

Haptically Augmented Teleoperation

Nicolas Turro
Inria/Stanford University Robotics Group
Nicolas.Turro@sophia.inria.fr

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

Abstract: This article presents various experiments conducted at the Stanford Robotics Group to enhance teleoperation thru the use of movement constraints. Several types of constraints have been studied : Safety constraints, limiting the movements of the slave robot into safe areas and robot and guidance constraints, helping the operator to move according to a pre-defined geometrical path (along a line, on a plane). Those constraints have been implemented using force feedback on the master device and potential fields are used to compute the amplitude of the force feedback according to the constraints.

1. Introduction

Stanford Robotics Group has a long background of designing robotic assistants for the execution difficult tasks. As described in [1] and [2], the two platforms ROMEO and JULIET demonstrated how some basic behaviours can be performed by robots. The next logical step in this research is to include teleoperation capabilities to those platform, in order for a human to guide the robots remotely and use the various elementary behaviours when needed. Teleoperation and behaviour could also be combined in order, for example, to remotely wipe out a blackboard : the robotic behaviour automatically enforce a contact with the brush, with a predetermined force normal to the surface, while the operator guide the brush in the directions tangential to the surface.

The use of a haptic device as master for the teleoperation helps the operator to perform precise and complex tasks : the force feedback is computed using data from the slave robot (from force sensors, from ultrasound distance sensors), or from geometrical constraints computed on the master site. Using this feedback, the operator may guide the slave robot safely in narrow corridors without collisions, or write on a blackboard applying the righth pressure with the pencil, or remotely interact with an human, or make perfect line movements.

In this paper, we present both the control framework we are using to achieve this haptically augmented teleoperation, and the first experimental results of our implementation.

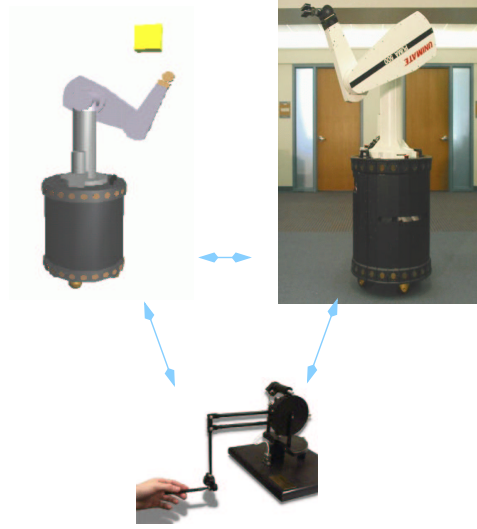


Figure 1. The teleoperation environment

2. Principles

As displayed in figure 1, our environment is built around three main components: a master robot, a slave robot and a virtual 3D representation of the robot and its environment. The whole control scheme we evaluated is presented in figure (2). It comprises two servo loops to control the master device and the slave robot. The two servo loops communicate with each other by sending their respective positions and velocities.

2.1. The basic teleoperation control

The core control scheme of the teleoperation reproduces movements of the master device on the slave robot, and it also provides the operator with some feedback of the interaction between the robot and its environment. The master and the slave robots have dramatically different geometry and dynamics properties. 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 .

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

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 performances, but with lower gains, making the robot more compliant and sensitive to external

