Proceedings of the 2006 IEEE International Conference on Robotics and Automation
Orlando, Florida - May 2006

Constrained Control for Surgical Assistant Robots

Ankur Kapoor, Ming Li and Russell H. Taylor
Dept. of Computer Science
Johns Hopkins University
{kapoor,liming,rht@cs.jhu.edu}

Abstract— **This paper presents an approach to implement virtual fixtures for surgical robot assistants. Our approach uses a weighted, multi-objective (both linear and nonlinear) constrained optimization framework to formalize a library of virtual fixtures for task primitives. By our formulation, we provide a library of virtual fixtures on task primitives and a way to assemble multiple virtual fixture objects. We implement the constrained optimization problem with both linear and nonlinear constraints, and discuss the trade-offs between them. Moreover, we introduce the notion of "soft" virtual fixture mechanism for robotic surgical assistance. The "soft" virtual fixtures enable a surgical tool to have some resistance inside safety regions and no resistance in preferred regions.**

I. Introduction

This paper presents an approach to implement virtual fixtures for surgical robot assistants. Most robotic assisted surgical procedures are characterized by restricted access to the workspace as well as constrained manipulation of a surgical tool. In such cases, the surgeons' ability can be augmented by techniques such as virtual fixtures (VF). Virtual fixtures [1], which have been discussed previously in the literature for both telerobotic and cooperative robots, are algorithms which provide anisotropic behavior to surgeons' motion commands in addition to filtering out tremor to provide safety and precision.

An important case of virtual fixtures is forbidden regions, where the surgical tool is restricted to certain regions in the workspace. Davies *et al.* [2] set active constraints to constrain the robot to cut the femur and tibia within a permitted region for prosthetic knee surgery. Park *et al.* [3] 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. The recent work by Bettini *et al.* on virtual fixtures [4] used admittance control laws to implement guidance virtual fixtures. These works are based either on a specific robot type or on a specific task.

Path planning and motion control is a well discussed area with a wide variety of proposed optimality criteria [5], [6], [7], [8]. Funda *et al.* [9] presented an optimal motion control method to control both redundant and deficient robotic systems in constrained working volumes. We extend Funda's work by applying the method to generate complicated virtual fixtures based on user input for surgical assistant robots.

Typically, surgical tasks have a certain degree of uncertainty that arises from factors such as registration errors, variations in anatomy and changes during procedures. Consider an example task of placing a surgical tool at a point in space. Depending on the nature of the procedure, one can define a region and tool placement within this region that would lead to the expected outcome. This region could be on the order of a few microns for retinal vein cannulation or hundreds of microns for a biopsy procedure. Furthermore, we can define another region where the surgeon might deliberately want to place the instrument to account for some uncertainties inherent in surgical procedures. In other words, we would like to have some compliance in the virtual fixture, while maintaining a preferred motion. Therefore, we define 3 different regions:

A) Preferred region: this region defines expected outcome.
B) Safety region: the tool could temporarily be in this region for fulfilling some expected task.
C) Forbidden region: The tool never could be here for safety purposes.

The relationship of these three regions depends on the surgical task. Figure 1 shows two typical examples.

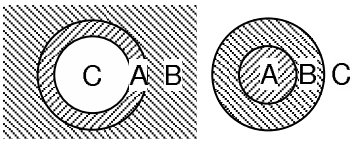


Fig. 1. Two examples of relationship between (A) Preferred region, (B) Safety region and (C) Forbidden region.

In this work we formalize a library of virtual fixtures for task primitives. Our paradigm covers the implementation of guidance virtual fixtures, and forbidden region virtual fixtures, with both "hard" and "soft" constraints. Our approach uses a weighted, multi-objective optimization framework to solve for incremental joint motion given the instantaneous kinematics of the manipulator and the geometric constraints. Note that we treat the robot as a purely kinematic device, independent of manipulator characteristic: teleoperative or cooperative controlled; admittance or impedance type.

In this paper, we first describe a general constrained motion control form for virtual fixtures, followed by constraint setup of the optimization problem for five basic task primitives. For each constraint type we explore the effects on performance of "hard" and "soft" fixtures. We compare two different algorithms for implementation of these constraints. Finally, we report and discuss the experimental results.

II. Constrained Motion Control for Virtual Fixtures

Virtual fixtures are task-dependent computer-generated constraints, which help a robotic manipulator perform a task by

0-7803-9505-0/06/$20.00 ©2006 IEEE

limiting its movement into restricted regions and/or influencing its movement along desired paths. In considering virtual fixtures for a surgical assistant robot, it is important to be able to place absolute bounds on the spatial motion of the different parts of the instrument as well as to specify desired nominal motions.

