Publisher's Statement: Human-Machine Collaborative Systems for Microsurgical Applications by D. Kragic, P. Marayong, M. Li, A.M. Okamura, and G.D. Hager

Abstract

The International Journal of Robotics Research, 24: 9,
(September 2005), p. 731. doi:10.1177/0278364905057059

Publisher's Statement

The Publisher wishes to include a reference to Dr. Rajesh Kumar's Ph.D. thesis, entitled "An Augmented Steady Hand System for Precise Micromanipulation," Johns Hopkins University, Baltimore MD (April 2001), which describes a framework for task level control on the Steady Hand Robot at JHU and which reports demonstrations of several representative tasks, including retinal cannulation, on dry lab and ex-vivo phantoms. The Publisher also refers to Figure 5 of the Article and wishes to reference the depiction of retinal cannulation in Dr. Kumar's thesis, which reflects a task sequence that was performed for Dr. Kumar's thesis and is depicted in the thesis as Figure 5.13.

Human–Machine Collaborative Systems for Microsurgical Applications

D. Kragic

Centre for Autonomous Systems
Stockholm, Sweden
danik@nada.kth.se

P. Marayong
M. Li
A. M. Okamura
G. D. Hager

Engineering Research Center for Computer Integrated
Surgical Systems and Technology
The Johns Hopkins University
Baltimore, MD 21218, USA

Abstract

Human–machine collaborative systems (HMCSs) are systems that amplify or assist human capabilities during the performance of tasks that require both human judgment and robotic precision. We examine the design and performance of HMCSs in the context of microsurgical procedures such as vitreo-retinal eye surgery. Three specific problems considered are: (1) development of systems tools for describing and implementing HMCSs, (2) segmentation of complex tasks into logical components given sensor traces of human task execution, and (3) measurement and evaluation of HMCS performance. These components can be integrated into a complete workstation with the ability to automatically "parse" traces of user activities into task models, which are loaded into an execution environment to provide the user with assistance using on-line recognition of task states. The major contributions of this work include an XML task graph modeling framework and execution engine, an algorithm for real-time segmentation of user actions using continuous hidden Markov models, and validation techniques for analyzing the performance of HMCSs.

KEY WORDS—human-machine cooperation, hidden Markov models, virtual fixtures, microsurgical procedures, XSD/XML

1. Introduction

The goal of the human–machine collaborative system (HMCS) project is to investigate human–machine cooperative execution of small-scale, tool-based manipulation activities. The motivation for collaborative systems is based on evidence suggesting that humans operating in collaboration with robotic mechanisms can take advantage of robotic speed and precision, but avoid the difficulties of full autonomy by retaining the human "in the loop" for essential decision making and/or physical guidance (Kumar et al. 1999; Taylor et al. 1999). Our work on HMCSs is specifically aimed at microsurgery (Kumar et al. 1999; Rothbaum et al. 2002), but the basic principles apply to many other fine-scale tasks such as opto-electronic assembly, assembling for microelectromechanical systems (MEMS; Feddema and Christenson 1999), and cell manipulation (Kapoor, Kumar, and Taylor 2003; Kumar, Kapoor, and Taylor 2003).

Our approach to HMCSs focuses on three inter-related problems. (1) **Synthesis**: developing systems tools necessary for describing and implementing HMCSs. (2) **Modeling**: given sensor traces of a human performing a task, segmenting those traces into logical task components and/or measuring the compatibility of a given HMCS structure to that sequence of components. (3) **Validation**: measuring and evaluating HMCS performance.

Figure 1 depicts the high-level structure of our HMCS framework, which includes three main components: the human, the augmentation system, and an observer. We assume that the user primarily manipulates the environment using the augmentation system, although unaided manipulation may also take place in some settings (shown by the dashed line). The user is able to visually observe the tool and surrounding environment, and directs an augmentation device using intuitive force and position commands. The system may also have access to endpoint force data, targeted visual data and other application-dependent sensors, e.g. intra-operative imaging. The role of the observer is to assess available sensor data

The International Journal of Robotics Research
Vol. 24, No. 9, September 2005, pp. 731-741,
DOI: 10.1177/0278364905057059

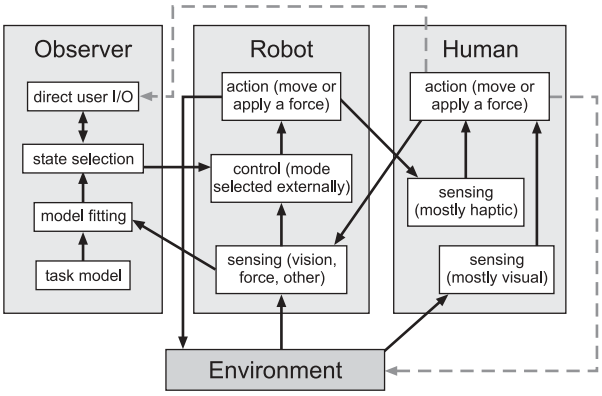

Fig. 1. Structure of a HMCS.

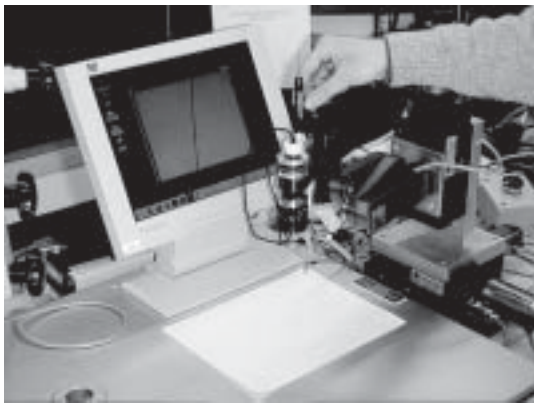

Fig. 2. HMCS experimental setup using the Johns Hopkins University Steady Hand Robot.

(including forces applied by the user) and initiate, modify, or terminate various forms of assistance. Optional direct interaction between the observer and the user may also be used to convey information or otherwise synchronize their behavior.

The basic notion of HMCSs is clearly related to traditional teleoperation, although the goal of HMCSs is not to "remotize" the operator (Hirzinger et al. 1991) but rather to provide appropriate levels of operator assistance depending on context. At one extreme, shared control (Guo, Tarn, and Bejczy 1995) can be viewed as an HMCS for manipulation tasks in which some degrees of freedom are controlled by machine and others by the human. At the other extreme, supervisory control (Sheridan 1986) gives a more discrete, high-level notion of human–machine interaction. Our notion of HMCSs essentially incorporates both views, combining them with broader questions of modeling manipulation activities consisting of multiple steps and varying level of assistance, and validating those models against human performance data.