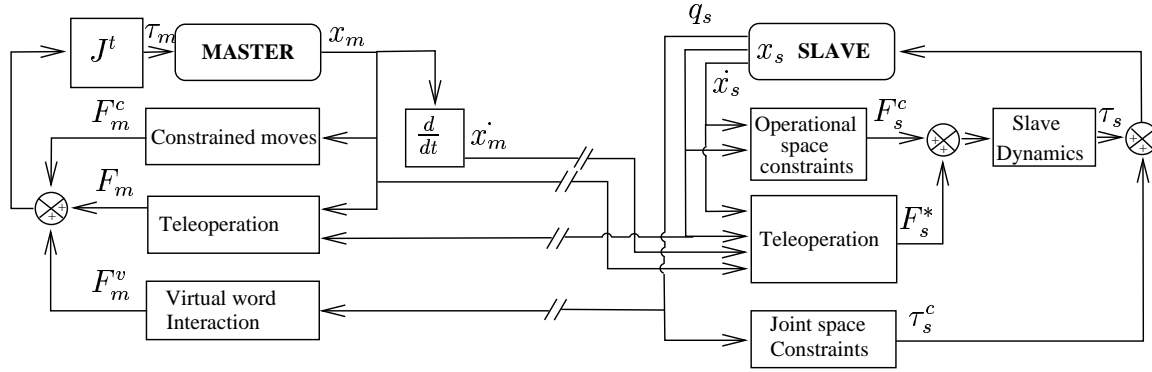


Figure 2. Complete control scheme

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 minimised 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. Constraints

Our main contribution was to evaluate different ways to constrain our teleoperation system. Three alternative have been studied :

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

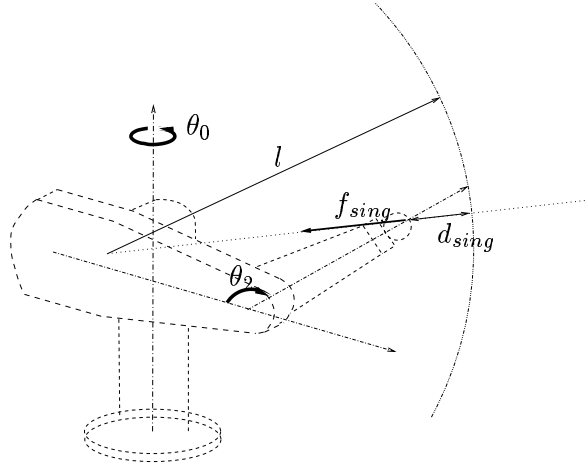


Figure 3. Singularity repulsive force

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 to move in given directions. Then, the basic control scheme described above will haptically render those slave constraints on the master device.

For example, our PUMA slave robot has a singularity on its second joint (elbow singularity) when the angle θ_2 (figure 3) 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 3. 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 length l of the maximal extension of the robot (corresponding to $\theta_2 = \pi$).

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

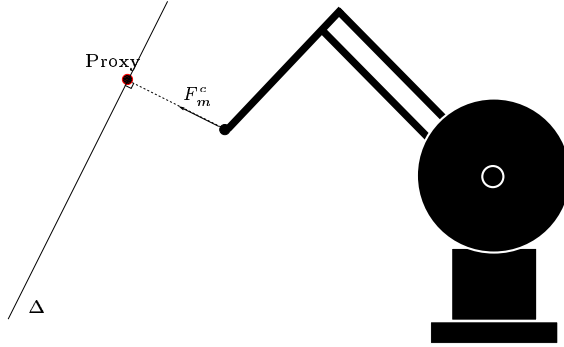


Figure 4. Attractive potential along a line

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 position of proxy to control the slave, instead of the real position of the master. Figure 4 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 complex virtual interactions are very hard to computed in haptical real-time (in the thousand Hz range).

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 6). 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.

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 5). 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

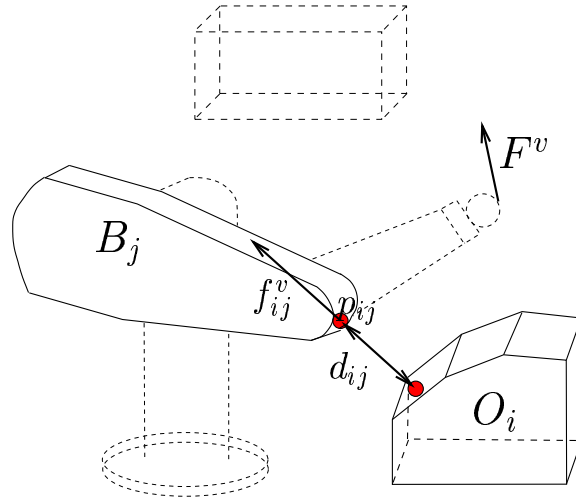


Figure 5. Interaction with the virtual environment: each Obstacle O_i produces a repulsion force f_{ij}^v on each part B_j of the robot, resulting in a force F_m^v at the end effector

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 object 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 :

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

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 these 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 Setup and results

We applied this framework to two configurations : first a fixed puma 560, and then a mobile manipulator.

