Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea · May 21-26, 2001

Haptically Augmented Teleoperation

Nicolas Turro
Inria/Stanford
Nicolas.Turro@sophia.inria.fr

Oussama Khatib
Robotics group
Stanford University
ok@robotics.stanford.edu

Eve Coste-Maniere
CHIR Medical Robotics group
INRIA Sophia Antipolis
Eve.Coste-Maniere@sophia.inria.fr

Abstract

1 Introduction

Recent progresses in robotic-aided telesurgery help the surgeons to achieve comparable performances during laparoscopic operations and classical open sky procedures. In state of the art systems ([3]), the master console provides a lightweight mechanism whose ergonomy approaches real surgical tools; slave robots are becoming sufficiently dextrous to reproduce the surgeon's motion with great fidelity; and optical systems and video restitution provide a 3D feedback convincing enough to really immerse the surgeon inside the body of its patient. Thus, the main drawbacks of laparoscopy (limited field of view, awkward tools..) can be overcome.

Existing systems, such as the JPL's RAMS system ([1]), already propose tremor elimination and scaling of motions, which are considerably useful for applications like micro-surgery in cardiac operations or eye surgery. The purpose of this article is to go beyond the strict master/slave scheme and actually *enhance* the operator's capabilities during teleoperation. We focused our work on the implementation of three types of constraints for the operators movements:

- constrained movement (along a curve or on a predefined surface)
- virtual obstacle avoidance
- geometric constraints to limit the robots workspace

Constraints for the operator's movements can be implemented mechanically, as described in [7]. Our approach uses a haptic master robot, and consists in adding constraint forces to its control scheme. Constraint forces are computed according to attractive or repulsive potential fields placed around constraints, as defined in [4].

This article presents the principles of these control enhancements which, we believe, will raise dramatically the level of safety and precision that surgeon can achieve, but those principles can also be applied to a wide variety of teleoperation applications (eg. human friendly robotics, cooperative robots...). We also describe one implementation on an experimental platform composed by a Phantom haptic device for the master device and a PUMA arm as the slave manipulator.

2 Principles

In this section, we will first explain the core control scheme of the teleoperation, which reproduces movements of the master device on the slave robot, and also provides the operator with some feedback of the interaction between the robot and its environment. Then we will explain how constraints can be superposed to this control scheme.

2.1 Teleoperation - Slave robot control

In robotic-aided laparoscopy surgery, the master and slave system have vastly different geometry and dynamic properties. The master device is designed to be lightweight and ergonomic, whereas the slave robot must match the geometric constraints of the laparoscopy (operate thru fixed *'ports'* into the human body), and usually are much more massive and voluminous than the master device. Thus, the master device controls the position of the end effector of the slave using *operational space control*: the master device absolute position is used to control the position of the end effector of the slave robot. We compute the force F_s^* to apply to the end effector using a PD controller whose goal position is the master position x_m with a scaling factor S and an offset $x_0 = Sx_{s_0} - x_{m_0}$:

0-7803-6475-9/01/$10.00© 2001 IEEE

$$F_s^* = -K_p(x_s - Sx_m + x_0) - K_v(\dot{x}_s - \dot{x}_m). \quad (1)$$

Then, the torque τ_s to send to the motors is computed using the dynamic (Mass Matrix Λ and gravity g) of the slave robot. Computing the dynamics of the robot is critical in our approach : A good position and velocity servoing could be achieved without it but would involve high gains K_p and K_v, making the robot stiff. Using the dynamics allows us to achieve the same performences, but with lower gains, making the robot more compliant and sensitive to external forces.

$$\tau_s = J^t \Lambda F_s^* + g. \quad (2)$$

2.2 Teleoperation feedback - Master control

On the master device, we provide a force feedback F_m computed using the offset between the master arm position and the position of the slave robot end effector. This way, when external forces (from environment) are applied to the slave robot, those forces will be felt on the master device because the error on the slave tracking will increase. This method doesn't require any force sensor. However, there is constantly an error between the master and the slave position, mostly due to friction on the slave and also to its inertia. Since we do not want to feel a constant drag on the master in free-space movements, we use a cubic function for the computation of F_m:

$$F_m = -K(x_m - \frac{1}{S}(x_s + x_0))^3$$

the gain K is chosen small to exploit the cubic function near zero, on its flat part. So, small tracking errors will be minimized whereas real contact forces will be amplified. We assume that the system is naturally damped (because the manipulator is handling the master device). Furthermore, since our master device is very light, and well balanced, its mass can generally be neglected and its dynamic does not have to be computed. Thus, the torques to send to the master device are computed directly from F_m using the master's Jacobian matrix.