Fig. 3. Cannulation of a central retinal vein. The retinal vasculature is readily apparent; the cannula is a pulled glass pipette visible in the upper portion of the image.

2. Synthesis

Our current system development efforts are motivated by the domain of retinal microsurgery, where tasks are performed near the limit of the manipulation capabilities of the human hand–eye system. A particularly challenging procedure is retinal vein cannulation: the introduction of a micro-pipette into a vessel of approximately 100 μm in diameter (Weiss 2001; Figure 3). We augment the surgeon's physical abilities using the Johns Hopkins University Steady Hand Robot (JHU SHR; Figure 2). Briefly, the JHU SHR is an admittance-controlled seven-degree-of-freedom (7-DOF) robot equipped with a force/torque sensor at the end-effector. Tools are mounted at the end-effector and "manipulated" by an operator holding a handle attached to the force sensor. The robot moves in proportion to the applied force, implementing a means of direct control for the operator. The robot is ergonomically appropriate for minimally invasive microsurgical tasks and provides micrometer-scale accuracy (Taylor et al. 1999). Two general forms of motion assistance have been investigated with the JHU SHR: force scaling (Roy and Whitcomb 2002) and motion guidance using virtual fixtures (Marayong and Okamura 2004). In this paper, we report on the latter.

2.1. Describing Spatial Motion Constraints

"Virtual fixtures" in our implementation provide cooperative control of the manipulator by "stiffening" a hand-held guidance mechanism against certain directions of motion or forbidden regions of the workspace. Studies on virtual fixtures for teleoperation have indicated that user performance can increase as much as 70% with fixture-based guidance (Rosenberg 1994). Here, we describe our framework for implementing virtual fixtures on the JHU SHR. In what follows, we

model the robot as a purely kinematic Cartesian device with tool tip position $\mathbf{x} \in SE(3)$ and a control input that is endpoint velocity $\mathbf{v} \in \mathcal{R}^6$, all expressed in the robot base frame. A human operator guides the robot by applying forces and torques $\mathbf{f} \in \mathcal{R}^6$ on the manipulator handle, likewise expressed in robot base coordinates.

We define SHR guidance geometrically by identifying a space of "preferred" directions of motion. Let us assume that we are given a $6 \times n$ time-varying matrix $D = D(t)$, $0 < n < 6$, representing the instantaneous preferred directions of motion. For example, if n is 1, the preferred direction is along a curve in SE(3); if n is 2, the preferred directions span a surface, and so forth.

We define two projection operators, the span and the kernel of the column space, as

$$\text{Span}(D) \equiv [D] = D(D'D)^{+}D' \quad (1)$$

$$\text{Ker}(D) \equiv \langle D \rangle = I - [D] \quad (2)$$

where $'$ denotes matrix transpose and $^{+}$ denotes pseudo-inverse for the case where D is (column) rank deficient. By decomposing the input force vector, $\mathbf{f}$, into two components, $\mathbf{f}_D \equiv [D]\mathbf{f}$ and $\mathbf{f}_\tau \equiv \mathbf{f} - \mathbf{f}_D = \langle D \rangle \mathbf{f}$, and introducing a new admittance ratio $k_\tau \in [0, 1]$ that attenuates the non-preferred component of the force input, we arrive at an admittance control

$$\mathbf{v} = k(\mathbf{f}_D + k_\tau \mathbf{f}_\tau) = k([D] + k_\tau \langle D \rangle)\mathbf{f}. \quad (3)$$

Thus, the final control law is in the general form of an admittance control with a time-varying gain matrix determined by $D(t)$. By choosing k, we control the overall admittance of the system. Choosing k_τ low imposes the additional constraint that the robot is stiffer in the non-preferred directions of motion. We refer to the case of $k_\tau = 0$ as a hard virtual fixture, since it is not possible to move in any direction other than the preferred direction. All other cases will be referred to as soft virtual fixtures. In the case $k_\tau = 1$, we have an isotropic admittance.

Two distinct types of guidance are possible with this basic framework.

1. Motion in a subspace: suppose we are supplied with a time-varying, continuous function $D = D(t)$. Applying eq. (3) yields a motion constraint within that subspace.

2. Motion to a target pose $\mathbf{x}_t \in SE(3)$: suppose that we have a vector $\mathbf{u} = f(\mathbf{x}, \mathbf{x}_t)$ that indicates the specific direction to the target. Then by choosing $D = \mathbf{u}$ and applying eq. (3), we create a virtual fixture that guides the user to the given target pose.

Although they use the same underlying formulation, the results of these two methods are quite different. In particular, note that when applying eq. (3) in the first case, there is no notion of a fixed reference as there is in the second case. For example, suppose the goal is to move the tip in a specific plane in space. If D contains the two vectors that span the planar surface, the position of the tool tip prefers to move parallel to that surface, but will not return to it if moved away from the surface.

In order to define motions in specific surfaces, we need to include a notion of feedback. Consider D as our preferred directions and let $\mathbf{u} = f(\mathbf{x}, S)$ now denote a signed distance of the tool tip to the reference surface S. We define a new preferred direction as

$$D_c(\mathbf{x}) = (1 - k_d)[\mathbf{D}]\mathbf{f}/\|\mathbf{f}\| + k_d \langle \mathbf{D} \rangle \mathbf{u} \quad 0 < k_d < 1, \quad (4)$$

combining the vectors that encode motion in the preferred directions and the vector correcting the tool tip back to S. The constant k_d governs how quickly the tool is moved toward the reference surface. We note that the division by $\|\mathbf{f}\|$ is undefined when no user force is present. As projection is invariant to scale, we write eq. (4) as

$$D_c(\mathbf{x}) = (1 - k_d)[\mathbf{D}]\mathbf{f} + k_d \|\mathbf{f}\| \langle \mathbf{D} \rangle \mathbf{u} \quad 0 < k_d < 1 \quad (5)$$

and apply eq. (3) with $D = D_c$.

One potential problem with this control law is that when the user applies no force, there is no virtual fixture because there is no defined preferred direction. Thus, there is a discontinuity at the origin. However, in practice the resolution of any force sensing device is usually well below the numerical resolution of the underlying computational hardware computing the pseudo-inverse, so the user will never experience this discontinuity.