We define different task frames associated with different parts of the instrument. For each of the task frames, we define actual state variables $\boldsymbol{x}$ and desired state variables $\boldsymbol{x}^d$. The state, $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{q}+\Delta\boldsymbol{q})$ is a function of joint variables $\boldsymbol{q}$ and joint incremental motion $\Delta\boldsymbol{q}$. The desired state, $\boldsymbol{x}^d = \boldsymbol{x}^d(\boldsymbol{\tau},\ \boldsymbol{q})$ is a function of users input $\boldsymbol{\tau}$ and joint variables $\boldsymbol{q}$.

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:

$$\begin{aligned} \Delta\boldsymbol{q}_{cmd} = \arg\min_{\Delta\boldsymbol{q}} & C(\boldsymbol{x}(\boldsymbol{q}+\Delta\boldsymbol{q}), \boldsymbol{s}, \boldsymbol{x}^d) \\ \text{s.t.} \quad & A(\boldsymbol{x}(\boldsymbol{q}+\Delta\boldsymbol{q}), \boldsymbol{s}) \leq \boldsymbol{b}, \\ \boldsymbol{s}_{up} \geq s \geq \boldsymbol{s}_{low} \geq 0; & \quad \Delta\boldsymbol{q}_{up} \geq \Delta\boldsymbol{q} \geq \Delta\boldsymbol{q}_{low} \end{aligned} \tag{1}$$

where $C(\boldsymbol{x}(\boldsymbol{q}+\Delta\boldsymbol{q}), \boldsymbol{s}, \boldsymbol{x}^d)$ is the objective function associated with the difference between the actual state variables $\boldsymbol{x}$ and the desired state variables $\boldsymbol{x}^d$. The inequality $A(\boldsymbol{x}(\boldsymbol{q}+\Delta\boldsymbol{q}), \boldsymbol{s}) \leq \boldsymbol{b}$ represents the constraint conditions. $\boldsymbol{s}$ is a vector of slack variables s_j. These constraints are used to force the solution vector $\Delta\boldsymbol{q}_{cmd}$ to satisfy certain critical requirements, such as restricting the motion of a certain part of the instrument within a strict motion envelope. The inclusion of $\boldsymbol{s}$ in the objective function provides a means of implementing "soft" constraints which may not need to be strictly enforced.

We can generate complicated constrained motions by combining the constrained motions on different task frames. If w_i is the weight associated with the task frame $\{i\}$ then the complicated virtual fixtures generated by constraining task frames $\{i,\ (i = 1,\ ...,\ N)\}$ can be formulated as

$$\begin{aligned} \Delta\boldsymbol{q}_{cmd} = \arg\min_{\Delta\boldsymbol{q}} & \sum_{i=1}^{N} w_i C_i(\boldsymbol{x}_i(\boldsymbol{q}+\Delta\boldsymbol{q}), \boldsymbol{s}_i, \boldsymbol{x}_i^d) \\ \text{s.t.} \quad & A_i(\boldsymbol{x}_i(\boldsymbol{q}+\Delta\boldsymbol{q}), \boldsymbol{s}_i) \leq \boldsymbol{b}_i, \\ \boldsymbol{s}_{i,up} \geq \boldsymbol{s}_i \geq \boldsymbol{s}_{i,low} \geq 0; & \quad \Delta\boldsymbol{q}_{up} \geq \Delta\boldsymbol{q} \geq \Delta\boldsymbol{q}_{low} \\ i = 1, \cdots, N. & \end{aligned} \tag{2}$$

where w_i gives 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 allows us to exploit the geometry of a particular task space motion and effectively trade off the various performance criteria.

Surgical robots often are kinematically redundant for the purpose of providing dexterous assistance. At the same time, task constraints such as the requirement that a tool pass through a cavity restricts dexterity [9]. Indeed, some special-purpose designs for minimally invasive surgery, such as the IBM LARS [10] and the JHU Steady Hand robot [11], may be kinematically deficient. Other robots such as the Intuitive daVinci [12] and Endorobotics [13] combine a kinematically constrained remote center of motion (RCM) mechanism with a kinematically redundant wrist. The ability to accommodate unique, special purpose mechanical designs (such as kinematically redundant or deficient) is important as well. Our formulation could easily integrate any behavior, such as asserting joint limitation for kinematically redundant robot or incorporating haptic information in the control strategy.

We discuss virtual fixtures for five task primitives. We model the robot task frame as a purely kinematic Cartesian device with the tool position $\boldsymbol{x}_p \in R^3$ and the tool orientation given by unit vector $\hat{\boldsymbol{l}}_t \in R^3$. The names and descriptions of these task primitives are listed.