2.3 Teleoperation feedback - Constraints

The main added value of our work is to augment this basic teleoperation scheme with some constraints in order to increase the system safety and the users dexterity. All those constraints are based on *potential field*, and we define several possibilities to introduce them in the basic control scheme described above, depending on the nature of the constraint.

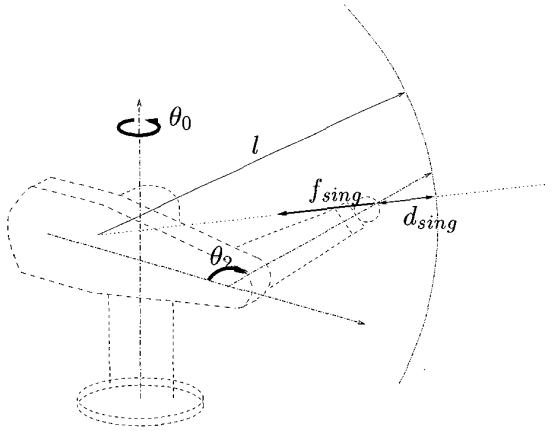

Figure 1: Singularity repulsive force

2.3.1 Slave-side constraints

Some constraints are critical for the safety of the slave robot, so it is preferable to plug them inside the *slave* controller, in case of a network or master device failure. For example, the master and the slave workspace are different most of the time: the joints limits and singularities are not the same. We propose to enforce those limitations using *repulsion fields*. Those fields will prevent the slave robot in given directions. Then, the basic control scheme described above will haptically render those slave constraints on the master device.

The repulsion fields can be either in the *joint space* or in the *operational space*. For example, our PUMA slave robot has a singularity on its second joint (elbow singularity) when the angle θ_2 (figure 1) reach the value π, or the end effector reaches a sphere whose radius is the length l of the arm fully extended.

For stability and safety reasons, we want to avoid this configuration, so we add to the slave control force F^* in equation (1) an *operational space repulsion force* f_{sing} whose direction is indicated on figure 1. The amplitude of $f_(sing)$ is proportional to the distance $\frac{1}{d_{sing}^3}$ between the end effector and a sphere whose radius is the lenght l of the maximal extension of the robot (corresponding to $\theta_2 = \pi$).

Furthermore, due to mechanical reasons, the angle θ_0 must stay between -180 and 180 degrees. A natural way to enforce this property is to add a repulsive torque on the corresponding joint's control. Since this torque τ_{sing} is in the *joint space*, we add it to τ_s (eq. (2)).

Since the amplitude of those repulsing forces grows faster than the PD control force and torque, the slave robot will not reach the undesired configurations. Meanwhile, the operator will 'feel' fast-growing force feedback if he tries to move the slave in those configurations. Thus, imposing constraints on the slave robot, in its control scheme, both in operational and joint space, actually constraints the operator's motions and makes them safer.

2.3.2 Virtual constraints

In this section, we will explain how potential fields can be used on the master's control, in order to help him to carry out some complex tasks, like moving on a perfect line, or help him stay out of some predefined zones.

Master servo loop constraints We propose to help the operator move the slave end effector on a predefined surface or curve by the following scheme. We project the operators cartesian position on the desired trajectory. We call this point a *Proxy*. The curve, or the control surface to follow is surrounded by an *attractive potential field* whose amplitude increases with the distance between the master end effector and the proxy. Then we apply the corresponding attractive force F_m^c on the haptic device's end effector. By choosing the appropriate gains, the operator will easily move on the unconstrained directions, but will have to fight high torques on its master device to go away from it. Subsequently, the slave robot, following the master device will move according to the predefined constraints. Moreover, to further enforce the constraint, we can use the potision of proxy to control the slave, instead of the real position of the master. Figure 2 displays the proxy and the corresponding force, when the constrained motion is along a line Δ.