To illustrate the difference between open-loop and closed-loop virtual fixtures, we applied both using the remote center of motion (RCM) module of the JHU SHR, which rotates the end-effector of the robot about a fixed point in the workspace. The task is to rotate the tool tip about the robot z-axis at a fixed angle. This creates a cone-shaped motion with the tip located at the RCM point. In both the open-loop and closed-loop cases a hard virtual fixture is used. It is evident from the plots in Figure 4 that, without any feedback, the position of the tool tip gradually spirals out from the reference path. At the same time, adding a small amount of feedback (in this case based on robot kinematic information) eliminates this error, just as it would in a traditional feedback control system.

For our microsurgical applications, virtual fixtures are based on both purely kinematic information and on external information provided by vision. We have implemented both two-dimensional (planar) and three-dimensional versions of the system at both macroscopic and microscopic scales. The remainder of this paper focuses on results obtained using the two-dimensional implementation shown in Figure 2. A CCD camera is attached to the robot and views a curve on a task plane. The images are processed using the XVision tracking system (Hager and Toyama 1998) to determine the tangent

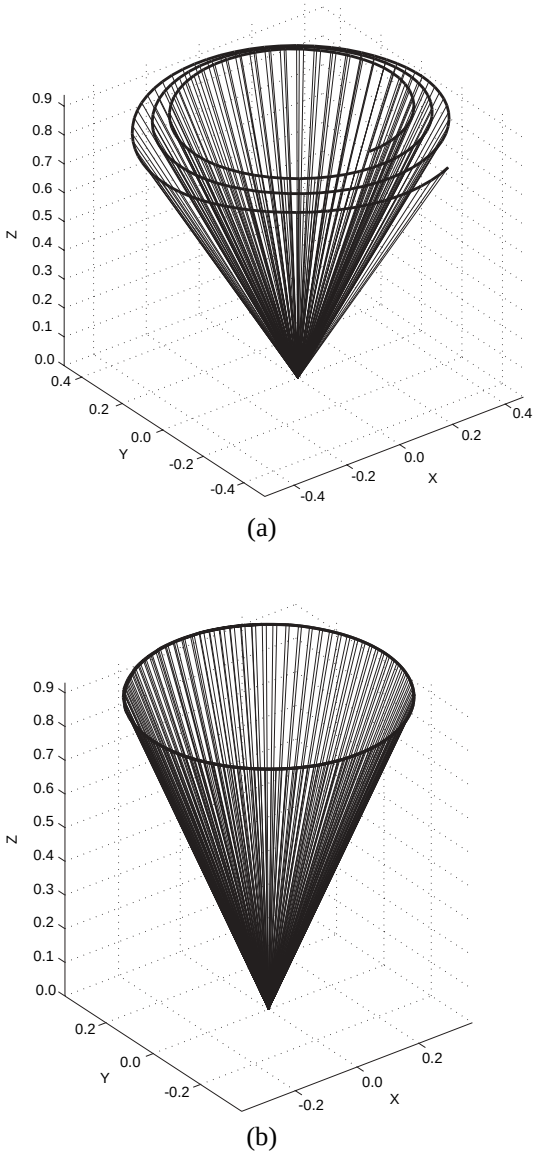

Fig. 4. Position of robot tool tip with open-loop virtual fixtures (a) and closed-loop virtual fixtures (b).

to the reference path at a point closest to the center of the image in real time. The center is marked with a cross, and a user attempts to move the robot to follow the curve (similar to tracking a blood vessel in preparation for retinal cannulation). In this case, if t_x and t_y are the components of the tangent to the curve in the image plane, and n_x and n_y are the vectors from the center to the closest point on the curve, then we have $u = (n_x, n_y, 0, 0, 0, 0)'$ and $D = (t_x, t_y, 0, 0, 0, 0)'$. We note that this approach has been explored previously in the visual servoing context by other authors (Maniere, Couvignou, and Khosla 1993). Extensions of this system can be found in Bettini et al. (2002, 2004) and Marayong et al. (2003). Experimental results with human users are presented in Section 4.

2.2. High-level Task Specification

Most microsurgical tasks consist of discrete, serial, quasi-static steps, each with a clearly defined outcome. Hence, models for such procedures can be defined by relatively simple graphs. For example, retinal vein cannulation involves positioning and orienting a needle to the vicinity of the vein, and inserting it when appropriate until contact is made. Upon contact, puncturing is performed, after which the needle can be safely withdrawn, as shown in Figure 5.

In order to provide appropriate assistance, it is necessary to create models of procedures and attach appropriate types or levels of assistance to each step in the task. The current system is composed of the three levels shown in Figure 7.

(i) Task graph modeling and generation allows the user to design and generate a graph for the task he/she wants to perform. For this purpose, a graphical user interface (Figure 6) has been designed. The user can easily generate the control task without any need for writing the code.

(ii) Task graph execution: given the task graph, the system automatically initiates all the basic processes required to execute the task.

(iii) A set of basic primitives is implemented and used during the execution of the first two levels. These primitives are those commonly required for surgical robot tasks. An example of a set of basic primitives is shown in Figure 5 where a vein cannulation task is considered. Our previous work provides a detailed presentation of all the primitives currently available in the system (Kragic and Hager 2003).

The instructions to the low-level motion controller of the robot are passed from the basic primitives. The primitives have a common interface with functions such as `Start()`, `Run()`, and `Stop()`. We assume that the task can be modeled by a set of events, basis vectors, and a procedure with a sequence of states. Events represent links or triggers between the states and are either sensory based (e.g. contact detected) or induced by the user (e.g. button pressed or predefined pose). Basis vectors span the task space of the robot. Using different types of operators on the basis vectors, we can easily define subspaces for preferred robot motion. States are represented by a set of transitions and constraints. Transitions are pairs `(event, newState)`; for each event there is a `newState` defined.

A set of basic primitives representing individual states can be combined to design complex tasks. Here, an important feature of the system is the ability to impose some basic requirements on the design of a task. For this reason, we have chosen to use an XML (Extensible Markup Language, http://www.xml.org) Schema Definition Language (XSD) representation for a general class of tasks as follows.

