an additional set of task constraints allow us to exploit the geometry of a particular task space motion and effectively trade off the various performance criteria. Our formulation could easily integrate any behavior, such as asserting joint limits, resolving redundancy or incorporating haptic information to the control strategy.

In this work we discuss virtual fixtures for five task primitives that form the basis of a virtual fixture library. The nomenclature used is presented in the table below.

$\mathbf{x}_i$	Cartesian state of task frame $\{i\}$. $\mathbf{x}_i \in \Re^6$. Subscript i can be omitted for compactness.
$\mathbf{x}_i^d$	Desired Cartesian state of task frame $\{i\}$.
$\mathbf{x}_{p,i}$	Translational component state of task frame $\{i\}$. $\mathbf{x}_{p,i} \in \Re^3$.
$\mathbf{x}_{r,i}$	Rotational component state of task frame $\{i\}$. $\mathbf{x}_{r,i} \in \Re^3$.
$\Delta\mathbf{x}$	Incremental Cartesian motion. $\Delta\mathbf{x} \in \Re^6$.
$\Delta\mathbf{q}$	Incremental joint motion. $\Delta\mathbf{q} \in \Re^n$, n is number of robot joints.
$J(q), J$	Jacobian relating the instantaneous kinematics to joint motion.
$\boldsymbol{\delta}$	Signed distance error between desired and current task frame. $\boldsymbol{\delta} \in \Re^6$
$\boldsymbol{\delta}_p$	Translational component of error.
$\boldsymbol{\delta}_r$	Rotational component of error.
$\hat{\mathbf{l}}$	Orientation of task frame. $\hat{\mathbf{l}} \in \Re^3$. E.g. $\hat{\mathbf{z}}$ component of tool tip rotation matrix.
$\hat{\mathbf{l}}^d$	Desired or reference orientation of task frame.
$\boldsymbol{\tau}$	User input, from force sensor or joystick or master.
$\hat{\mathbf{d}}$	Predefined direction specified by user. $\hat{\mathbf{d}} \in \Re^3$. E.g normal to plane, direction of path.
ϵ_i	Small positive numbers. $i = 1, 2, \ldots$

Using the formulation shown in (22.3), one or more of these primitives applied to one or more task frames can be combined to create complex fixtures. Complicated surgical tasks can then be composed from a sequence of these customized virtual fixtures. The names and descriptions of these task primitives are listed below.

- **Stay on a point:** Keep the tool position represented by $\mathbf{x}_p$ fixed on the reference position $\mathbf{x}_p^d$.
- **Maintain a direction:** Keep the tool orientation represented by $\hat{\mathbf{l}}$ aligned with the reference direction $\hat{\mathbf{l}}^d$.

- **Move along a line:** Keep the tool position, $\mathbf{x}_p$ on line L which has the direction $\hat{\mathbf{d}}$ and passes through point $\mathbf{L}_0$. At the same time, the tool should move along L proportional to the human's input $\boldsymbol{\tau}$.
- **Rotate around a line:** Keep the tool orientation, $\hat{\mathbf{l}}$ perpendicular to line L which has the direction $\hat{\mathbf{d}}$ and passes through point $\mathbf{L}_0$. At the same time, the tool should rotate around L proportional to the human's input $\boldsymbol{\tau}$.
- **Stay above a plane:** Keep the tool position, $\mathbf{x}_p$ stay above a plane Π which has the normal direction $\hat{\mathbf{d}}$ pointing to the free half space and passes through point $\mathbf{P}_0$. At the same time, the tool should move proportional to the human's input $\boldsymbol{\tau}$.

We define a desired nominal behavior together with constraints specifying how far actual behavior can differ from the nominal for each of these primitives. The terms in the objective function of (22.3) are used to relate the desired motion to user input, while the constraints place an absolute bound on the motion. In Sec. 22.3 we elaborate on constraints for each of the five primitive virtual fixtures.

22.2.2 Linearly Constrained Control

The general form of the optimization problem desribed by (22.3) has many variants, both for the objective function and the constraints. As in Funda's work [21], we specialize (22.3) to produce a quadratic optimization problem with linear constraints. We use linear constraints because of the efficiency and the robustness of the computation. The objective function is a two-norm of motion error in different task frames. Because the velocity of a surgical robot is relative low, we can use robotic instantaneous kinematics to map the different task frames to joint variables. The incremental motion of a certain task space is approximated as $\Delta\mathbf{x} = J(\mathbf{q})\Delta\mathbf{q}$. Then we set an optimization problem over incremental joint motion $\Delta\mathbf{q}$. The cost function for task frame $\{i\}$ has the form $\left\|J_i(\mathbf{q})\Delta\mathbf{q} - \Delta\mathbf{x}_i^d\right\|_2^2$, and the constraint has the form $A_i \cdot J_i(\mathbf{q})\Delta\mathbf{q} \leq \mathbf{b}_i$. This inequality is linear in terms of our variables $\Delta\mathbf{q}$. The matrix A_i and vector $\mathbf{b}_i$ are based on the virtual fixture primitive for task frame $\{i\}$.

Then the complicated virtual fixtures generated by combining task frames $\{i,\ (i = 1,\ ...,\ N)\}$ can be expressed as

$$\begin{bmatrix} A_1 & 0 \\ & \ddots \\ 0 & A_N \end{bmatrix} \begin{bmatrix} J_1(\mathbf{q}) \\ \vdots \\ J_N(\mathbf{q}) \end{bmatrix} \Delta\mathbf{q} \leq \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_N \end{bmatrix} \tag{22.4}$$

where $J_i(\mathbf{q})$ $(1 \leq i \leq N)$ is the Jacobian matrix that maps Cartesian velocities of frame $\{i\}$ to the joint space. If the constraints are placed on the rotational (with subscript r) and translational (with subscript p) components of the same frame, the virtual fixtures can be expressed as

$$\begin{bmatrix} A_p \\ A_r \end{bmatrix} J(\mathbf{q})\Delta\mathbf{q} \leq \begin{bmatrix} \mathbf{b}_p \\ \mathbf{b}_r \end{bmatrix} \tag{22.5}$$

Our control algorithm need not to be modified for singularity, but a safety check is important for actual robot control. Our optimization algorithm forms a least squares problem with linear inequality constraints (algorithm LSI in [26]). At each control loop, we check the solution of LSI before we command robot to move. If the singularity is reached, then there will be no solution for LSI. If the robot is very close to singularity, the solution for the LSI will be extremely large. In both cases, we should stop the robot. Moreover an objective term $\|W_q \Delta q\|_2^2$ can be added to the overall objective function, where W_q is a diagonal matrix specifying the relative weights between different joints. Such an objective function ensures reasonable and smooth joint velocities when the robot is near singular points. The weights are chosen empirically and we typically use weights in the range 1×10^{-3} to 1×10^{-9}.