This location of the constraints gives the better haptic feedback, since it runs at the speed of the master controller. However, despite ongoing work ([5] complex virtual interactions are very hard to computed in haptical real-time (in the thousand Hz range).

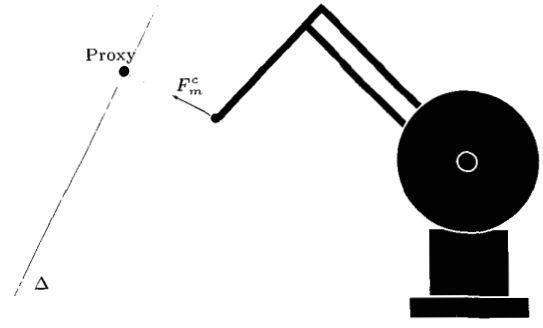

Figure 2: Attractive potential along a line

Master side virtual environment Constraining the movements of the end effector on a predefined curve or surface is simple enough to be incorporated to the servo loop, and running at the same frequency as the control F_m. However, we would like to put constraints on the whole robot movements (not only its end effector) and interact with complex virtual scenes (represented by thousands of triangles, for example) leading to much more complex computations.

To fulfill this requirement, we propose to integrate a model of the real slave robot inside a virtual environment (figure 4). In this 3D environment, the robot will follow the moves of the real one, but will also interact with models of real objects as well as purely virtual obstacles. We compute this interaction asynchronously with respect to the master's control, usually at a much slower rate.

This virtual world can also be displayed on the master screen, providing a convenient visual feedback to the operator. Should the slave and the master robots be in separate rooms, this virtual visual feedback is a good complement to a classical video display of the slave, since it can provide any point of view and any zoom of the real scene.

To model the interaction of the robot in its virtual environment, we define this environment with a set of n convex objects O_i (figure 3). Each object O_i is surrounded with a predefined repulsive potential field whose amplitude and range of action can be parameterized. Then, we compute the resulting forces of this potential field on each rigid moving part B_j of our robot. For each body B_j, we compute the shortest distance d_{ij} between the body and any obstacle. This distance computation will also provide the point of application p_{ij} and the direction of the partial virtual interaction force f_{ij}^{vp} between the part B_j of the robot and the ob-

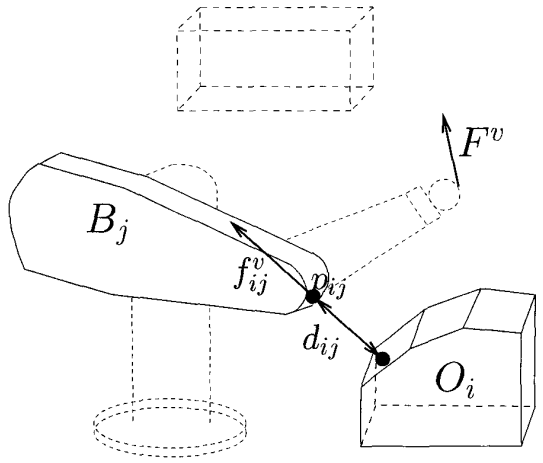

Figure 3: Interaction with the virtual environment: each Obstacle O_i produces a repulsion force f_{ij}^v on each part B_j of the robot, resulting in a force F_m^v at the end effector

ject O_i. Once this is done for all couples of objects and robot bodies, the final virtual interaction force applied by the environment will be :

$$F_m^v = J^{t^{-1}} \sum_{i=1}^{n} \sum_{j=1}^{m} J_{p_{ij}}^t f_{ij}^{vp} \qquad (3)$$

where $J_{p_{ij}}$ is the intermediate Jacobian of the slave robot at the point p_{ij}. Once computed, this force F_m^v will be sent to the master servo loop, and be added to F_m. Due to the slow update frequency of this forces, the amplitude of f_{ij}^{vp} should not vary too fast in order to preserve the haptic feedback stability and is likely to need experimental tuning.