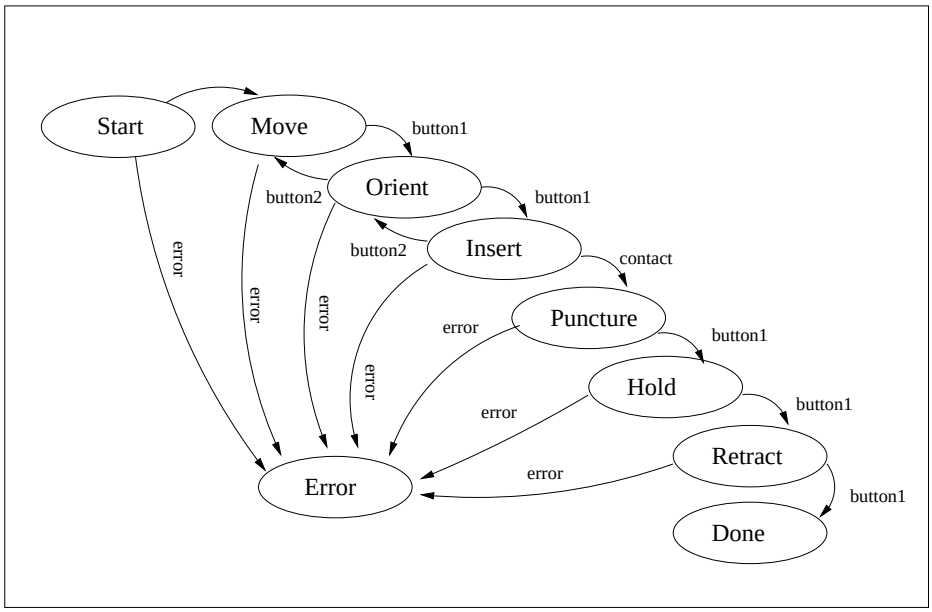

Fig. 5. A graph example for vein cannulation.

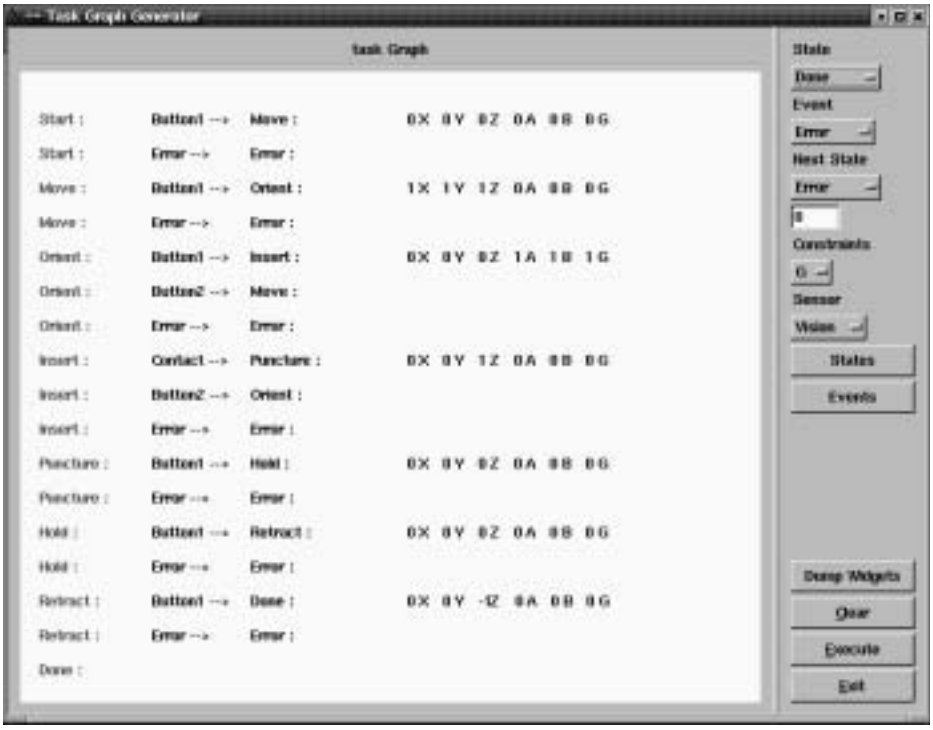

Fig. 6. Graphical user interface for generating task graphs.