Additonally, it is straightforward to incorporate a per joint rate limit by using additional inequality constraints of the form

$$\begin{bmatrix} I \\ -I \end{bmatrix} \Delta \mathbf{q} \leq \begin{bmatrix} \mathbf{q}_{max} - \mathbf{q} \\ \mathbf{q} - \mathbf{q}_{min} \end{bmatrix} \tag{22.6}$$

Besides this, the norm of the cost function gives an indication if an appropriate solution that would satisfy all the constraints is possible. A large norm implies no appropriate solution exists for the given set of constraints. Under such conditions the safest behavior for a surgical robot would be to stop motion. Alternatively some constraints can be relaxed if possible, but we suggest that this should be done only through some user intervention.

22.3 Basic Geometric Constraints for the Virtual Fixture Library

Next we present linearized approximations of constraints for five task primitives. That is, we provide a method to set A and $\mathbf{b}$ in the inequality linear constraints (22.4). We model the robot i^{th} task frame as a purely kinematic Cartesian device with the task frame position $\mathbf{x}_p \in \Re^3$ and the task frame orientation given by unit vector $\hat{\mathbf{l}} \in \Re^3$. Given a reference target, we define the signed distance error $\boldsymbol{\delta} = \left[\boldsymbol{\delta}_p{}^T, \boldsymbol{\delta}_r{}^T \right]^T \in \Re^6$ from the reference target frame to the task frame. The incremental motion for each computational loop is $\Delta \mathbf{x} = \left[\Delta \mathbf{x}_p{}^T, \Delta \mathbf{x}_r{}^T \right]^T \in \Re^6$. We denote translational components by subscript p, and rotational components expressed in Rodriguez angles by subscript r. We assume that both the distance error $\boldsymbol{\delta}$ and the incremental motion $\Delta \mathbf{x}$ are very small and that for small angles, $\boldsymbol{\delta}_r$ and $\Delta \mathbf{x}_r$ approximate Euler Angles. $\epsilon_{1,2,3,4,5}$ are small positive values.

22.3.1 Stay on a Point (VF1)

The first basic geometric constraint we describe is to keep the task frame position on a given target point $\mathbf{x}_p^d \in \Re^3$. The signed errors are then set as $\boldsymbol{\delta} \equiv [\boldsymbol{\delta}_p^T, \boldsymbol{\delta}_r^T]^T = [(\mathbf{x}_p - \mathbf{x}_p^d)^T, \mathbf{0}^T]^T$. We require that after the incremental motion, the task frame

position $\mathbf{x}_p + \Delta\mathbf{x}_p$ to be as close to the target point $\mathbf{x}_p^d$ as possible. As shown in Fig. 22.1(a), the constraint can be expressed as

$$\|\boldsymbol{\delta} + \Delta\mathbf{x}\|_2 = \|\boldsymbol{\delta}_p + \Delta\mathbf{x}_p\|_2 \leq \epsilon_1 \tag{22.7}$$

which implies that the various projections of vector $\boldsymbol{\delta}_p + \Delta\mathbf{x}_p$ on the pencil through $\mathbf{x}_p^d$ are less than ϵ_1. We approximate the sphere of radius ϵ_1 by considering a polyhedron with $n \times m$ vertices, and rewrite (22.7) by linear inequalities.

$$\begin{gathered}\left[\cos\alpha_{1i}\cos\beta_{1j}\ \cos\alpha_{1i}\sin\beta_{1j}\ \sin\alpha_{1i}\ 0\ 0\ 0\right]\cdot(\boldsymbol{\delta}+\Delta\mathbf{x}) \leq \epsilon_1, \\ i = 0, 1, \cdots, n-1; \quad j = 0, 1, \cdots, m-1.\end{gathered} \tag{22.8}$$

where $\alpha_{1i} = \frac{i2\pi}{n}$ and $\beta_{1j} = \frac{j2\pi}{m}$. Then we set A and $\mathbf{b}$ as

$$A = \begin{bmatrix} \cos\alpha_{11}\cos\beta_{11} & \cos\alpha_{11}\sin\beta_{11} & \sin\alpha_{11}\ 0\ 0\ 0 \\ & \cdots & \\ \cos\alpha_{1n}\cos\beta_{1m} & \cos\alpha_{1n}\sin\beta_{1m} & \sin\alpha_{1n}\ 0\ 0\ 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \epsilon_1 \\ \cdots \\ \epsilon_1 \end{bmatrix} - A\boldsymbol{\delta}. \tag{22.9}$$

Note that (22.7) and (22.8) are equivalent only if $m = n = \infty$. For finite values of m and n, (22.8) results in a polyhedron. 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 (22.8) are a better approximation to (22.7) for larger values of $n \times m$. However, more constraints require more time to solve the optimization problem. From (22.8), 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 $n \times m$. On a Pentium IV, 2.0GHz, 512MB computer the average time for each computational loop is 4.1 ms to solve the problem with 8 constraints and 7 decision variables, 7.2 ms if the number of constraints is 16 and 14.3 ms if the number of constraints is 32.

22.3.2 Maintain a Direction (VF2)

This basic geometric constraint is to maintain the task frame orientation $\hat{\mathbf{l}}$ along a given direction $\hat{\mathbf{l}}^d$. The signed errors are then set as $\boldsymbol{\delta} \equiv [\boldsymbol{\delta}_p^T, \boldsymbol{\delta}_r^T]^T = [\mathbf{0}^T, (\hat{\mathbf{l}}^d \times \hat{\mathbf{l}})^T]^T$. We require that after the incremental motion, the angle between the new task frame orientation $\hat{\mathbf{l}}'$ and $\hat{\mathbf{l}}^d$ is close to zero. As shown in Fig. 22.1(b), we can approximate this constraint for a small angle assumption as

$$\|\boldsymbol{\delta}_r + \Delta\mathbf{x}_r\|_2 \leq \epsilon_2 \tag{22.10}$$

where ϵ_2 defines the size of the range that can be considered as the desired direction. Applying the approach described in Sec. 22.3.1 to the angular components, we set A and $\mathbf{b}$ as

$$A = \begin{bmatrix} 0\ 0\ 0 & \cos\alpha_{21}\cos\beta_{21} & \cos\alpha_{21}\sin\beta_{21} & \sin\alpha_{21} \\ & \cdots & & \\ 0\ 0\ 0 & \cos\alpha_{2n}\cos\beta_{2m} & \cos\alpha_{2n}\sin\beta_{2m} & \sin\alpha_{2n} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \epsilon_2 \\ \cdots \\ \epsilon_2 \end{bmatrix} - A\boldsymbol{\delta}. \tag{22.11}$$

where $\alpha_{2i} = \frac{i2\pi}{n}$ and $\beta_{2j} = \frac{j2\pi}{m}$.