3 Experimental results

In this section we describe our implementation of those principles on the robotic platform of the Stanford Robotic Manipulation Group.

3.1 Hardware architecture

The master device is a Phantom with 6 degrees of freedom. On the used model, only the first three joints of this phantom can be read and controlled. Thus our experiments will deal only with positions of the end effector, not its orientation. The slave robot is a PUMA 560.

We use a linux PC quadri-processor Pentium Pro 200 MHz to control the Phantom, run the display and compute the virtual constraints on the master device. The slave controller runs on a PC Pentium II 333 MHz, using the QNX realtime OS. Both PCs are linked together thru a switch, using a dedicated 100Mb/s ethernet line. This experimentals setup prefigures experiments on real medical robots.

3.2 Software implementation

On the Master PC, we use three separate processes to perform the different tasks, thus exploiting the physical processors. The three process communicate using UNIX UDP sockets. As a consequence, the distance computations described in section 2.3.2 is running asynchronously with the control, typically, much slower.

We implemented this distance computation using the Proximity Query Package (PQP) [2], freely provided by the deparment of Computer Science of the university of North Carolina. For our tests, we modelled the shoulder and the elbow of the puma using respectively 46 and 26 triangles. A simple environment was modelled using two sets of 22 and 12 triangles.

The Master PC sends its position to the slave PC at the same rate as its servo loop, always providing the freshest value to the slave. In order to achieve optimal performance, this communication uses INET UDP sockets with fixed length 200 bytes packets. This size being smaller than the ethernet MTU (which is 1500 bytes), each position update is completely sent on the network in one shot, avoiding data fragmentation.

The slave controller runs at a slower rate than the master controller, and sends its positions at the same rate as its control loop, which is slower than the masters one. In order to avoid accumulation of data coming from the master, we set the buffer size in the IP stack to the length of one of the packet we receive. Using this technique, we do not need to flush the socket each time we read it: each new update from the master erases any older data present in the socket buffer.

The graphical display of the scene is implemented using the client/server architecture described in [8], using *Mesa* implementation of openGL on a 3DFX graphical adapter (figure 4).

3.3 Performance

In the most complex case, moving on a line, with all virtual obstacles enabled, the master controller runs at 5000 Hz, which is appropriate to achieve a good feedback on the master device.

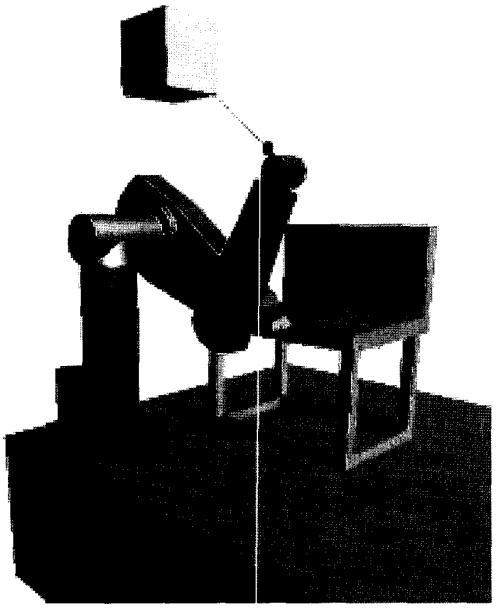

Figure 4: 3D graphical display of the scene, including the robot, real objects (the bench) and virtual ones (the floating cube).

A servo rate of 600 Hz was proven sufficient to run smoothly the slave PUMA 560.

The interaction with the virtual environment was running at 200 Hz, beeing limited by the CPU power. A faster computation would greatly improve the haptic feedback.

3.4 Feedback

The following plots display the feedback (amplitude in the vertical direction) felt by the operator on the master device, during different interactions :

3.4.1 Real Contact

Figure 5 displays the force feedback on the master robot when the slave end effector makes two consecutive contacts with an horizontal plane (z= -0.06 m). When the robot moves in free space, the feedback force is almost zero. When the robot hits the plane, the feedback on the master device rapidly increases as the error between the slave and the master augment. Contact with any part of the robot (not necessarily its end effector) would give a similar feedback to the operator, in the direction of the applied force.