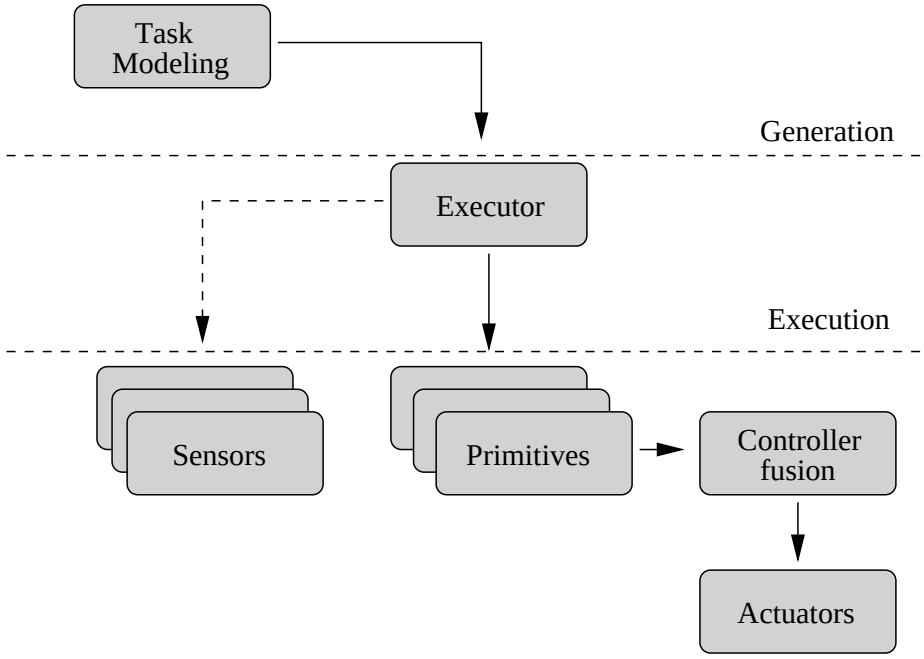

Fig. 7. The HMCS architecture has three levels: modeling/generation, execution, and basic primitives.

```
<xs:element name="procedures">
    <xs:complexType><xs:sequence>
        <xs:element ref="event" minOccurs="1"
                    maxOccurs="unbounded"/>
       <xs:element ref="basisVector" minOccurs="0"
                       maxOccurs="6"/>
        <xs:element ref="procedure" minOccurs="1"
                    maxOccurs="unbounded"/>
-------------------------------------------------
 <xs:element name="procedure">
    <xs:complexType> <xs:sequence>
        <xs:element name="description"
                    type="xs:string"
            minOccurs="0" maxOccurs="1"/>
        <xs:element ref="state" minOccurs="1"
            maxOccurs="unbounded"/>
      <xs:attribute name="name" type="xs:string"
            use="required"/>
-------------------------------------------------
<xs:element name="state">
    <xs:complexType><xs:sequence>
        <xs:element name="description"
                    type="xs:string"
                minOccurs="0" maxOccurs="1"/>
        <xs:element ref="transition"
                minOccurs="1"
                maxOccurs="unbounded"/>
        <xs:element ref="constraint"
                minOccurs="0"
                maxOccurs="unbounded"/>
      <xs:attribute name="name"
                    type="xs:string"
            use="required"/>
-------------------------------------------------
<xs:element name="event">
    <xs:complexType><xs:sequence>
        <xs:element name="description"
                    type="xs:string"
                minOccurs="0" maxOccurs="1"/>
      <xs:attribute name="type"  type="xs:ID"
            use="required"/>
-------------------------------------------------
  <xs:element name="constraint">
    <xs:complexType> <xs:sequence>
        <xs:element name="description"
                    type="xs:string"
            minOccurs="0" maxOccurs="1"/>
      <xs:attribute name="constr" type="xs:IDREF"
      use="required"/>
      <xs:attribute name="weight"
                    type="xs:double"
      use="required"/>
```

There may be a number of constraints defined for each state. These constraints are directly connected with the behavior of the system through a control algorithm. Constraints may include the level of programmed compliance/stiffness of the robot, the geometric definition of a virtual fixture, etc. According to this specification schema, XML (http://www.xml.org) can now be used for task graph generation.

A task graph modeling system allows a user to interactively design a graph for the task to be performed and save the graph as an XML file. A task graph execution engine reads the XML description and coordinates the execution of the graph. The graph itself references a library of basic primitives for providing user assistance at each step in the task, and a set of transition conditions. Thus, at each state, the graph executor initiates the appropriate assistance mode and monitors for transition events to subsequent states. Currently, there are three ways of generating a task graph: (1) using the graphical interface, (2) directly writing an XML file, and (3) automating off-line task modeling. We now discuss the last of these methods in detail.

3. Modeling and Sensing Human Intent

One problem that arises in HMCSs is that of providing appropriate assistance to a user based on the intent or context of his or her actions. In order to do so, the system must have a model of the task being performed, and a means for relating the actions of the user to that model. We have investigated continuous hidden Markov models (HMMs) as a means of modeling tasks from sensor traces of user task execution. A good overview of HMM theory and application is provided in Rabiner (1989). We have also developed a new algorithm for HMM recognition that allows for real-time segmentation of user actions. This recognition method allows an HMCS to automatically transition between primitives based on natural user behavior, rather than button clicks or other "artificial" inputs.

3.1. Off-line Task Modeling

The basic hypothesis of off-line task modeling is that, for restricted domains such as microsurgery, it is possible to model tasks using a small vocabulary of primitive "gestemes" with an associated assistance mode. We have tested our hypothesis using data acquired from task execution with the JHU SHR and modeled tasks using the HMM Toolkit (HTK; Hundtofte, Hager, and Okamura 2002; see http://htk.eng.cam.ac.uk).

The input data acquired from the robot consisted of seven variables: Cartesian forces and torques expressed in the robot's end-effector coordinates, and the magnitude of translation between the previous and current position readings. We assign a five-state linearly sequential left-to-right (SLR) structure to the HMMs of several gestemes. Two tasks were investigated: (1) a peg-in-hole task and (2) a paint task, which are analogous to retinal vessel cannulation and retinal membrane peeling, respectively. For the peg-in-hole task, five gestemes were used: *place*, *position*, *insert*, *withdraw* and *remove*. For the paint task, the four gestemes were: *place*, *position*, *paint*, and *remove*. Five users participated in the experiment and each user performed ten runs. Each gesteme was trained independently on a training set, then gestemes were run in parallel to segment a test set (Figure 8). In this application, training of individual models is done without any knowledge of the global structure of the network.

The obtained segmentation accuracy of the system over all users was around 84% for the peg-in-hole task and 90% for the paint task. In addition, we found that we could use gestemes trained for the paint task in the peg-in-hole task, and the resulting decrease in recognition performance was only 1.08%. These results, although preliminary, suggest that for suitably constrained situations, it is plausible that complex tasks can be modeled using a small set of manipulation primitives.

3.2. On-line Recognition and Assistance

Given a task model, we now ask whether it is possible to determine the appropriate task state on-line. To do this, we

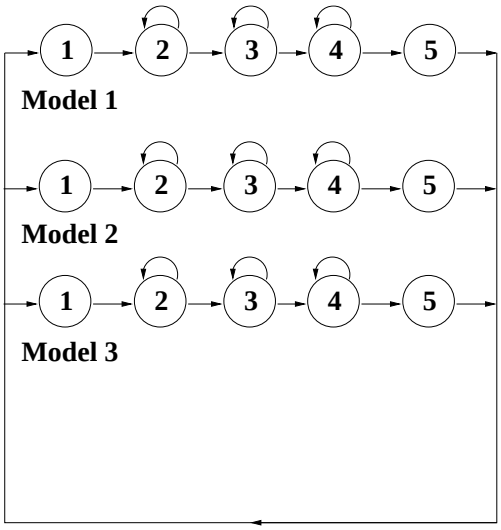

Fig. 8. Network for real-time, continuous HMM recognition. One model describes the user motion at any given time.

have developed an on-line continuous HMM recognition algorithm (Li and Okamura 2003). As before, the structure of each HMM is five-state SLR and each model operates in parallel. In our implementation, the input to the algorithm is an observation vector of user-applied forces, and the output is the highest likelihood HMM. Suppose we have M potential models and each model has N states. For a given model λ, let $\phi_{j,\lambda}(t)$ be the likelihood of observing vector o_1 to o_t and being in state j at time t. The partial likelihood of each model can be computed as

$$\phi_{j,\lambda}(t) = \left[\sum_{i=1}^{N} \phi_{i,\lambda}(t-1)a_{ij}\right] b_j(o_t) \tag{6}$$

with

$$\phi_{1,\lambda}(1) = 1 \text{ and } \phi_{j,\lambda}(1) = 0, \forall\ 1 < j \leq N \tag{7}$$

where a_{ij} is the probability of the transition from state i to state j, and $b_j(o_t)$ is the probability of the observation vector o_t in state j.