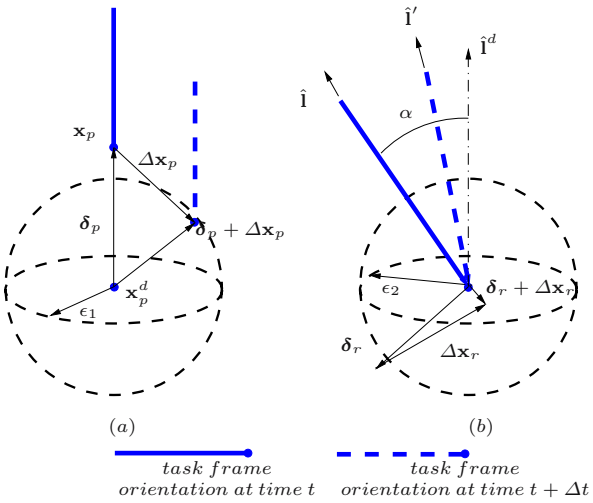

Fig. 22.1. Geometric relation for (a) "stay on a point" and (b) "maintain a direction"

22.3.3 Move Along a Line (VF3)

The next basic geometric constraint is to guide the task frame position to move along a reference line in 3D space, given by $L : L(s) = \mathbf{L}_0 + \hat{\mathbf{d}} \cdot s, s \in (-\infty, \infty)$. We require that after each incremental motion, the translational component of the task frame $\mathbf{x}_p + \Delta\mathbf{x}_p$ to be along (or close to) the reference line (see Fig. 22.2(a)). If the actual position is off the path because of some external disturbance, the control algorithm should drive the task frame back to the line. The geometric constraint should envelop the reference line and absorb the disturbance.

From the given line L, we can compute the closest point, $\mathbf{x}_p^{cl}$ to $\mathbf{x}_p$ on L. The signed errors are then set as $\boldsymbol{\delta} \equiv [\boldsymbol{\delta}_p^T, \boldsymbol{\delta}_r^T]^T = [(\mathbf{x}_p - \mathbf{x}_p^{cl})^T, \mathbf{0}^T]^T$. We define a vector $\mathbf{u}_p$ as the projection of vector $\boldsymbol{\delta}_p + \Delta\mathbf{x}_p$ on the plane Π which is perpendicular to line L. As shown in Fig. 22.2(a), our requirement is equivalent to $\|\mathbf{u}_p\|_2$ be close to zero, which can be written as

$$\|\mathbf{u}_p\|_2 \leq \epsilon_3 \tag{22.12}$$

To determine $\mathbf{u}_p$ from $\boldsymbol{\delta}_p + \Delta\mathbf{x}_p$ we need to compute a rotation matrix R_3, which would transform plane Π to the XY plane of the world (or robot) coordinate frame. Though R_3 is not unique, to compute R_3, we first define an arbitrary vector, which is not aligned with $\hat{\mathbf{l}}$, then we generate two unit vectors that span the plane Π. Any unit vector with arbitary angle α_3 in the plane Π with o as origin can be written in the world coordinate frame as

$$R_3 \left[\cos\alpha_3 \ \sin\alpha_3 \ 0 \right]^T \tag{22.13}$$

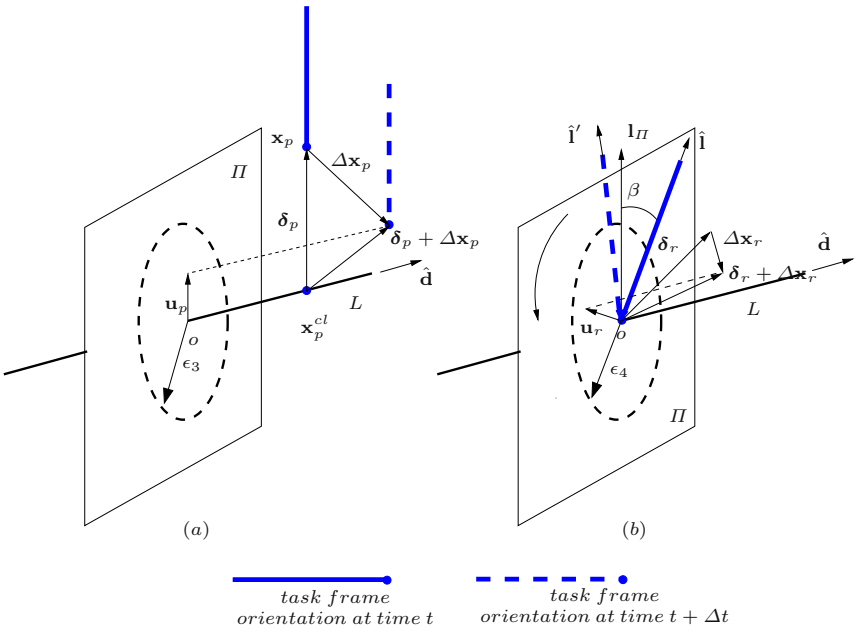

Fig. 22.2. Geometric relation for (a) "move along a line" and (b) "rotate around a line"

Form (22.12) implies that the projections of $\mathbf{u}_p$ on the pencil at 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 the origin, and rewrite (22.12) as

$$\begin{gathered}\left[R_3 \left[\cos\alpha_{3i} \ \sin\alpha_{3i} \ 0\right]^T \ 0 \ 0 \ 0\right] \cdot (\boldsymbol{\delta} + \Delta\mathbf{x}) \leq \epsilon_3, \\ i = 0, 1, \cdots, n-1.\end{gathered} \tag{22.14}$$

where $\alpha_{3i} = \frac{i2\pi}{n}$. We can set A and $\mathbf{b}$ as,

$$A = \begin{bmatrix} R_3 \left[\cos\alpha_{31} \ \sin\alpha_{31} \ 0\right]^T \ 0 \ 0 \ 0 \\ \cdots \\ R_3 \left[\cos\alpha_{3n} \ \sin\alpha_{3n} \ 0\right]^T \ 0 \ 0 \ 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \epsilon_3 \\ \cdots \\ \epsilon_3 \end{bmatrix} - A\boldsymbol{\delta}. \tag{22.15}$$

22.3.4 Rotate Around a Line (VF4)

Given a line L, with direction $\hat{\mathbf{d}}$, the geometric constraint is to rotate the task frame orientation around the lines while keeping the orientation of the task frame on the plane Π that is perpendicular to L. Even if an external disturbance changes the orientation away from the plane Π, our virtual fixture is required to drive it back.

