Haptic Interaction with Virtual Avatars

François Keith, Paul Evrard, Jean-Rémy Chardonnet, Sylvain Miossec, and
Abderrahmane Kheddar

AIST/CNRS Joint Japanese-French Robotics Laboratory, Tsukuba, Japan
{francois.keith,evrard.paul,jr-chardonnet,sylvain.miossec,
abderrahmane.kheddar}@aist.go.jp

Abstract. In this paper we present an interactive dynamic simulator
for virtual avatars. It allows creation and manipulation of objects in a
collaborative way by virtual avatars or between virtual avatars and users.
The users interact with the simulation environment using a haptic probe
which provides force feedback. This dynamic simulator uses fast dyna-
mics computation and constraint-based methods with friction. It is part
of a general framework that is being devised for studies of collaborative
scenarios with haptic feedback.

Key words: Dynamic simulator, constraint-based methods, user-object-
avatar haptic interaction

1   Introduction

Recently, several simulators for virtual avatars have been developed for many
purposes and applications. In the computer graphics community, avatars dy-
namic simulators are proposed with physics-based controllers for off-line or in-
teractive realistic animation in gaming or motion generation of digital actors,
e.g. see [1][2], but interactivity with haptic feedback has not been of a major
concern. In the robotics field, dynamic simulators have also been developed for
the purpose of planning or sensory control simulation. For example, Son *et al.* [3]
proposed a general framework for robotic dynamic simulation with interactive ca-
pabilities including force feedback. This simulator accounts for constraint-based
contact modeling with friction; it uses a hybrid method to switch between dif-
ferent contact statuses. Khatib's team proposed an impressive framework, called
SAI, for interactive dynamic simulation [4] using the operational space formu-
lation with prioritized tasks [5] and was probably among the firsts to use hap-
tic feedback for interaction with a virtual avatar. At AIST, an open platform
named OpenHRP has been developed to be dedicated to the general humanoid
studies [6]. The next release (version 3) is forecast to be distributed freely with
open source code. However, OpenHRP is not interactive and does not include
force feedback. There are other one that are or will be commercially available
(e.g. R-Station or Microsoft Robotic Studio). In virtual prototyping, Duriez *et
al.* [7] showed interactive simulation for deformable objects using constraint-
based methods and solving contact with friction using iterative algorithms that

include haptic feedback. Using the same basis as in [4], Chardonnet *et al.* [8] proposed a fast dynamic simulator for humanoids without discrete Coulomb's friction cones and able of handling complex shapes.

This work extends our previous framework to handle fast constrained-based dynamic computation with force feedback. Comparing to the state-of-the-art, we demonstrate that our simulator is able to handle complex shapes in real-time and integrate force feedback without any specific treatment. Our framework intends to include not only task-driven simulations, but also cognitive aspects linked to haptic interaction such as haptic patterns of communication and advanced interaction with digital actors that can be either virtual or real (robots). This paper focuses only on the dynamic and computer haptics with details concerning the proposed integrative software architecture. This framework is devised to integrate developments in digital actors control with a focus on haptic collaborative tasks and communication.

2   Software Architecture

2.1   Main requirements

Our goal is to realize a high fidelity haptic interaction with virtual avatars that are driven by an autonomous or a preprogrammed behavior. The user will be interacting with the virtual avatar using any haptic device. A large paradigm of possible scenarios are envisaged; as a result the contact space formulation must comply with several constraints such as allowing dynamic interaction between bodies of different kinematics structure and with different materials composing the surrounding virtual environment. The contact space formulation needs to subtly integrate the avatar dynamics together with other phenomena such as impacts, static and dynamic friction, and deformations. Altogether, behavioral knowledge will be integrated within a haptic module to drive virtual avatar interactions. In this study we distinguish two behavioral modules:

- haptic interaction induced from tasks (e.g. a human operator manipulates a virtual object in a collaborative way with an autonomous virtual avatar);
- haptic interaction induced from communication (e.g. a human operator touching directly a virtual avatar: hand shaking, taping, etc.).

2.2   Implementation of the Direct Dynamic Model

To compute the direct dynamic model, we use Featherstone's Articulated Body Method [9, 10], which focuses on a chain of joints with a single degree of freedom.

We detail the algorithm with multiple DOF joints merely presented in [9]. Using null-inertia virtual bodies and joints to model a joint with multiple DOF (like in the SAI framework) causes numerical problems and may induce instability of the simulation. Hence we have to consider multiple DOF joints as a specific joint, without calling to any artifacts. We give as an example the case of 3DOF spherical joints (found in humans and animals).

Articulated Body Method for multiple DOF joints. We use the spatial notations similarly to [9], for more concise writing. In particular, the ˆ symbol represents spatial variables. The explanations of spatial notations are given in [9, 10]. We simply rewrite the three recursions with the extension to n_{DOF}–DOF joints. For the sake of clarity, we restrict the reasonning to serial chain robots. The extension to a branched robot is straightforward and detailed in [9].