1) **Stay on a point**: Keep the tool position $\boldsymbol{x}_p$ on the reference position $\boldsymbol{x}_0$.
2) **Maintain a direction**: Keep the tool orientation $\hat{\boldsymbol{l}}_t$ aligned with the reference direction $\hat{\boldsymbol{l}}_r$.
3) **Move along a line**: Keep the tool position $\boldsymbol{x}_p$ on line L which has the direction $\hat{\boldsymbol{l}}_r$ and passes through point $\boldsymbol{x}_0$. At the same time, the tool should move along L proportional to the users input $\boldsymbol{\tau}$.
4) **Rotate around a line**: Keep the tool orientation $\hat{\boldsymbol{l}}_t$ perpendicular to line L which has the direction $\hat{\boldsymbol{l}}_r$ and passes through point $\boldsymbol{x}_0$. At the same time, the tool should rotate around L proportional to the users input $\boldsymbol{\tau}$.
5) **Stay above a plane**: Keep the tool position $\boldsymbol{x}_p$ above the plane Π that has the normal direction $\hat{\boldsymbol{d}}$ pointing to the free half space and passes through point $\boldsymbol{x}_0$. At the same time, the movement of the tool should be proportional to the users input $\boldsymbol{\tau}$.

For each of these primitives, we can define a desired nominal behavior together with constraints specifying how far actual behavior can differ from the nominal. Terms are added to the objective function to derive the desired motion related to the user's input, while the constraints place an absolute bound on the motion. Table I shows the nominal errors and constraints for five task primitives.

By our formulation, we provide a library of virtual fixtures on task primitives and a way to assemble multiple virtual fixture objects. Customized virtual fixtures for complicated surgical tasks can be treated as the combination of one or more objects assigned on single or multiple task frames. Figure 2 shows the concept of generating customized virtual fixtures from the virtual fixture library.

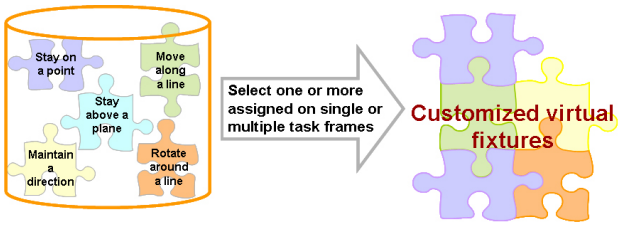


Fig. 2. Customized virtual fixture

TABLE I

THE NOMINAL ERROR TERMS AND CONSTRAINTS FOR FIVE TASK PRIMITIVES

Task Primitives	Nominal Error Term	Constraints
Stay on a point	$\boldsymbol{\delta}_p = \boldsymbol{x}_p - \boldsymbol{x}_0$	$\frac{1}{2}\|\boldsymbol{\delta}_p\|_2^2 \leq \epsilon_m$
Maintain a direction	$\boldsymbol{\delta}_r = \hat{\boldsymbol{l}}_t \times \hat{\boldsymbol{l}}_r$	$\frac{1}{2}\|\boldsymbol{\delta}_r\|_2^2 \leq \epsilon_m$
Move along a line	$\boldsymbol{\delta}_p = R_p^T[\boldsymbol{x}_p - \boldsymbol{x}_0 - [(\boldsymbol{x}_p - \boldsymbol{x}_0)\cdot\hat{\boldsymbol{l}}_r]\hat{\boldsymbol{l}}_r]$	$\|\boldsymbol{\delta}_p(1)^2 + \boldsymbol{\delta}_p(2)^2\|_2^2 \leq \epsilon_m$
Rotate around a line	$\boldsymbol{\delta}_r = R_r^T \times[\frac{\hat{\boldsymbol{l}}_t - \hat{\boldsymbol{l}}_r(\hat{\boldsymbol{l}}_t\cdot\hat{\boldsymbol{l}}_r)}{\|\hat{\boldsymbol{l}}_t - \hat{\boldsymbol{l}}_r(\hat{\boldsymbol{l}}_t\cdot\hat{\boldsymbol{l}}_r)\|_2} \times\hat{\boldsymbol{l}}_t]$	$\|\boldsymbol{\delta}_r(1)^2 + \boldsymbol{\delta}_r(2)^2\|_2 \leq \epsilon_m$
Stay above a plane	$\hat{\boldsymbol{d}}^T \cdot \frac{\boldsymbol{x}_p - \boldsymbol{x}_0}{\|\boldsymbol{x}_p - \boldsymbol{x}_0\|_2}$	$\hat{\boldsymbol{d}}^T \cdot (\boldsymbol{x}_p - \boldsymbol{x}_0) \geq \epsilon_m$

R_p and R_r are rotation matrices that would transform a plane Π perpendicular to $\hat{\boldsymbol{l}}_p$ and $\hat{\boldsymbol{l}}_r$ respectively to world coordinates. The maximum allowed error is denoted by ϵ_m, a small positive number. $\boldsymbol{\delta}(i)$ represents the i^{th} component of vector $\boldsymbol{\delta}$