As shown in Fig. 22.2(b), $\hat{\mathbf{l}}_{\Pi}$ is the unit vector of the projection of the orientation of the task frame $\hat{\mathbf{l}}$ on plane Π. The signed errors are set as $\boldsymbol{\delta} \equiv [\boldsymbol{\delta}_p^T, \boldsymbol{\delta}_r^T]^T = [\mathbf{0}^T, (\hat{\mathbf{l}}_{\Pi} \times \hat{\mathbf{l}})^T]^T$. We define a vector $\mathbf{u}_r$ as the projection of vector $\boldsymbol{\delta}_r + \Delta\mathbf{x}_r$ on the plane Π. Our constraint that after the incremental rotation the task frame is on plane Π is equivalent to $\|\mathbf{u}_r\|_2$ being close to zero:

$$\|\mathbf{u}_r\|_2 \leq \epsilon_4 \tag{22.16}$$

We compute R_4 in the same fashion as R_3. Then we write (22.16) as

$$\begin{gathered}\left[\,0\;0\;0\;R_4\left[\,\cos\alpha_{4i}\;\sin\alpha_{4i}\;0\,\right]^T\,\right]\cdot(\boldsymbol{\delta}+\Delta\mathbf{x}) \leq \epsilon_4,\\ i = 0, 1, \cdots, n-1.\end{gathered} \tag{22.17}$$

where $\alpha_{4i} = \frac{i2\pi}{n}$. Then A and $\mathbf{b}$ are set as:

$$A = \begin{bmatrix} 0\;0\;\;0\;\;R_4\left[\,\cos\alpha_{41}\;\sin\alpha_{41}\;0\,\right]^T \\ \cdots \\ 0\;0\;\;0\;\;R_4\left[\,\cos\alpha_{4n}\;\sin\alpha_{4n}\;0\,\right]^T \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \epsilon_4 \\ \cdots \\ \epsilon_4 \end{bmatrix} - A\boldsymbol{\delta}. \tag{22.18}$$

22.3.5 Stay Above a Plane (VF5)

This basic constraint is to prevent the task frame position from penetrating the given plane Π. From the given plane $\Pi(s)$, we can easily compute $\mathbf{x}_p^{cl}$ on $\Pi(s)$ which is the closest point to $\mathbf{x}_p$. The signed errors are set as $\boldsymbol{\delta} \equiv [\boldsymbol{\delta}_p^T, \boldsymbol{\delta}_r^T]^T = [(\mathbf{x}_p - \mathbf{x}_p^{cl})^T, \mathbf{0}^T]^T$.

As shown in Fig. 22.3, this constraint can be expressed by

$$\hat{\mathbf{d}}^{\,T} \cdot (\boldsymbol{\delta}_p + \Delta\mathbf{x}_p) \geq 0 \tag{22.19}$$

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

$$\begin{aligned} A &= \left[\,-\hat{\mathbf{d}}^{\,T}\;0\;0\;0\,\right], \\ \mathbf{b} &= -A\boldsymbol{\delta} \end{aligned} \tag{22.20}$$

If we further want to confine the task frame position on the plane, we can add constraints

$$\hat{\mathbf{d}}^{\,T} \cdot (\boldsymbol{\delta}_p + \Delta\mathbf{x}_p) \leq \epsilon_5 \tag{22.21}$$

where ϵ_5 is a small positive number, which defines the range of error tolerance. Then A and $\mathbf{b}$ are set as

$$A = \begin{bmatrix} -\hat{\mathbf{d}}^{\,T} & 0\;0\;0 \\ \hat{\mathbf{d}}^{\,T} & 0\;0\;0 \end{bmatrix}, \; \mathbf{b} = \begin{bmatrix} 0 \\ \epsilon_5 \end{bmatrix} - A\boldsymbol{\delta}. \tag{22.22}$$

All the small values $\epsilon_{1,2,3,4,5}$ specify how much the robot can drift away from the reference direction. If they are small (close to zero), then the user can only move the manipulator along the desired direction. If they are relatively large, then the user has more freedom to deviate from the programmed virtual fixture.

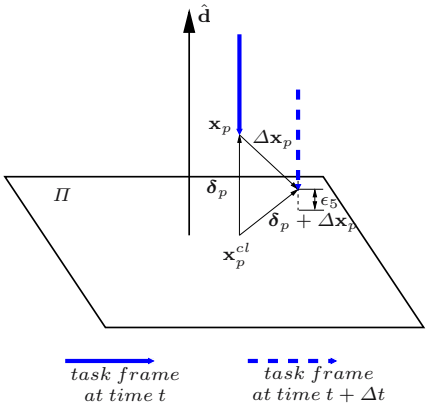

Fig. 22.3. Geometric relation for "stay above a plane"

22.4 Applications and Experiments

In this section we demonstrate customized virtual fixtures generation using the virtual fixture library on task primitives [22, 27]. As a preliminary test-bed we have used a "remote center of motion" JHU Steady Hand robot [24], which is equipped with a tool holder and a 6-DoF force-torque sensor (ATI Nano43 F/T transducer) mounted on the tool handle. The Optotrak (Northern Digital Inc, Waterloo, CA) infrared optical position tracking system was used for robot calibration. Our control algorithm is independent of manipulator characteristic. Moreover, the desired user input can be obtained either from a master robot, a joystick or a force sensor attached to the robot.

22.4.1 Application Task 1: Path-Following

Numerous surgical situations require surgeons to follow a predetermined path, while maintaining certain other constraints. One such example is the cutting procedure for a knee prosthetic implant. Knee prosthetic implants are used to replace the bearing surfaces of the knee. Normally when done by manual technique, the surgeon positions cutting blocks (jigs) on the bone and then cuts the femur and tibia to the required shape using a hand held saw to mount the prosthetic components in position. In the robotic procedure adopted by systems such as Robodoc [28], Acrobat [29] and CASPAR [30], a mill is used for the cutting procedure, and the blade is required to cut along the planned path on the bone. Meanwhile, the cutting edge of the tool should be kept perpendicular to the cutting plane in order to provide more efficient force.