We first present the recursive kinematic equations. At the same time we present the kinematic model of the joints. Let $_a\hat{\mathbf{X}}_b$ be the transformation from frame attached to b to frame attached to a, Li the link i and Ji the joint i.
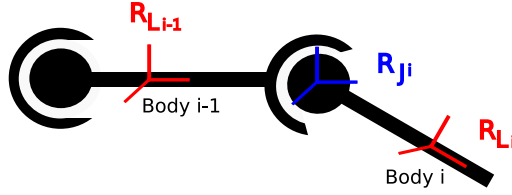

Fig. 1. Frames associated to links and joints.

The frame of a body is attached to its CoM, the frame of a joint is attached to its center of rotation. Only $_{Ji}\hat{\mathbf{X}}_{Li}$ is computed from the joint parameterization. Its computation will be detailed later for the spherical joint and different parameterization. The recursion equations at the position level are:

$$_0\hat{\mathbf{X}}_{Ji} = {}_0\hat{\mathbf{X}}_{L(i-1)}{}_{L(i-1)}\hat{\mathbf{X}}_{Ji}$$
$$_0\hat{\mathbf{X}}_{Li} = {}_0\hat{\mathbf{X}}_{Ji}{}_{Ji}\hat{\mathbf{X}}_{Li} \tag{1}$$

At the velocity level, the recursion relation is given by:

$$\hat{\mathbf{v}}_{\mathbf{i}} = \hat{\mathbf{v}}_{\mathbf{i-1}} + \hat{\mathbf{S}}_i\hat{\mathbf{v}}_{\mathbf{ri}} \tag{2}$$

where $\hat{\mathbf{S}}_i(6 \times n_{\mathrm{DOF}})$ maps the reduced spatial relative velocity $\hat{\mathbf{v}}_{ri}(n_{\mathrm{DOF}} \times 1)$ of joint i to the relative velocity in the world coordinates. It is obtained from:

$$\hat{\mathbf{S}}_i = {}_0\hat{\mathbf{X}}_{Ji}\hat{\mathbf{S}}_{ri} \tag{3}$$

where $\hat{\mathbf{S}}_{ri}$ maps $\hat{\mathbf{v}}_{ri}$ to the local coordinates. $\hat{\mathbf{S}}_{ri}$ will be constant if the joint velocity parameters in $\hat{\mathbf{v}}_{ri}$ are simply translational and rotational velocity components. $\hat{\mathbf{S}}_{ri}$ could be non-constant for other velocity parameters. We will choose such $\hat{\mathbf{v}}_{ri}$ to have $\hat{\mathbf{S}}_{ri}$ constant. The acceleration recursive relation is obtained by derivation of (2).

$$\hat{\mathbf{a}}_i = \hat{\mathbf{a}}_{i-1} + \hat{\mathbf{S}}_i\hat{\mathbf{a}}_{ri} + \hat{\mathbf{v}}_{i-1}\hat{\times}\hat{\mathbf{S}}_i\hat{\mathbf{v}}_{ri} \tag{4}$$

If $\hat{\mathbf{S}}_{ri}$ is not constant, the term $_0\hat{\mathbf{X}}_i\dot{\hat{\mathbf{S}}}_{ri}\hat{\mathbf{v}}_{ri}$ will be added.

Given the introduction of multiple-DOF joints, the first recursion writes:

$$\begin{aligned}
\hat{\mathbf{S}}_i &= {}_0\hat{\mathbf{X}}_{Ji}\hat{\mathbf{S}}_{ri} \\
\hat{\mathbf{v}}_i &= \hat{\mathbf{v}}_{i-1} + \hat{\mathbf{S}}_i\hat{\mathbf{v}}_{ri} \\
\hat{\mathbf{c}}_i &= \hat{\mathbf{v}}_{i-1}\hat{\times}\hat{\mathbf{S}}_i\hat{\mathbf{v}}_{ri}
\end{aligned} \tag{5}$$

where $\hat{\mathbf{c}}_i$ is an intermediate computation needed in following recursions coming from (4). If $\hat{\mathbf{S}}_{ri}$ is not constant, the previously presented term is added.

In order to write the second recursion, we need first to define the multiple-DOF actuator model. The spatial force produced by the actuator $\hat{\mathbf{f}}^a$ is:

$$\hat{\mathbf{f}}_i^a = \hat{\mathbf{F}}_i\mathbf{Q}_i \tag{6}$$

where $\mathbf{Q}_i(n_{\text{DOF}}\times 1)$ is the vector of actuator forces and/or torques, $\hat{\mathbf{F}}_i$ maps $\mathbf{Q}_i$ to the spatial force produced by the actuator $\hat{\mathbf{f}}^a$ expressed in the world coordinates. It is obtained from:

$$\hat{\mathbf{F}}_i = {}_0\hat{\mathbf{X}}_{Ji}\hat{\mathbf{F}}_{ri} \tag{7}$$

$\hat{\mathbf{F}}_{ri}$ will be later given for the spherical joint.