The likelihood of state 1 of a model λ in time $t > 1$ can be defined as

$$\phi_{1,\lambda}(t) = \frac{1}{M}\sum_{m=1}^{M} \phi_{N,m}(t-1) \quad \text{for} \quad 1 < \lambda \leq M. \tag{8}$$

For each model, we define the sum of the likelihood of all

states as the total likelihood at time t as

$$L(o_1, o_2, ..., o_t|\lambda) = \sum_{j=1}^{N} \phi_{j,\lambda}(t), \quad (9)$$

where the model with the highest total likelihood represents the current hypothesis.

We tested the on-line recognition algorithm with the JHU SHR in a planar environment, where the goal of the human operator was to follow a sinusoidal path, but avoid a portion of the path covered by a circle. Virtual fixture assistance for path following was provided or removed based on the results of HMM recognition, which was computed every 33 ms. In our experiment, we included three models/gestemes (HMMs) of user actions: (1) *silence* (user is doing nothing), (2) *follow curve* (user is following the curve), and (3) *avoid curve* (user is not following the curve). The force applied by the user, f, was decomposed into the directions parallel to reference curve δ and the normal direction τ. $||f_\delta||$, $||f_\tau||$ and $||f \cdot \delta||$ were the elements of the observation vector used to train the HMMs.

The accuracy of recognition and the sensitivity of our algorithm to training in differing environments were examined by an experiment where (1) the same sine curve was used for both training and recognition, and (2) different sine curves were used for training and recognition. The average accuracy of real-time continuous recognition among all subjects was larger than 90% in both cases.

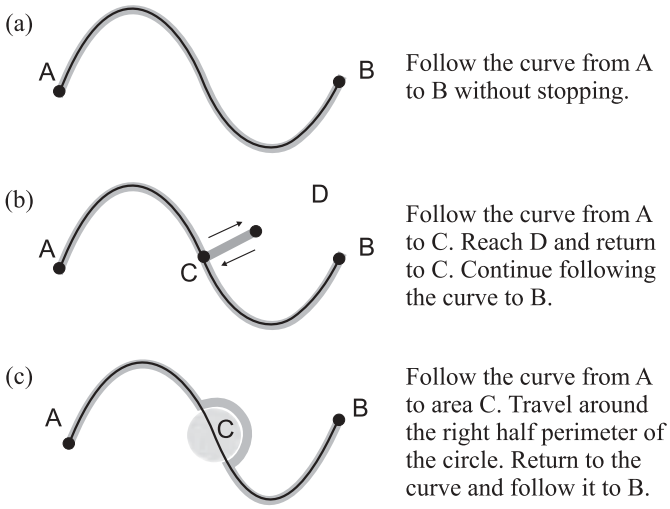

Fig. 9. Task descriptions for HMCS performance experiments: (a) path following, (b) off-path targeting, and (c) avoidance. The black line denotes the virtual fixture reference path, and the path to be followed by the user is dark gray.

4. Validation

During task execution, speed and accuracy are generally used as metrics to evaluate performance. There is typically a trade-off between these two metrics, as described by Fitts law (Fitts 1954). In this section, we demonstrate that the addition of virtual fixture guidance improves speed and accuracy in comparison to unaided SHR manipulation. The experimental setup makes use of the curve-following virtual fixture described in Section 2.1.

4.1. Virtual Fixtures

We present two experiments that examine the effects of virtual fixture guidance with admittance ratios ranging from complete guidance (τ = 0) to no guidance (τ = 1). Experiment I determines the relationship between level of virtual fixture guidance and performance of a path following task. Experiment II studies the effect of virtual fixture guidance on manipulation tasks in which the user must move away from the path defined by the virtual fixture. In both experiments, the reference path for virtual fixture guidance was a 0.39 mm thick sine curve with a 35 mm amplitude and 70 mm wavelength. For Experiment II, we added a circle of radius 10 mm with its center located at the midpoint of the sine curve. The subjects were provided with instructions as shown in Figure 9, and told to move along the path as quickly as possible without sacrificing accuracy, considering accuracy and speed with equal emphasis. Experiment I included five subjects performing the path following task (Figure 9(a)) three times with 11 admittance ratios from 0 to 1 (0, 0.1, . . .). Experiment II included eight subjects performing each of the three tasks shown in Figure 9 three times with four discrete admittance ratios corresponding to four guidance levels (0 = complete, 0.3 = medium, 0.6 = soft, and 1 = none). At the run time, time and error data were recorded during travel from Point A to Point B in Figure 9. The error represents the sum of deviation from the reference path. For the off-path targeting task (Figure 9(b)), we recorded the time it took each subject needed to return to the curve after leaving it. The error recorded in this task is the amount of overshoot/undershoot from the target. For the avoidance task (Figure 9(c)), we recorded the time needed to avoid the circle, and no error was measured. For the data obtained in Experiment II, we performed ANOVA and multiple pair-wise comparisons using Tukey's method to determine significant differences. Data are shown in Table 1.

For Experiment I, the data indicate that task error and execution time have linear relationships with admittance ratio, as shown in Figure 10. This is to be expected, since the output velocity of the SHR is linearly proportional to admittance ratio. For Experiment II, the average execution time and error were used to determine the improvement in performance with different guidance levels. For the path following task, a decrease

Table 1. Experiment II. Experimental Results for Eight Subjects: Rows 1–4 for Path Following, Rows 5–7 for Off-Path Targeting, and Rows 8–10 for Avoidance

			Time (s)		Error (mm)	
	Admittance Ratio (k_τ)	Guidance	Average	Standard Deviation	Average	Standard Deviation
1	0	Complete	18.710	1.357	0.061	0.013
2	0.3	Medium	19.647	1.879	0.078	0.020
3	0.6	Soft	20.425	2.042	0.121	0.030
4	1	None	24.745	5.405	0.229	0.087
5	0.3	Medium	9.754	3.664	1.256	0.456
6	0.6	Soft	7.148	3.178	0.910	0.180
7	1	None	6.256	2.353	0.702	0.437
8	0.3	Medium	14.681	5.629	–	–
9	0.6	Soft	11.871	4.959	–	–
10	1	None	9.317	2.954	–	–