Modeling of Task

We model the femur cutting task as a task to guide the tip of a long straight tool following a 2D b-spline curve C_1 in plane Π while keeping the tool shaft

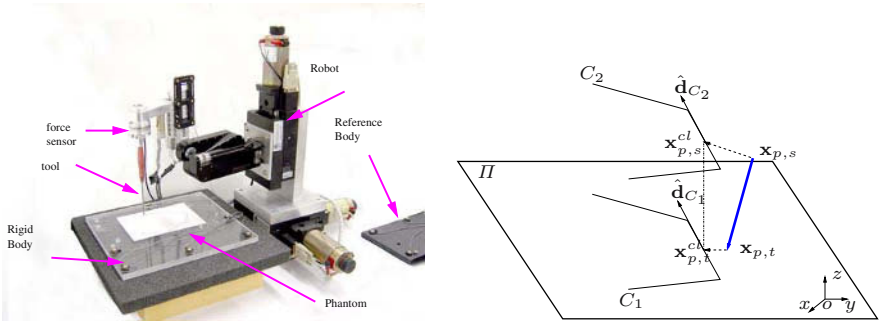

Fig. 22.4. (left) Experimental setup, (right) Geometric relation for the task "path-following"

perpendicular to the plane. The geometric relation is shown in Figure 22.4. We assume that the path C_1 and the cutting plane are known in the robot coordinate frame by using an appropriate registration method. During the procedure, the tip of the tool (task frame $\{t\}$) is allowed to move along the planned path C_1. At the same time, a point, $\mathbf{x}_{p,s}$ on the tool shaft (task frame $\{s\}$) is only allowed to move along the second path C_2, which is a translation of C_1 above the target plane. $\mathbf{x}_{p,t}^{cl}$ is the closest point to the tip of the tool on C_1 and $\mathbf{x}_{p,s}^{cl}$ is the projection of $\mathbf{x}_{p,t}^{cl}$ on C_2. We use each of these points and the tangent at these points to create two sets of constraints according to formulation VF3 in Sec. 22.3.3.

Experimental Results

We mounted a straight tool on the robot end-effector. We drew a set of line segments on a flat plastic plate, assumed to be a plane, and attached Optotrak LEDs to the plate. We used a digitizer to gather sample points on the line segments and then we generated a 5th degree b-spline curve in the target coordinate frame by interpolating these sample points. We used an 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 $0.32 \pm 0.19\, mm$. The trajectory of the tool tip with respect to the b-spline curve and the error profile of a trial are shown in Figure 22.5. The large 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 large errors on the sharp turnings where the tangent direction changes dramatically.

To evaluate the effect of the communication delay, we compared the tip error with different time intervals for the control loop. We removed the communication between the robot and Optotrak from the loop, only robot encoders and kinematics were used to record the tool tip motion. As shown in Fig. 22.6, the error

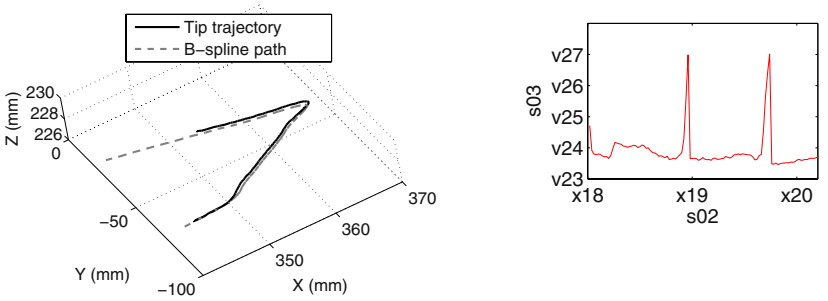

Fig. 22.5. (left) The trajectory of the tool tip with respect to the reference b-spline curve (right) The magnitude of the tool tip position error measured by Optotrak in "path-following" task

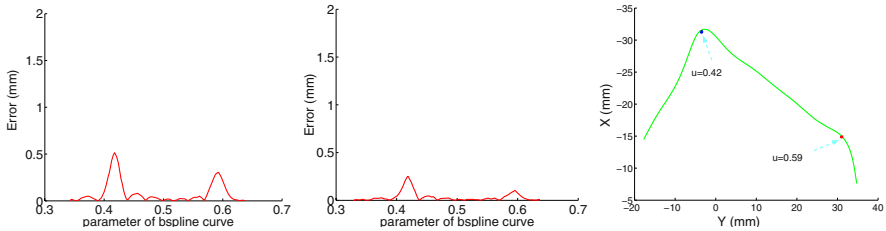

Fig. 22.6. The magnitude of the tool tip position error measured by the robot encoders and kinematics with different time interval for each control loop in "path-following with fixed orientation" task (left) around 150ms each loop, (middle) around 40ms each loop, (right) bspline curve and the position on which the large errors occur.

at the sharp turning decreased in the case that the time interval of each loop is shorter. In this case, the velocity is updated more frequently. Especially at the sharp turning, before the tool tip goes too far, the new velocity is computed and applied. The prompt action of the robot reduces the error. With the same time interval ($150\,ms$), the error recorded by the Optotrak is larger than that by the robot encoders and kinematics. This is due to the system registration error and the accuracy of the Optotrak.

22.4.2 Application Task 2: Virtual RCM

In percutaneous needle insertions and also in robotic-assisted minimally invasive surgery, the surgical tools are inserted into the human body through a port. It is highly desireable to limit the motion of the tools at the entry port, and provide sufficient degrees of freedom for manipulation of tools inside the body. Some surgical robots [3, 23, 25] constrain the tool motion by providing a mechanical isocenter mechanism also known as remote center of motion (RCM). This task

demonstrates a virtual remote center of motion (RCM) configuration, which provides an isocentric motion that is fundamental to percutaneous procedures. Our virtual fixture paradigm can implement virtual RCM on any given position other than mechanical RCM. Moreover, it can also be used in robots which do not provide a mechanical RCM.