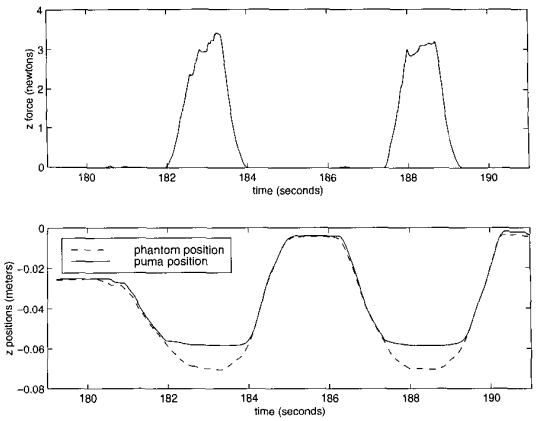

Figure 5: Force Feedback during a real contact on a plane $z = -0.06m$

3.4.2 Virtual Obstacles: Repultion fields

Contact with a virtual obstacle results in a feedback profile displayed in figure 6. The best results for computing f_{ij}^{vp} in equation (3) were achieved using a cubic function of the distance to the obstacle. As in the previous section, the obstacle is a z=-0.06m plane. The effects of the repulsion field are effectively felt at 1cm from the obstacle (about 2 Newtons), but the feedback profile is much less steep than the one associated with a real contact. Because of the relatively slow computation of the interaction with the virtual world (200 Hz), it was necessary to use low gains for the computations of f_{ij}^{vp} (eq. (3)) to ensure the master control stability. Small steps that can be observed on the plots are also due to this slow rate of computation.

3.4.3 Movement along a line: Attractive fields

Figure 7 display the force feedback during a constrained motion along an horizontal line (z = 0.018 m). Here, the feedback is much more stronger than in the two other exemples: when the user is moving along the constraint line (between start and $t = 48$ and $t = 50$ to the end), the feedback force is almost zero, but when the operator tries to go away from the line(between $t = 48$ and $t = 50$), the maximum saturated force of five newtons is quickly reached (for an error of 5 milimeters). This clearly shows the advantages of including the constraint forces computation inside the master servo loop whenever the computation time is compatible with 'haptic'

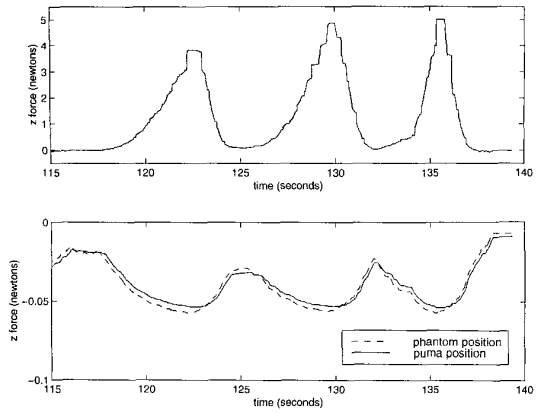

Figure 6: Force feedback during interaction with a virtual obstacle

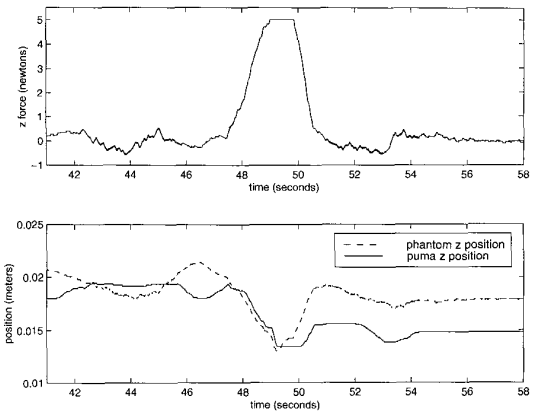

Figure 7: Force feedback during a constrained line movement: trying to go away from the constraint triggers a strong feedback

real-time.

4 Summary and future trends

This work provides a control framework summarized in figure 8 to enrich robotic teleoperation with various types of constraints. Those constraints will increase the safety and the dexterity of the operator in tele-surgery as well as various tele-exploration applications. This framework was extensively experimented, providing a demonstration stable enough to be run tens of times for our visitors.