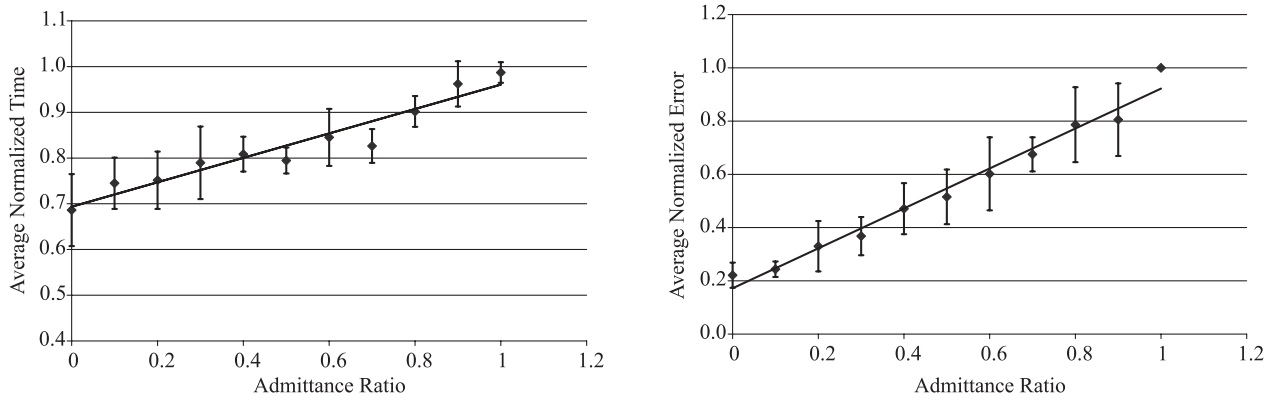

Fig. 10. Average normalized time and error versus admittance ratio for the path following task.

in admittance ratio significantly reduces error, except between medium and complete guidance. However, a decrease in admittance ratio does not improve execution time significantly. We note that none of the subjects performed worse in both time and error when admittance ratio decreased. Complete guidance resulted in the best performance, but any guidance at all resulted in shorter time and/or higher accuracy compared to no guidance. For the targeting task, stronger guidance slowed task execution. However, the execution times for no guidance and soft guidance do not differ significantly. The analysis also shows no significant difference in error for all guidance levels. In general, reducing guidance reduces the time and error during target acquisition. For the avoidance task, only the execution time was considered. The results indicate that less guidance reduces the execution time.

The results show that the selection of virtual fixture guidance level is a task-dependent process. For surgical applications, error reduction is typically more important than the amount of time required to perform the task. For tasks that require significant independent movement from the user, such as object avoidance and off-path targeting, strong guidance will reduce accuracy and increase execution time. Therefore, a lower amount of guidance is recommended. Based on the linear relationship between admittance and performance found in Experiment I, we developed admittance ratio selection parameters that can be used to determine an appropriate guidance level (Marayong and Okamura 2004). Using equal weighting for error versus time, and path following versus path avoidance, we found that an admittance ratio of approximately 0.6 is optimal. However, on-line admittance tuning

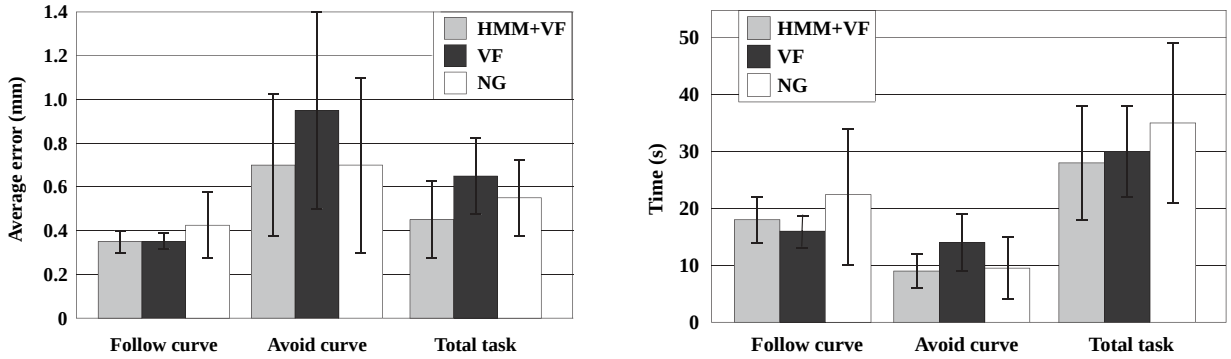

Fig. 11. Mean and standard deviation of average error (left) and time (right) for the *follow curve* section of the task, *avoid curve* section of the task, and *total* task. *HMM+VF* indicates that the virtual fixture was switched off by the HMM when it was detected that the user intended to move away from the curve. *VF* indicates a constant virtual fixture with admittance ratio $k_\tau = 0.3$, and *NG* indicates no guidance ($k_\tau = 1$).

based on user preference or automatic intent recognition is recommended to achieve the full benefit of virtual fixture guidance, as described in the following section.

4.2. On-line Assistance

In this experiment, eight subjects were asked to follow the sine curve (model *follow curve*) and the edge of the circle (model *avoid curve*) in three different modes: (1) *HMM+VF* where real-time recognition was used to determine when to apply or remove the virtual fixture, (2) *VF* where a virtual fixture with constant admittance ratio $k_\tau = 0.3$ was applied, and (3) *NG* where the virtual fixture was removed (no guidance, $k_\tau = 1$). The error (deviation from the curve) and time were again used to validate the performance. To create a "gold standard" for the correct segmentation, the subjects were asked to press a space bar when they intended to transition from one model to another. The results presented in Figure 11 show the improvements in performance when the combined algorithm (*HMM+VF*) is used. This is expected, as the *HMM+VF* algorithm provides the user with assistance when he/she wants to follow the curve, and removes the assistance otherwise.

5. Conclusions and Future Work

We have presented the structure, analytical models and experimental results of a HMCS. This includes a method for implementing guidance virtual fixtures with an admittance controlled robot, as well as techniques for automatic recognition of user activities. These components are integrated through a task graph and execution engine specifically designed for serial procedures such as those encountered in microsurgery.

These components are being assembled into a complete microsurgical workstation for vitreo-retinal eye surgery (see Figure 12). For various microsurgical procedures, the system will automatically "parse" traces of user execution into a task model, load the task model into the execution environment, and provide the user with assistance using on-line recognition of task state. The system will be tested using *ex vivo* and *in vivo* experiments involving evaluation of system performance during operation by surgeons.