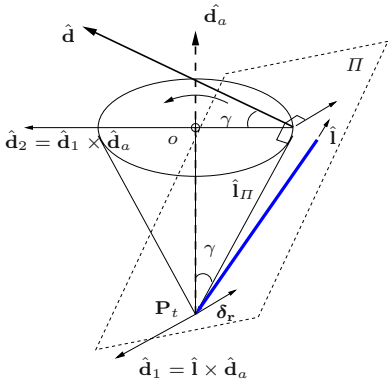

Fig. 22.7. Geometric relation for the task "Virtual RCM"

Modeling of Task

A surgical procedure using the virtual RCM can be modeled as to keep a selected position on the tool staying on a given insert port on a patient skin, while moving the orientation of the tool to follow the preplanned trajectory. The position and tool trajectory can be defined by the surgeon. In our work, we simplify the task as to keep the tip of a straight tool pivoting on a given point $\mathbf{P}_t$ (which is other than any mechanical RCM of the given robots) while rotating the tool shaft around a given direction $\hat{\mathbf{d}}_a$ with a fixed angle γ. The cone shape in Fig. 22.7 shows the desired trajectory of the tool shaft. We have combined the motion primitives VF1 and VF4 as presented in sections 22.3.1 and 22.3.4 respectively to create the virtual fixture for this task.

Experimental Results

For the experiment we set the given direction as $\mathbf{d}_a = [0\ -0.2\ 1]^T$, the fixed angle as $\gamma = 15\,\text{deg}$ and the pivot point as $\mathbf{P}_t = [0\ 0\ -10]^T\,(mm)$ with respect to the robot mechanical RCM. The values (ϵ_i) for positional and angular tolerance was selected as $0.1mm$ and $0.2deg$ respectively.

The trajectory of the tool and the error profile, which are measured by the robot encoders, are shown in Figure 22.8. The average pivot point position error for five trials is $0.01 \pm 0.01\ mm$ measured by the robot encoders. The average tool orientation angle error is $0.03 \pm 0.02\,\text{deg}$.

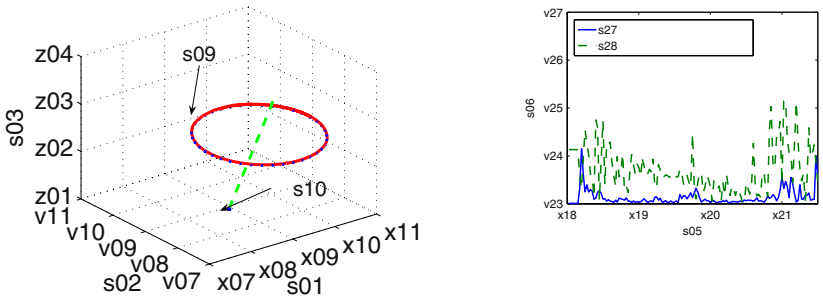

Fig. 22.8. Experimental results of the "virtual RCM" task measured by the robot itself. (left)The trajectories of a point on the tool shaft and the virtual RCM points. The circle shows the actual trajectory of a point on the tool shaft, the dots close to the circle represents the desired trajectory of the point. (right) The error profile for both the pivot point position error and the tool orientation angle error.

22.4.3 Application Task 3: Suturing

Suturing is considered to be one of the most difficult and time consuming minimally invasive surgical procedures. The surgeon faces the challenge of the limited and constrained motion as well as the loss of direct visualization. The suturing task was observed and analyzed as performed in training videos. This task involves the following steps 1) (Select) Determine a suitable entry and exit point for the suture needle leaving sufficient room from the edge to be sutured together. 2) (Align) Grasp the needle, move and orient it such that the tip is aligned with the entry point. 3) (Bite) Entry and exit "bites" are made such that the needle passes from one tissue to the other. 4) (Loop) Create a suture loop to tie a knot. 5) (Knot) Secure the knot under proper tension.

In this application, we address the align and bite steps of the suturing process, where the primary challenges are manipulation of a curved needle under non-ideal haptic conditions using a robot with complex kinematics. In the align step, the goal is to move the robot to align the position and the orientation of the suture needle such that it pierces the tissue correctly, while minimizing extraneous motion of the needle and robot. The goal of the bite step is to move the needle tip from the entry point to the exit point with minimum damage to the tissue through which the needle passes.

Task Modeling

We assume that the entry and the exit points are known in the robot coordinate frame. These could be specified by the surgeon using a tracked instrument or by using a computer vision system registered to the robot coordinate frame, to determine suitable points on the surface based on distance from the edge to be sutured together. Fig. 22.9 (top left) shows the various task frames $\{i\}$ associated with the suturing task.

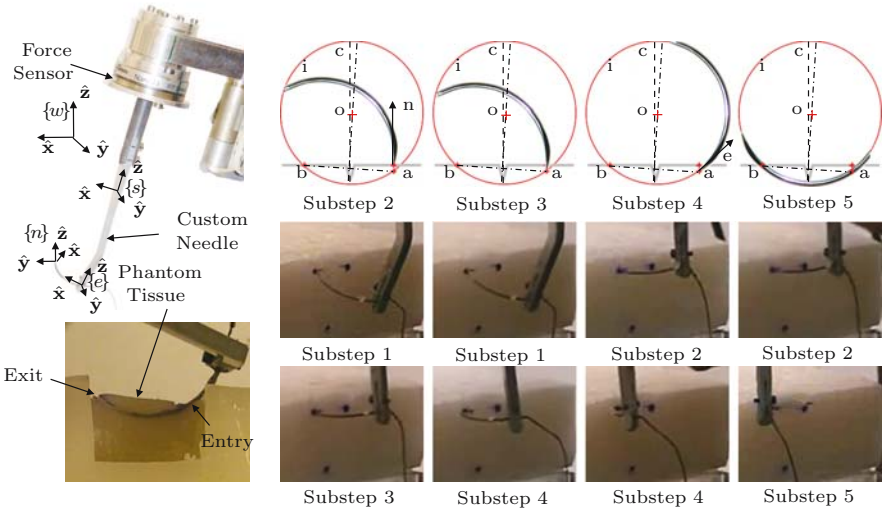

Fig. 22.9. (top left) Custom needle holder, needle and assigned frames. (bottom left) Actual path of needle. Top of phantom was removed after the expermient for verification. (top right) Sketch of sequence of substeps. a - Entry Point, b - Exit Point, c - Center line of wound, e - Entry direction, i - Ideal path of needle, n - Surface normal, o - Center of rotation. (bottom right) Sequence of substeps in phatom.

In our approach the required VF constraints for each substep are analyzed and broken into a combination of one or more of the basic constraints. We make use of the common structure between the different substeps and construct generalized constraints that take the desired target into consideration. Furthermore we utilize the sequential nature of the task to switch between different substeps. The switch could be triggered when the error between the current value and target decreases below a threshold. Fig. 22.9 shows these substeps, along with the entry, and the exit points on a phantom tissue.