III. OUR CONSTRAINED CONTROL ALGORITHM

A. Algorithm Overview

The general form of the optimization problem in (1) has many variants, both for the objective function and constraints. Our objective function is a two-norm, $\|\cdot\|_2$ of motion error in different task frames. We use robotic instantaneous kinematics to map the different task frames to joint variables, and set an optimization problem over incremental joint motion $\Delta\boldsymbol{q}$. The objective function is expressed as

$$\begin{aligned} & C(\boldsymbol{x}(\boldsymbol{q}+\Delta\boldsymbol{q}), \boldsymbol{s}, \boldsymbol{x}^d) \\ &= \left\|\boldsymbol{x}(\boldsymbol{q}+\Delta\boldsymbol{q}) - \boldsymbol{x}^d\right\|_2^2 + \|w_s\boldsymbol{s}\|_2^2 \\ &= \left\|\boldsymbol{x}(\boldsymbol{q}) + J(\boldsymbol{q})\cdot\Delta\boldsymbol{q} - (\boldsymbol{x}(\boldsymbol{q}) + \Delta\boldsymbol{x}^d)\right\|_2^2 + \|w_s\boldsymbol{s}\|_2^2 \\ &= \left\|J(\boldsymbol{q})\cdot\Delta\boldsymbol{q} - \Delta\boldsymbol{x}^d\right\|_2^2 + \|w_s\boldsymbol{s}\|_2^2 \end{aligned} \tag{3}$$

We assume that the robot is holding the surgical instrument. The instruments and all the constraints are known in the robot coordinate frame. The basic control loop repeats the following steps:

Step 1: Describe a desired incremental motion of the surgical instrument based upon user inputs $\boldsymbol{\tau}$. In order to make the system intuitive, the incremental motion should be proportional to the user input.

Step 2: Analyze and decompose the task into task primitives for different task frames. Set the basic geometric constraints on each task frame.

Step 3: Use the robot and the task kinematic equations to produce a new constrained optimization problem, in which the instrument motion variables and other task variables have been projected onto incremental joint variables. This problem has the form:

$$\begin{aligned} \arg\min_{\Delta\boldsymbol{q}} & \sum_{i=1}^{N} w_i\left(\left\|(\Delta\boldsymbol{x}_i - \Delta\boldsymbol{x}_i^d)\right\|_2^2 + \|w_{s,i}\cdot\boldsymbol{s}_i\|_2^2\right) \\ \text{s.t.} \quad & A(\Delta\boldsymbol{x}_i) - \boldsymbol{s}_i \leq \boldsymbol{b}_i, \quad \Delta\boldsymbol{x}_i = J_i\cdot\Delta\boldsymbol{q}, \\ & \boldsymbol{s}_{i,up} \geq \boldsymbol{s}_i \geq \boldsymbol{s}_{i,low} \geq 0, \quad \Delta\boldsymbol{q}_{up} \geq \Delta\boldsymbol{q} \geq \Delta\boldsymbol{q}_{low} \end{aligned} \tag{4}$$

where $\Delta\boldsymbol{q}$ is the desired incremental motions of the joint variables. $\Delta\boldsymbol{x}_i$ and $\Delta\boldsymbol{x}_i^d$, $i = 1, ..., n$ represent computed and desired variables of different task spaces, respectively. $\Delta\boldsymbol{x}^d$ includes the user input $k\boldsymbol{\tau}$ where $\boldsymbol{\tau}$ is the user input and k is a scalar for tuning the ratio between the incremental motion and the desired input. The user input could be obtained from a force sensor or from a master robot. To ensure safety we can also define an upper bound for the incremental motion magnitude. J_i is the Jacobian matrix relating task space i to the robot joint space. w_i defines weights selected so that the errors of critical motion elements are close to zero, while errors in other non-critical motions simply stay as low as possible within tolerances allowed by the constraint set. We must ensure proper scaling of weights corresponding to different components. Otherwise, the result of the optimization $\Delta\boldsymbol{q}$ will be skewed, causing incorrect control behavior. $w_{s,i}$, determines the turning of "softness" of the constraints on the i^{th} task frame. The greater the value of $w_{s,i}$ the harder the constraint, that is it is harder for user to deviate from preferred region.

Step 4: Use known numerical methods [14] [15] to compute incremental joint motions $\Delta\boldsymbol{q}$, and use these results to move the robot.

B. Linear Approximations

There are computational trade-offs between linearly constrained and nonlinearly constrained least squares problems. The algorithms for solving linearly constrained least squares problems (such as least squares inequality methods, active set methods, etc.) are usually less complex than the algorithms for solving the nonlinearly constrained least squares problems (such as reduced-gradient methods, sequential quadratic programming methods, etc.). Solving linearly constrained least squares problem can take less computation time.