3.1. Hardware architecture and implementation

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.

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

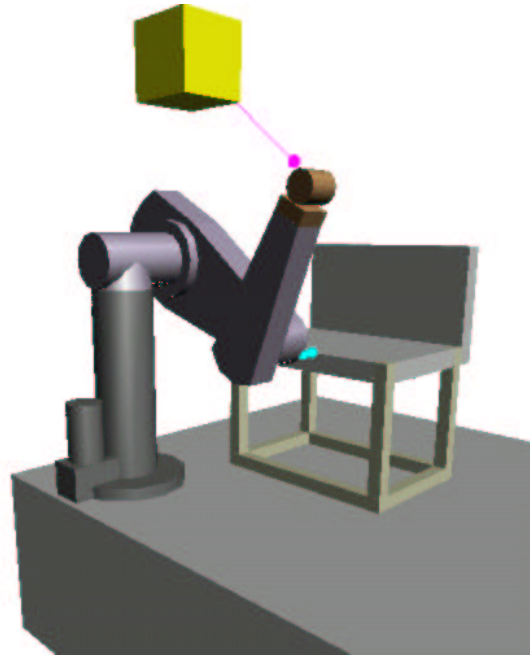


Figure 6. 3D graphical display of the scene, including the robot, real objects (the bench) and virtual ones (the floating cube).

100Mb/s ethernet line. 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, [3]), freely provided by the department of Computer Science of the university of North Carolina.

The graphical display of the scene is implemented using the client/server architecture described in [4], using *Mesa* implementation of OpenGL on a 3DFX graphical adapter (figure 6).

3.2. 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.

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, being limited by the CPU power. A faster computation would greatly improve the haptic feedback.

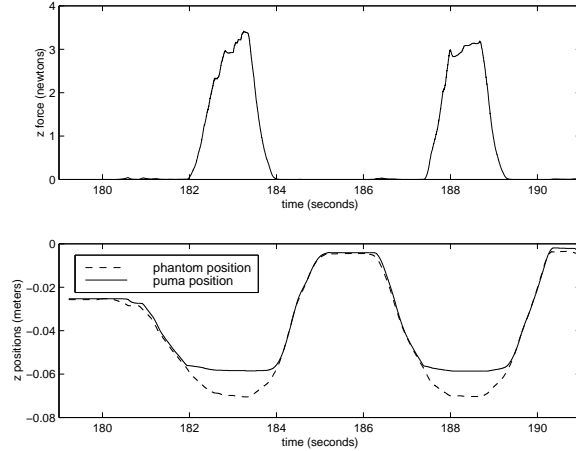


Figure 7. Force Feedback during a real contact on a plane $z = -0.06m$

3.3. Fixed Puma experiments

In this case, the slave and the master communicate thru a dedicated 100Mb/s ethernet cable. 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, each new update from the master erases any older data present in the socket buffer.

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

3.3.1. Real Contact

Figure 7 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 augments. 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.

3.3.2. Virtual Obstacles: Repulsion fields

Contact with a virtual obstacle results in a feedback profile displayed in figure 8. 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

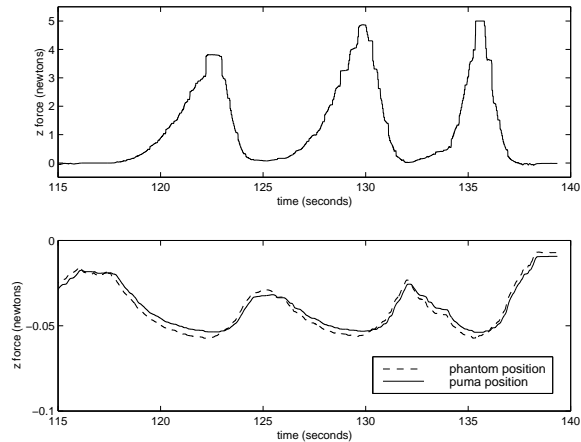


Figure 8. Force feedback during interaction with a virtual obstacle

obstacle is a $z=-0.06\text{m}$ 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.