The second recursion equations are then given by

$$\begin{aligned}
\hat{\mathbf{p}}_i^v &= \hat{\mathbf{v}}_i\hat{\times}\hat{\mathbf{I}}_i\hat{\mathbf{v}}_i - \mathbf{f}^E \\
\hat{\mathbf{I}}_i^A &= \hat{\mathbf{I}}_i + \hat{\mathbf{I}}_{i+1}^A - \hat{\mathbf{h}}_{i+1}\mathbf{d}_{i+1}^{-1}\hat{\mathbf{h}}_{i+1}^S \\
\hat{\mathbf{p}}_i &= \hat{\mathbf{p}}_i^v + \hat{\mathbf{p}}_{i+1} + \hat{\mathbf{I}}_{i+1}^A\hat{\mathbf{c}}_{i+1} + \hat{\mathbf{h}}_{i+1}\mathbf{d}_{i+1}^{-1}\mathbf{u}_{i+1} \\
\hat{\mathbf{h}}_i &= \hat{\mathbf{I}}_i^A\hat{\mathbf{S}}_i \\
\mathbf{d}_i &= \hat{\mathbf{S}}_i^S\hat{\mathbf{h}}_i \\
\mathbf{u}_i &= \hat{\mathbf{S}}_i^S\hat{\mathbf{F}}_i\mathbf{Q}_i - \hat{\mathbf{h}}_i^S\hat{\mathbf{c}}_i - \hat{\mathbf{S}}_i^S\hat{\mathbf{p}}_i
\end{aligned} \tag{8}$$

where $\hat{\mathbf{h}}_i(6\times n_{\text{DOF}})$, $\mathbf{d}_i(n_{\text{DOF}}\times n_{\text{DOF}})$, $\mathbf{u}_i(n_{\text{DOF}}\times 1)$ depends on the number of DOF, while other terms have the same size as in the algorithm for 1-DOF joints. For the last body of the chain, the $i+1$ terms of the two first rows are eliminated. The notation ^{S} is the spatial transpose defined by

$$\forall\,\hat{\mathbf{A}} = \begin{bmatrix}\mathbf{A}_{up}\\\mathbf{A}_{dn}\end{bmatrix},\ \hat{\mathbf{A}}^S = \begin{bmatrix}\mathbf{A}_{dn}^T & \mathbf{A}_{up}^T\end{bmatrix} \tag{9}$$

where $\mathbf{A}_{up}$ and $\mathbf{A}_{dn}$ are two $3\times n_{\text{DOF}}$ matrices.

The third recursion is given by:

$$\begin{aligned}
\hat{\mathbf{a}}_0 &= -\hat{\mathbf{I}}_0^{-1}\hat{\mathbf{p}}_0 \\
\dot{\hat{\mathbf{v}}}_{ri} &= \mathbf{d}_i^{-1}(\mathbf{u}_i - \hat{\mathbf{h}}_i^S\hat{\mathbf{a}}_{i-1}) \\
\hat{\mathbf{a}}_i &= \hat{\mathbf{a}}_{i-1} + \hat{\mathbf{c}}_i + \hat{\mathbf{S}}_i\dot{\hat{\mathbf{v}}}_{ri}
\end{aligned} \tag{10}$$

Case of spherical joints. Whatever the spherical joints position parameterization, we use relative rotation velocity and acceleration as velocity and acceleration parameters. This choice gives a simple expression for $\hat{\mathbf{S}}_{ri}$,

$$\hat{\mathbf{S}}_{ri} = \begin{bmatrix}\mathbf{I}_{3\times 3}\\\mathbf{0}_{3\times 3}\end{bmatrix} \tag{11}$$

The spherical joint will be used to simulate human elbow joint and hip joint. The actuator system is composed of several muscles in parallel. The real model for such an actuation system is very complex. We choose an abstract actuator model which allows to apply three components of torque around the three axis of the joint frame. Subsequently, the matrix $\hat{\mathbf{F}}_{ri}$, representing the spherical joint model, can be written as:

$$\hat{\mathbf{F}}_{ri} = \begin{bmatrix} \mathbf{0}_{3\times3} \\ \mathbf{I}_{3\times3} \end{bmatrix} \tag{12}$$

The previous choice of parameterization in velocity has the advantage to be independent of the choice of parameterization in position. Many position parameterizations are possible for spherical joints: quaternion, Euler angles or exponential map. One can refer to [11] for a comparison of those parameterizations. For simulation, we choose to use the quaternion for two reasons: (i) there is no singularity and (ii) the computation of the jacobian is faster (there are few calculations). The fact that there are four parameters instead of possibly only three is not a problem for simulation.