In this section we present a specialized form of (1) to produce a quadratic optimization problem with linear constraints of the form of (5). The computation for a linear constrained quadratic optimization problem is efficient and robust.

$$\begin{aligned} A\cdot\Delta\boldsymbol{x} \;-\; \boldsymbol{s} \;&\leq\; \boldsymbol{b}, \\ \boldsymbol{s}_{up} \;\geq\; \boldsymbol{s} \;&\geq\; \boldsymbol{s}_{low} \geq \quad 0, \\ \Delta\boldsymbol{q}_{up} \;\geq\; \Delta\boldsymbol{q} \;&\geq\; \Delta\boldsymbol{q}_{low} \end{aligned} \tag{5}$$

In the linear approximation, we use a set of hyperplanes to bound a polyhedron to approximate a geometric constraint region. For example, the constraints $\frac{1}{2}\|\delta_p\|_2^2 \leq \epsilon_m$ in Table I defines a spherical error tolerance region. We could easily generate a polyhedron bounding the sphere as shown in Figure 3. Obviously, one possible method to guarantee the solution of the linearized optimization problem falling inside the inscribed sphere is to increase the number of the hyperplanes. As the number of the hyperplanes increases, the volume of the polyhedron reduces and the polyhedron approaches the inscribed sphere.

However, more linear constraints require more time to solve the optimization problem. The loss of efficiency can be more prominently noticed in the case of soft constraints, where a slack variable is associated with each constraint. An increase

in the number of constraints implies an increase in the number of variables to solve. Moreover, when linear constraints are used to approximate nonlinear constraints, some error may be introduced into the system and also the number of the constraints can be large if we want a solution close to the original problem.

Figure 3 shows the relation of polyhedron defined by $Ax \leq \boldsymbol{b}$ with different numbers of hyperplanes and the specified spherical error tolerance region.

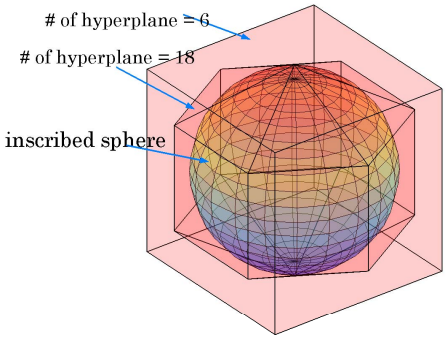


Fig. 3. The polyhedron determined by $Ax \leq \boldsymbol{b}$ with different numbers of hyperplanes. The inscribed sphere defines the ideal error tolerance region.

C. Sequential Quadratic Programming

Currently, sequential quadratic programming methods (SQP) are considered to be the most efficient methods for solving nonlinear programs of small to medium size, and a wealth of these have been developed. We used the method of Spellucci [15], which is available freely via www.netlib.org. For solving the nonlinear constrained optimization problem, we need to have a reasonable initial guess. If the initial guess is ill-conditioned, it will either have a longer computation time to solve the optimization problem, or worse the algorithm will not converge to a reasonable solution. We can make progress here by observing that for typical surgical cases the user input will change at a much slower rate compared to the rate of our control algorithm. This allows us to use the previous incremental motion as the initial guess to compute the current incremental motion. However, we need a strategy to provide a reasonable initial guess for the very first step. A solution is to use the solution of the linear approximation described earlier.

SQPs are iterative algorithms that solve the subproblem (6) given $[\Delta \boldsymbol{q}^k, \boldsymbol{s}^k]^T$ as the guess for the optimal solution $[\Delta \boldsymbol{q}^\star, \boldsymbol{s}^\star]^T$ and a symmetric positive semi-definite matrix B_k. The descent direction $\boldsymbol{d}^k$ which is the solution of the subproblem, along with a step size determines the next estimate $[\Delta \boldsymbol{q}^{k+1}, \boldsymbol{s}^{k+1}]^T$. The methods in the literature differ in strategies to define the active set of equalities $\mathcal{A}_k$ and the technique to define and update B_k. As a rule of thumb, SQPs take about 5-50 iterations, with each iteration being a least squares problem having $n \times m$ constraints in the worst case, where n is the number of variables and m is the number of nonlinear constraints.

$$\begin{aligned} & \nabla C(\boldsymbol{x}(\boldsymbol{q}+\Delta \boldsymbol{q}^k), \boldsymbol{s}^k, \boldsymbol{x}^d)^T \boldsymbol{d} + \tfrac{1}{2} \boldsymbol{d}^T B_k \boldsymbol{d} \\ s.t. \quad & \nabla A_{\mathcal{A}_k}(\boldsymbol{x}_i(\boldsymbol{q}+\Delta \boldsymbol{q}^k), \boldsymbol{s}^k)^T \boldsymbol{d} \leq \boldsymbol{b}_{\mathcal{A}_k}, \end{aligned} \tag{6}$$

Most SQP algorithms can determine the gradients using either simple forward difference, symmetric difference or a higher order approximation. The speed of the computation is greatly enhanced if the gradient can be computed analytically because it avoids evaluations of the constraints. Moreover the precision and the numerical stability are compromised when using numerical gradients due to discretization errors. Table II shows the gradients for the five basic constraints.