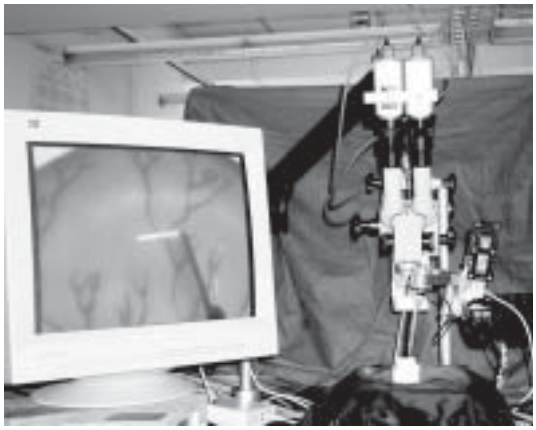

Fig. 12. The eye microsurgery testbed. The system includes the JHU SHR for tool manipulation, a stereo video microscope, and a stereoscopic display system. The methods described in this paper are being tested in simulated surgical environments using this system.

Acknowledgments

The authors thank Russell Taylor and Blake Hannaford for their insight related to this research. This material is based upon work supported by the National Science Foundation,

under Grants No. EEC-9731478, IIS-0099770, ITR-0205318, and the Swedish Foundation for Strategic Research through the Centre for Autonomous Systems.

References

Bettini, A., Lang, S., Okamura, A., and Hager, G. 2002. Vision assisted control for manipulation using virtual fixtures: experiments at macro and micro scales. *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, DC, May 11–15, pp. 3354–3361.

Bettini, A., Marayong, P., Lang, S., Okamura, A. M., and Hager, G. D. 2004. Vision assisted control for manipulation using virtual fixtures. *IEEE International Transactions on Robotics and Automation* 20(6):953–966.

Feddema, J. T. and Christenson, T. R. 1999. Parallel assembly of high aspect ratio microstructures. *SPIE Microrobotics and Microassembly* 3834:153–164.

Fitts, P. M. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology* 47:381–391.

Guo, C., Tarn, T. J., and Bejczy, A. 1995. Fusion of human and machine intelligence for telerobotic systems. *Proceedings of the International Conference on Robotics and Automation*, Nagoya, Japan, pp. 3110–3115.

Hager, G. D. and Toyama, K. 1998. The XVision system: a general purpose substrate for real-time vision applications. *Computer Vision and Image Understanding* 69(1):21–27.

Hirzinger, G., Grunwald, G., Brunner, B., and Heindl, H. 1991. A sensor-based telerobotic system for the space robot experiment. *Proceedings of the 2nd International Symposium on Experimental Robotics*, Toulouse, France, June 25–27, pp. 222–238.

Hundtofte, C. S., Hager, G. D., and Okamura, A. M. 2002. Building a task language for segmentation and recognition of user input to cooperative manipulation systems. *Proceedings of the 10th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Orlando, FL, March, pp. 225–230.

Kapoor, A., Kumar, R., and Taylor, R. 2003. Simple biomanipulation tasks with a "steady hand" cooperative manipulator. *Proceedings of the 6th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI 2003)*, Vol. I, Montreal, Canada, *Lecture Notes in Computer Science*, Vol. 2878, Springer-Verlag, Berlin, pp. 141–148.

Kragic, D. and Hager, G. D. 2003. Task modeling and specification for modular sensory based human-machine cooperative systems. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS03)*, Las Vegas, NV, October 27–31, pp. 3192–3197.

Kumar, R., Goradia, T. M., Barnes, A., Jensen, P., Whitcomb, L. L., Stoianovici, D., Auer, L. M., and Taylor, R. H. 1999. Performance of robotic augmentation in common dextrous surgical motions. *Medical Image Computing and Computer-Assisted Interventions (MICCAI 1999)*, Cambridge, UK, pp. 1108–1115.

Kumar, R., Kapoor, A., and Taylor, R. H. 2003. Preliminary experiments in robot/human microinjections. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, Las Vegas, NV, October 27–31, Vol. 4. pp. 3186–3191.

Li, M. and Okamura, A. M. 2003. Recognition of operator motions for real-time assistance using virtual fixtures. *Proceedings of the 11th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, IEEE Virtual Reality*, Los Angeles, CA, March, pp. 125–131.

Maniere, E. C., Couvignou, P., and Khosla, P. 1993. Robotic contour following based on visual servoing. *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 1993)*, Yokohama, Japan, July 26–30, pp. 716–722.

Marayong, P. and Okamura, A. M. 2004. Speed–accuracy characteristics of human–machine cooperative manipulation using virtual fixtures with variable admittance. *Human Factors* 46(3):518–532.

Marayong, P., Li, M., Okamura, A. M., and Hager, G. D. 2003. Spatial motion constraints: theory and demonstrations for robot guidance using virtual fixtures. *Proceedings of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, September 14–19, pp. 1954–1959.

Rabiner, L. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257–286.

Rosenberg, L. 1994. Virtual Fixtures. Ph.D. Thesis, Department of Mechanical Engineering, Stanford University.

Rothbaum, D., Roy, J., Stoianovics, D., Berkelman, P., Hager, G., Taylor, R., Whitcomb, L., Francis, H., and Niparko, J. 2002. Robot-assisted stapedotomy: micropick fenestration of the stapes footplate. *Otolaryngology—Head and Neck Surgery* 127(5):417–426.

Roy, J. and Whitcomb, L. L. 2002. Adaptive force control of position/velocity controlled robots: theory and experiment. *IEEE Transactions on Robotics and Automation* 18(2):121–137.

Sheridan, T. B. 1986. Human supervisory control of robot systems. *Proceedings of the International Conference on Robotics and Automation*, San Francisco, CA, pp. 808–812.

Taylor, R. H., Jensen, P. S., Whitcomb, L. L., Barnes, A. C., Kumar, R., Stoianovici, D., Gupta, P., Wang, Z., deJuan, E., and Kavoussi, L. 1999. A steady-hand robotic syatem for microsurgical augmentation. *International Journal of Robotics Research* 18:1201–1210.

Weiss, J. N. 2001. Injection of tissue plasminogen activator into a branch retinal vein in eyes with central retinal vein occlusion. *Opthalmology* 108(12):2249–2257.