Align Step. *(Substep 1)* First the needle tip is allowed to move in a straight line such that the needle tip coincides with the desired entry point; at the same time its orientation is allowed to change only about an axis such that this motion will result in the tangent at the needle tip being coincident with the normal to the surface at the entry point by using primitives VF3 and VF4. *(Substep 2)* In the next substep, the orientation of the normal to the needle plane is allowed to change, such that the needle plane coincides with the line joining the entry and exit points. Assistance is provided by not allowing any motion of the needle tip or the tangent at the needle tip by using the using primitives VF1 and VF2 respectively. *(Substep 3)* Once the desired orientations are reached we allow the surgeon to penetrate the tissue by a small distance, *(Substep 4)* followed by motion constraints that would let the surgeon bring the tangent at the needle tip to coincide with the desired entry direction without changing the plane normal and tip position by using primitives VF2 and VF1 repectively. The align step is

completed once the desired orientation is reached, which is computed using the entry and exit points specified by the surgeon and the needle radius. In all these substeps only those motions that bring the needle closer to the desired position and orientation are allowed.

Bite Step. *(Substep 5)* Once the entry and exit points are determined, and the radius of needle is known, clearly the trajectory of the needle tip that would cause minimum damage to the tissue lies on a circle with the entry and exit points as points on a chord and with radius equal to the needle radius. To ensure sufficient depth of penetration in the tissue we ensure that the needle plane is parallel to the line joining the entry and exit points and the surface normal at the entry point. In this step our constrained motion algorithm permits only those motions that satisfy these constraints by using primitives VF1 and VF2 for needle center and a normal to the needle plane respectively.

Experimental Results

For these experiments we selected a 3/8 circle $30mm$ cutting needle from Ethicon (needle diameter $1mm$). We recorded the encoder readings of the robot joints and used direct kinematics of the robot to verify our algorithm by measuring the errors between the ideal target path and that followed by the robot. Fig. 22.9 (bottom right) shows the progression of different substeps for one of the trials. Fig. 22.9 (bottom left) shows the phantom with a portion cut out so that the actual path taken by the needle is visible, the entry and exit points are $13.5mm$ apart. As seen in Fig. 22.9, we have selected an angle that places limits on performing the suture manually, to emphasize the ability of our algorithm to assist in non-favorable orientations.

Table 22.1. The error (mm) in ideal and actual points as measured by the Optotrak

	Entry	Exit
Robot	0.6375	0.7742
Manual	-	2.1

Fig. 22.10 shows the errors between actual and ideal robot motion as measured by the robot encoders and kinematics for different substeps. The values (ϵ_i) for positional and angular tolerance were selected as $0.5mm$ and $0.25deg$. We also demonstrate our algorithm using a phantom tissue. Since the phantom is opaque, the measurements available are the entry and exit points of the needle. Table 22.1 presents the differences between the user specified targets and the actual ones as measured by the Optotrak.

As expected, the errors measured by the Optotrak are higher than measured by encoders alone, because this represents the overall accuracy of the system, which also includes errors arising from calibration of the needle and accuracy of the Optotrak ($0.1mm$). The residual calibration errors appear as errors in

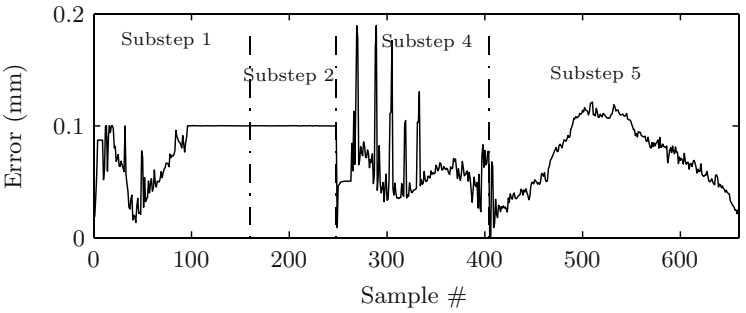

Fig. 22.10. The magnitude of error in the needle tip with respect to ideal path as measured by robot encoders

the entry point errors in Table 22.1. Average errors for free hand suturing as performed by four users (5 trials each), using the same needle holder and without robot assistance, are presented in Table 22.1. We believe that robot assistance can improve accuracy especially in a constrained environment such as that of endoscopic surgery. Moreover, robot assisted motions did not require multiple trials and large undesirable movements of tissue, which is often the case in free hand suturing.

22.5 Conclusion

This chapter described a new method to generate spatial motion constraints for surgical robots that provide sophisticated ways to assist surgeons. Our approach is based on the optimized constrained control. We set the objective function based on the user input that can be obtained through a force sensor, joystick or a master robot. We set the linearized subjective function based on five basic geometric constraints. The combinations of one or more basic geometric constraints for the same or different task frames could generate customized virtual fixtures for complicated surgical tasks. Theoretically, different virtual fixtures can be implemented by using this method if we know the instantaneous kinematics of the manipulator and the geometric constraints [31].

Our approach provides the link between surgeon-understandable task behaviors and low level control for surgical assistance robots. The strength of this approach is that it is extensible to include additional constraints that are important in robotic assisted surgery, such as collision avoidance, anatomy-based constraints and joint limits by using the instantaneous kinematic relationship between the task variables and the robot joints. In [32, 33], we extended our algorithm to create anatomy-based motion constraints for a path-following task in a constrained workspace. We integrated a 3-D geometric model of the workspace to generate virtual fixtures to guide the tool tip along the paths while preventing the tool shaft from entering forbidden regions for sinus surgery.