TABLE II
THE GRADIENT OF THE CONSTRAINTS FOR THE FIVE TASK PRIMITIVES.

Task Primitives	Gradient of constraints
Stay on a point	$\boldsymbol{\delta}_p^T J_p$
Maintain a direction	$\boldsymbol{\delta}_r^T J_r$
Move along a line	$(H\boldsymbol{\delta}_p)^T (H R_p^T J_p)$
Rotate around a line	$(H\boldsymbol{\delta}_r)^T (H R_r^T J_r)$
Stay above a plane	$\boldsymbol{d}^T J_p$

We denote the Jacobian of the task frame as $J = [J_p; J_r] \in R^{6\times N}$, N being number of joints; For our experimental setup N=7. H is a diagonal matrix given by $\mathrm{diag}(1, 1, 0)$.

IV. EXPERIMENTS

In the conventional laparoscopic approach the surgeons often use the compliance provided by the tissue around the entry port to reach areas otherwise hard to reach. For the existing remote center of motion (RCM) robots such as Intuitive daVinci, IBM LARS and JHU Steady Hand, the RCM is a fixed point in space which is typically aligned at the entry port and the remaining degrees of freedom are then used to perform manipulations. In other words, the mechanism is mechanically constrained at a point, and loses the ability to take advantage of compliance. An analogous situation occurs in the retinal vein cannulation procedure. The task there is to guide a tool that passes through a port to follow a path along the retinal vein until the desired target point for cannulation is reached. The geometric relation is shown in Figure 4. To demonstrate the concept of a "soft" customized virtual fixture we select the above mentioned application, likely to be encountered in many surgical situations. The concept here is that the entry port be constrained within a region, and if the tool is in that region, the surgeon experiences no resistance (Refer figure 1). Outside this region, the surgeon experiences some resistance which is related to the distance from the inner region. Moreover no motion that moves the tool outside of the outer region is permitted. A similar strategy is applied for the tool tip. We compare the performance with both linear and nonlinear constraints, and the performance with both hard and soft constraints.

We use the "JHU Steady-Hand" - a 7-DoF robot which has for $10\mu m$ position resolution our experiments. A 6-DoF force

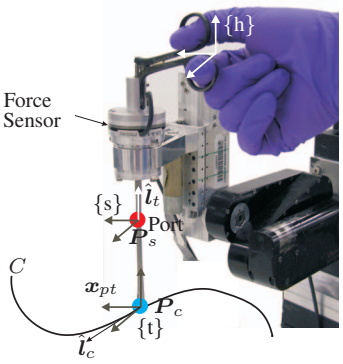


Fig. 4. The Task to guide a tool that passes through a port to follow a path

sensor is integrated at the end-effector. An 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 predefined a sinusoid curve in a virtual environment, shown visually on the computer screen, with $20mm$ amplitude and $15mm$ wavelength which serves as a reference path. We define a position $30mm$ above the center of the sinusoid ($y = \sin(x), x = 0$) as the port.

We implemented virtual fixtures for this task by combining task primitives on two different task frames: the *tool tip frame* $\{t\}$ and the *tool shaft frame* $\{s\}$.

The *tool tip frame* $\{t\}$ 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. We constrain the tip to move along curve C. For each computational loop, the tool tip position $\boldsymbol{x}_{pt}$ and the tool orientation $\hat{\boldsymbol{l}}_t$ are obtained from the robot encoders and its kinematics. We compute the closest point $\boldsymbol{P}_c$ on the sinusoid curve C to $\boldsymbol{x}_{pt}$. The tangent direction $\hat{\boldsymbol{l}}_c$ of the sinusoid on $\boldsymbol{P}_c$ serves as the reference direction.

The *tool shaft frame* $\{s\}$ refers to a coordinate frame whose orientation is parallel to the tool holder. For each time interval, we calculate the point $\boldsymbol{P}_s$ on the tool which is closest to the port. We set the origin of frame $\{s\}$ on $\boldsymbol{P}_s$. We also compute the tool direction $\hat{\boldsymbol{l}}_t$. We constrain the origin $\boldsymbol{P}_s$ of frame $\{s\}$ to move along $\hat{\boldsymbol{l}}_t$.