3.3.3. Movement along a line: Attractive fields

Figure 9 display the force feedback during a constrained motion along an horizontal line ($z = 0.018\text{ m}$). 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 millimeters). This clearly shows the advantages of including the constraint forces computation inside the master servo loop whenever the computation time is compatible with 'haptic' real-time.

3.4. The mobile manipulator

The mobile manipulator consists in a puma arm mounted on an holonomic nomadics XR4000 base. The master and the slave are communicating using a RANGELAN2 radio ethernet network, which is the main difference from the fixed experimental platform.

The bandwidth of the radio link, which is supposed to be 1.2 Mbits per seconds was not sufficient to get more than a few tens of updates per seconds. This rate was experimentally proven sufficient to send goal position to the slave robot, but not to provide a realistic force feedback on the master device, as in the fixed platform case. In consequence, information coming from the slave robot have been used only to update the 3D virtual representation of the

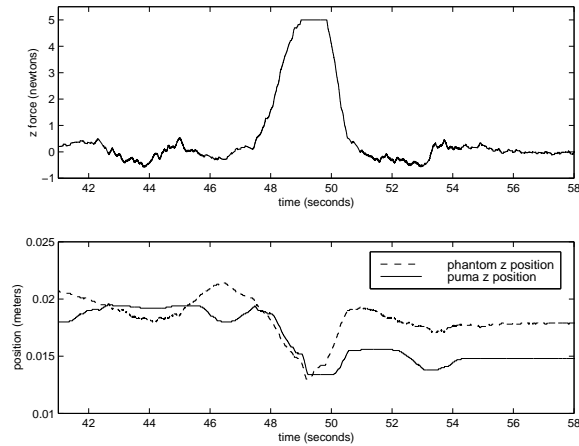


Figure 9. Force feedback during a constrained line movement: trying to go away from the constraint triggers a strong feedback

robot, and the force feedback on the master could only be computed using the local constraints in the master control loop (movements of the master device constrained along predefined geometrical primitives).

4. Conclusion

This framework was extensively experimented, providing a demonstration stable enough to be run tens of times for our visitors.

The experimental results in the fixed puma configuration are promising, and we plan to combine soon teleoperation and compliant motion with multi-contacts on the environment.

In the case of the mobile manipulator, our preliminary experiments show that our control framework is not adequate for low bandwidth. It could be complemented, for example, with a wave transmission scheme ([5]).

References

- [1] Khatib O, 1999 Mobile Manipulation: The Robotic Assistant. *Journal of Robotics and Autonomous Systems*, vol. 26, 1999, pp. 175-183
- [2] Khatib O, Yokoi K, Brock O, Chang K, Casal A Robots in Human Environments: Basic Autonomous Capabilities. *International Journal of Robotics Research*, vol. 18, no. 7, 1999, pp. 684-696
- [3] Gottschalk S, Lin M, Manocha D 1996 OBB-Tree: A Hierarchical Structure for Rapid Interference Detection. *Proceedings of SIGGRAPH*, Annual Conference Series, pp 171-180, <http://www.cs.unc.edu/geom/SSV/>
- [4] Ruspini D C, Kolarov K, Khatib O, 1997 The Haptic Display of Complex Graphical Environments. *SIGGRAPH*, pp. 345-352
- [5] Niemeyer G, Slotine J-J, 1998 Towards Force-Reflecting Teleoperation over the Internet. *Proc. of the 1998 IEEE Int. Conf. on Robotics and Automation*, pp. 1909-1915