However, this first haptic teleoperation experiment raises several issues that will be investigated in the future.

Preliminary experiments show that in the case of network delays greater than 200 milliseconds, this control framework is not sufficient, and must be complemented, for example, with a wave transmission scheme ([6]). The ability to cope with delays longer than 100 milliseconds opens the gates of teleoperation over the internet, for example.

Our current virtual interaction is implemented without using the proxy principles as defined in [8], because those principles model only a moving point in a virtual word, and we need to compute the interaction of whole 3D bodies (the robots parts). But as a consequence, the notion of inside or outside obstacle is not present in our implementation. In the near future, we plan to integrate previous work that better model interaction with virtual obstacles.

We also plan to use this framework to control a real medical teleoperation platform, using a patient 3d model for the virtual environment. On such platform, the slave robot will be more precise than our PUMA and will have less friction, leading to better performance for our control framework.

References

[1] Steve Charles, Hari Das, Tim Ohm, Curtis Boswell, Guillermo Rodriguez, Robert Steele, and Dan Istrate. Dexterity-enhanced telerobotic microsurgery. In *Proc. of the 8th Int. Conf. on Advanced Robotics (ICAR '97)*, 1997.

[2] Stefan Gottschalk, Ming Lin, and Dinesh Manocha. Obb-tree: A hierarchical structure for rapid interference detection. In Holly Rushmeier, editor, *Computer Graphics Proceedings, Proceedings of SIGGRAPH, Annual Conference Series, New Orleans, Louisiana*, pages 171–180, August 1996. http://www.cs.unc.edu/ geom/SSV/.

[3] Gary S. Guthart and J. Kenneth Jr. Salisbury. The intuitive telesurgery system: Overview and application. In *Proc. of the 2000 IEEE Int. Conf. on Robotics and Automation*, volume I, pages 618–621, 2000.

[4] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, Spring 1986.

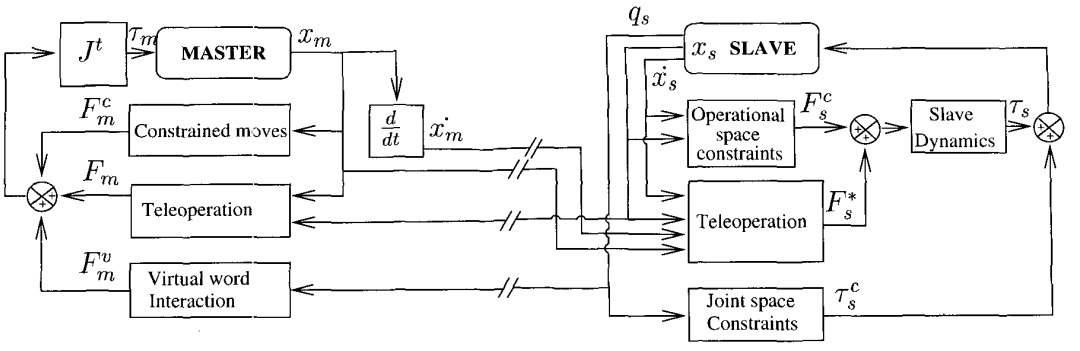

Figure 8: Complete control scheme

[5] Jean-Christophe Lombardo, Marie-Paul Cani, and Fabrice Neyret. Real-time collision detection for virtual surgery. In *Computer Animation, Geneva*, May 1999.

[6] Gnter Niemeyer and Jean-Jacques E. Slotine. Towards force-reflecting teleoperation over the internet. In *Proc. of the 1998 IEEE Int. Conf. on Robotics and Automation*, pages 1909–1915, 1998.

[7] Schneider O., Troccaz J., Chavanon O., and Blin D. Padyc : a synergistic robot for cardiac puncturing. In *Proc. of the 2000 IEEE Int. Conf. on Robotics and Automation*, volume III, pages 2883–2888, 2000.

[8] Diego C. Ruspini, Krasimir Kolarov, and Oussama Khatib. The haptic display of complex graphical environments. In *SIGGRAPH*, pages 345–352, 1997.