We combine these two task primitives together to generate the virtual fixtures for the task. We require that our handle motion be proportional to the user's force input $\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{F}_h^T & \boldsymbol{0}^T \end{bmatrix}^T$ in the handle frame $\{h\}$. The origin of $\{h\}$ is defined at the position at which the user holds the handle, and the x, y, z axes are aligned with the end-effector of the robot.

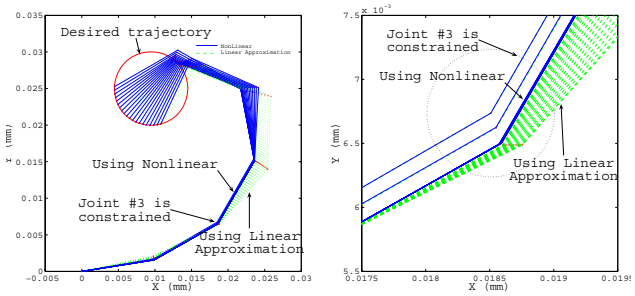
V. Results and Discussion

Our first test was a comparison of the computation time for solving the least squares problem with linear constraints and a nonlinear program. The results were obtained on a Pentium IV 2GHz machine, with 512MB of memory and are shown in Table III. The basic control loop as described in steps 1-4 in section III-A has a periodicity of $30ms$. The user performs the task of moving from one end to another of the curve and the time for computation of step 3 is recorded. Table III shows the average time for computation over the samples taken for the duration of complete experiment. The experiment is repeated using a linear solver with different number of hyperplanes as well as non-linear solver. For the linearly constrained problem, the time for computation increases with the number of hyperplanes used to represent the constraints. The nonlinear solver takes about $9ms$. Though it solves 5 to 50 iterations of a quadratic program in (6), the number of constraints per subprogram is usually small because an "active" set is used. For this work we used the distance function as the constraint, which can also be readily approximated by a convex polyhedra to give a linearly constrained problem. This might not be the case for general constraints, in which case the nonlinear solver might be the only solution.

TABLE III
Time for computation

# Hyperplanes	4	8	16	32	Nonlinear
Time(ms)	2.2680	4.1225	7.2842	14.3549	9.4017

Figure 5 shows the result of simulations using a planar 6 link manipulator as a test case. Since the linear approximation circumscribes the actual constraints, this leads to larger motion of the joints.

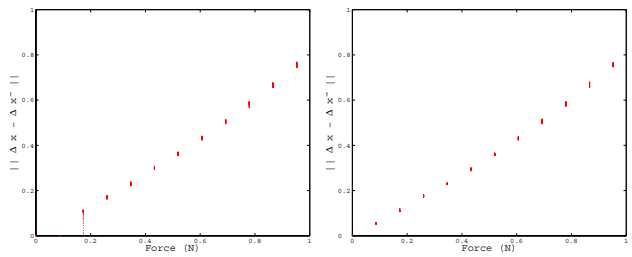


(a) (b)

Fig. 5. (a) A example trajectory of a planar 6 link robot with joint #3 constrained to a point. (b) Close-up of joint #3 showing difference between linear approximation and nonlinear constraints

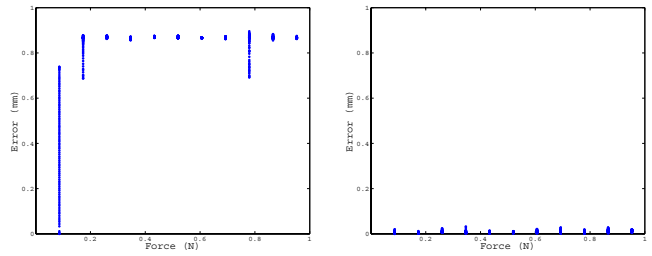
Figure 6 shows the part of the objective function given by $||(\Delta x - \Delta x^d)||_2^2$. This objective represents how closely the handle moves with respect to the force applied by the user. A larger value creates a perception of resistance to the user. A gradually increasing force in the direction normal to the nominal path was applied. The deviations from the path are shown in Figure 7. As seen in these figures, the objective function for the "soft" constraint does not increase much until a threshold of error is reached, meaning that the robot handle follows the user command. In the case of "hard" constraints, this value increases with the slightest error in position.

Figure 8 shows the plots of positions of the tip and the entry port for both the "soft" as well as "hard" constraints. The trajectories of the tool tip, the user handle and the point on the tool closest to the entry point are shown.