References

1. P. Berkelman, P. Cinquin, J. Troccaz, J. Ayoubi, C. Letoublon, and F. Bouchard. A compact, compliant laparoscopic endoscope manipulator. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1870–1875, 2002.
2. J. Rosen, J. D. Brown, L. Chang, M. Barreca, M. Sinanan, and B. Hannaford. The bluedragon - a system for measuring the kinematics, the dyanmics of minimally invasive surgical tools in-vivo. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1876–1881, 2002.
3. G. S. Guthart and J. K. Salisbury. The intuitive telesurgery system: Overview, application. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 618–621, 2000.
4. M. Chodoussi, S. E. Butner, and Y. Wang. Robotic surgery-the transatlantic case. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1882–1888, 2002.
5. M. C. Cavusoglu, W. Williams, F. Tendick, and S. S. Sastry. Robotics for telesurgery: second generation berkeley/ucsf laparoscopic telesurgical workstation, looking toward the future applications. *Industrial Robot, Special Issue on Medical Robotics*, 30(1):22–29, 2003.
6. P. S. Schenker, H. Das, and R. T. Ohm. Development of a new high-dexterity manipulator for robot-assisted microsurgery. In *Proc. SPIE - The International Society for Optical Engineering: Telemanipulator and Telepresence Technologies*, 2351:191–198, 1995.
7. P. S. Jensen, K. W. Grace, R. Attariwala, J. E. Colgate, and M. R. Glucksberg. Toward robot-assisted vascular microsurgery in the retina. *Graefes Archive for Clinical and Experimental Ophthalmology*, 235(11):696–701, 1997.
8. M. Mitsuishi, Y. Iizuka, H. Watanabe, H. Hashizume, and K. Fujiwara. Remote operation of a micro-surgical system. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1013–1019, 1998.
9. R. H. Taylor, H. A. Paul, P. Kazanzides, B. D. Mittelstadt, W. Hanson, J. F. Zuhars, B. L. Musits B. Williamson, E. Glassman, and W. L. Bargar. An image-directed robotic system for precise orthopaedic surgery. *IEEE Transactions on Robotics and Automation*, 10(3):261–275, 1994.
10. J. Wurm, H. Steinhart, K. Bumm, M. Vogele, C. Nimsky, and H. Iro. A novel robot system for fully automated paranasal sinus surgery. In *International Congress Series*, 1256:633–638, 2003.
11. I. W. Hunter, T. D. Doukoglou, S. R. Lafontaine, P. G. Charette, L. A. Jones, M. A. Sagar, G. D. Mallinson, and P. J. Hunter. A teleoperated microsurgical robot, associated virtual environment for eye surgery. *Presence*, 2(4):265–280, 1993.
12. M. Mitsuishi, T. Watanabe, H. Nakanishi, T. Hori, H. Watanabe, and B. Kramer. A tele-micro-surgery system with co-located view, operation points, a rotational-force-feedback-free master manipulator. In *2nd Int. Symp. Medical Robotcis and Computer-Assisted Surgery (MRCAS)*, pages 111–118, 1995.
13. M. Mitsuishi, H. Watanabe, H. Nakanishi, H. Kubota, and Y. Iizuka. Dexterity enhancement for a tele-micro-surgery system with multiple macro-micro co-located operation point manipulators, understanding of the operator's intention. In *3rd Int. Symp. Medical Robotcis and Computer-Assisted Surgery (MRCAS)*, pages 821–830, 1997.

14. S. E. Salcudean and G. Bell S. Ku. Performance measurement in scaled teleoperation for microsurgery. In *1st Int. Symp. Medical Robotics and Computer-Assisted Surgery (MRCAS)*, pages 789–798, 1997.
15. B. L. Davies, S. J. Harris, W. J. Lin, R. D. Hibberd, R. Middleton, and J. C. Cobb. Active compliance in robotic surgery - the use of force control as a dynamic constraint. In *Proc. Inst. Mech. Eng. H*, 211(4):285–292, 1997.
16. S. Park, R. D. Howe, and D. F. Torchiana. Virtual fixtures for robotic cardiac surgery. In *Proc. Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 1419–1420, 2001.
17. A. Bettini, P. Marayong, S. Lang, A. M. Okamura, and G. D. Hager. Vision assisted control for manipulation using virtual fixtures. In *IEEE Transactions on Robotics*, 20(6):953–966, 2004.
18. R. Kumar, G. D. Hager, A. Barnes, P. Jensen, and R. H. Taylor. An augmentation system for fine manipulation. In *Proc. Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 956–965, 2000.
19. P. Marayong, M. Li, A. M. Allison, and G. D. Hager. Spatial motion constraints: theory, demonstrations for robot guidance using virtual fixtures. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1954–1959, 2003.
20. D. Aarno, S. Ekvall, and D. Kragic. iadaptive virtual fixtures for machine-assisted teleoperation tasks. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1151–1156, 2005.
21. J. Funda, R. H. Taylor, B. Eldridge, S. Gomory, and K. G. Gruben. Constrained cartesian motion control for teleoperated surgical robots. *IEEE Transactions on Robotics and Automation*, 12(3):453–465, 1996.
22. M. Li, A. Kapoor, and R. H. Taylor. A constrained optimization approach to virtual fixtures. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 2924–2929, 2005.
23. R. H. Taylor, J. Funda, B. Eldridge, K. Gruben, D. LaRose, S. Gomory, M. Talamini, L. R. Kavoussi, and J. Anderson. A telerobotic assistant for laparoscopic surgery. *IEEE Eng. Med. Biol. Mag.*, 14:279–287, 1995.
24. R. H. Taylor, P. Jensen, L. L. Whitcomb, A. Barnes, R. Kumar, D. Stoianovici, P. Gupta, Z. Wang, E. deJuan, and L. Kavoussi. A steady-hand robotic system for microsurgical augmentation. *International Journal of Robotics Research*, 18(12):1201–1210, 1999.
25. M. Cavasoglu, F. Tendick, M. Cohn, and S. Sastry. A laparoscopic telesurgical workstation. *IEEE Transactions on Robotics and Automation*, 15(4):728–739, 1999.
26. C. Lawson and R. Hanson. *Solving Least Squares Problems.* Prentice-Hall, 1974.
27. A. Kapoor, M. Li, and R. H. Taylor. Spatial motion constraints for robot assisted suturing using virtual fixtures. In *Proc. Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 89–96, 2005.
28. U. Wiesel and M. Boerner. First experiences using a surgical robot for total knee replacement. In *Proc. Computer Assisted Orthopaedic Surgery (CAOS Intl)*, pages 143–146, 2001.
29. S. J. Harris, K. L. Fan, R. D. Hibberd, and B. L. Davies. Experiences with robotic systems for knee surgery. In *Proc. 3rd Int. Conf. Medical Robotics and Computer Assisted Surgery*, pages 757–766, 1997.
30. S. Mai and W. Siebert. Planning and technique using the robot system 'caspar' for tkr. In *Proc. Computer Assisted Orthopaedic Surgery (CAOS Intl)*, pages 278–288, 2001.

31. A. Kapoor, N. Simaan, and R. H. Taylor. Suturing in confined spaces: constrained motion control of a hybrid 8-dof robot. In *Proc. IEEE Int. Conf. Advanced Robotics*, pages 452–459, 2005.
32. M. Li and R. H. Taylor. Optimum robot control for 3d virtual fixture in constrained ent surgery. In *Proc. Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 165–172, 2003.
33. M. Li and R. H. Taylor. Spatial motion constraints in medical robot using virtual fixtures generated by anatomy. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1270–1275, 2004.