(a) (b)

Fig. 6. The value of objective with respect to force applied normal to constraint (a) "soft", where $w_{s,i} = 1 \times 10^{-3}$ (b) "hard", where $w_{s,i} = 1 \times 10^{+3}$



(a) (b)

Fig. 7. The value of error with respect to force applied normal to constraint (a) "soft", where $w_{s,i} = 1 \times 10^{-3}$ (b) "hard", where $w_{s,i} = 1 \times 10^{+3}$

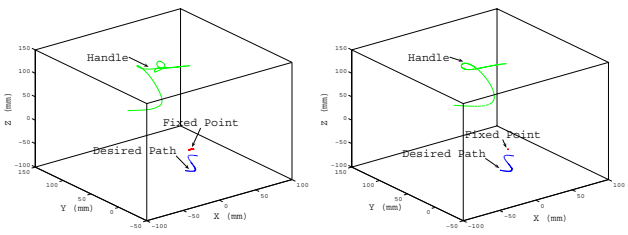
VI. Conclusion

We presented a weighted constrained optimization framework to formalize a library of "virtual fixtures" for surgical robot assistants. The virtual fixtures can be customized for a particular surgical task by combining one or more objectives and constraints assigned to single or multiple task frames. We implement the constrained optimization problem with both linear and nonlinear constraints, and discuss the trade-offs between linear approximations and nonlinear constraints.

The constraints for the task primitives are often nonlinear; nevertheless we can use linear expressions to approximate the constraints. Solving linearly constrained least squares problems can take less computation time. However, when linear constraints are used to approximate nonlinear constraints, more linear constraints are required for a tighter approximation. The higher the number of constraints used for approximation, the larger the computation time required to solve the problem. There is a trade off between accuracy and speed between linear and nonlinear constraints. One can choose to use a linear approximation with fewer number of hyperplanes if accuracy is not a concern, whereas nonlinear gives better accuracy.

We also introduced a "soft" virtual fixture mechanism for robotic surgical assistance. The "soft" virtual fixtures enable a surgical tool to have some freedom inside safety regions. This mechanism gives surgeons control over some degree of uncertainty that arises from factors such as registration errors, variations in anatomy and changes during procedures, allowing them to move the tool away from the computed path.

In the future, we wish to explore the possibility of using other nonlinear constraints and to apply this technique to other robots having higher redundancy and dexterity and to use a piecewise linear or nonlinear objective for the slack variables.



(a) (b)

Fig. 8. Trajectories of points on robot (a) "soft", where $w_{s,i} = 1 \times 10^{-3}$ (b) "hard", where $w_{s,i} = 1 \times 10^{+3}$

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. B. Rosenberg, "Virtual fixtures: Perceptual tools for telerobotic manipulation," in *IEEE Virtual Reality Annual International Symposium*, Seattle, USA, 1993, pp. 76–82.

[2] 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," *Proc. Inst. Mech. Eng. H*, vol. 211, no. 4, pp. 285–292, 1997.

[3] S. Park, R. D. Howe, and D. F. Torchiana, "Virtual fixtures for robotic cardiac surgery," *Proc. Intl. Conf. MICCAI*, pp. 1419–1420, 2001.

[4] A. Bettini, P. Marayong, S. Lang, A. M. Okamura, and G. D. Hager, "Vision assisted control for manipulation using virtual fixtures," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 953–966, 2004.

[5] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulator," *Intl. J. Robot. Res.*, vol. 6, no. 2, pp. 3–15, 1987.

[6] C. Klein and B. E. Blaho, "Dexterity measures for the design and control of kinematically redundant manipulators," *Intl. J. Robot. Res.*, vol. 2, no. 6, pp. 72–83, 1987.

[7] J. M. Hollerbach and K. C. Suh, "Redundancy resolution of manipulators through torque optimization," *IEEE Trans. Robot. Automat.*, vol. 3, no. 4, pp. 308–316, 1987.

[8] R. J. Spiteri, D. K. Pai, and U. M. Ascher, "Programming and control of robots by means of differential algebraic inequalities," *Intl. J. Robot. Res.*, vol. 16, no. 2, pp. 135–145, 2000.

[9] J. Funda, R. H. Taylor, B. Eldridge, S. Gomory, and K. G. Gruben, "Constrained cartesian motion control for teleoperated surgical robots," *IEEE Trans. Robot. Automat.*, vol. 12, no. 3, pp. 453–465, 1996.

[10] 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.*, vol. 14, pp. 279–287, 1995.

[11] 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," *Intl. J. Robot. Res.*, vol. 18, no. 12, pp. 1201–1210, 1999.

[12] G. Guthart and K. Salisbury, "The intuitive TM telesurgery system: Overview and application," in *IEEE International Conference on Robotics and Automation*, San Francisco, USA, 2000, pp. 618–621.

[13] M. Cavasoglu, F. Tendick, M. Cohn, and S. Sastry, "A laparoscopic telesurgical workstation," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 4, pp. 728–739, 1999.

[14] C. Lawson and R. Hanson, "Solving least squares problems," *Englewood Cliffs, NJ: Prentice-Hall*, 1974.

[15] P. Spellucci, "An SQP method for general nonlinear programs using only equality constrained subproblems," *Math. Prog.*, vol. 82, pp. 413–448, 1998.