We present how to compute the rotation matrix and the derivative of the position parameterization for integration purpose. We use the notation $\omega = \hat{\mathbf{v}}_{ri}$.

The quaternions, noted $q = (q_0, \mathbf{q}) = (q_0, q_1, q_2, q_3)$, do not have singularity and a normalized quaternion represents a rotation of angle q_0 around the $\mathbf{q}$ axis. The corresponding rotation matrix $_{Ji}\mathbf{R}_{Li}$ is [12]:

$$_{Ji}\mathbf{R}_{Li} = \begin{bmatrix} q_0{}^2 + q_1{}^2 - \frac{1}{2} & q_1 q_2 + q_0 q_3 & q_1 q_3 - q_0 q_2 \\ q_1 q_2 - q_0 q_3 & q_0{}^2 + q_2{}^2 - \frac{1}{2} & q_2 q_3 + q_0 q_1 \\ q_1 q_3 + q_0 q_2 & q_2 q_3 - q_0 q_1 & q_0{}^2 + q_3{}^2 - \frac{1}{2} \end{bmatrix} \tag{13}$$

The derivative of quaternion can be linked to the angular velocity [13]:

$$\dot{q} = \frac{1}{2}\omega' \circ q \ \text{ where } \ \omega' = (0, \omega) \tag{14}$$

2.3 Constraint-based force computation

The general forward dynamics equation for multi-body articulated systems is generally written in the following closed-form:

$$\ddot{\mathbf{q}} = \mathbf{A}(\mathbf{q})^{-1}\left(\boldsymbol{\Gamma}(\mathbf{q}) - \mathbf{b}\left(\mathbf{q}, \dot{\mathbf{q}}\right) - \mathbf{g}(\mathbf{q})\right) + \mathbf{A}(\mathbf{q})^{-1}\mathbf{J}_c^T\mathbf{f}_c \tag{15}$$

where $\mathbf{A}$ is the inertia matrix, $\boldsymbol{\Gamma}$ the joint torques, $\mathbf{b}$ the Coriolis effect and the known external forces, $\mathbf{g}$ the gravity and $\mathbf{f}_c$ the external forces, typically contact forces. This equation can be written in operational space (here the contact space):

$$\mathbf{a}_c = \boldsymbol{\Lambda}_c^{-1}\mathbf{f}_c + \mathbf{a}_{\text{free}} \tag{16}$$

where $\boldsymbol{\Lambda}_c^{-1} = \mathbf{J}_c\mathbf{A}^{-1}\mathbf{J}_c^T$ is the operational space inertia matrix [14], $\mathbf{a}_{\text{free}}$ is the free acceleration of the contact points that is computed using Featherstone's algorithm [9]. Considering contact forces is not an easy issue because they are

unknown in simulation. We use constraint-based methods to solve this problem; they explicitly enforces for non-penetration constraints in the equations that is:

$$0 \leq \mathbf{f}_c \perp \mathbf{a}_c \geq 0 \tag{17}$$

The last two equations form a linear complementary problem (LCP) and can be easily solved using for example Lemke's algorithm. However, adding Coulomb's friction introduces a non-linearity in equation (16). This is generally written as:

$$\|\mathbf{f}_t\| \leq \mu f_n \tag{18}$$

where μ is the coefficient of friction. Keeping an LCP form to solve friction requires discreet friction cones which highly increases the size of the Operational Matrix and the computation time with a compromise on accuracy. Thus we favor iterative methods, more specifically a Gauss-Seidel type method [15, 7, 8], which does not require friction cone discretization. Compared to LCP formulation, Gauss-Seidel type method is much faster and more precise [7]. Once contact forces are computed, the dynamics are updated. We get an overall computation in $\mathcal{O}(nm+m^2)$ with n the number of bodies and m the number of contact points.

2.4   Integration

Our software developments were realized under an object oriented integrative framework called AMELIF[1] developed in C++. This framework is still under development and is forecast to be published in the future. It proposes a structure and an API for the representation of virtual scenes including articulated bodies, and interfaces for driving simulations and manipulating the elements of the scene. AMELIF has been created in order to allow fast and easy prototyping of virtual reality algorithms, and most particularly algorithms related to virtual avatars. The modularity of our framework allows the customization and replacement of most components without modifying the core application, thus guarantying consistency between the developments and interoperability between customized components. AMELIF is a cross platform framework and has been successfully tested under the Windows and Ubuntu Linux operating systems.

AMELIF includes a core application for displaying and running the simulations, and a core library that provides, among others, interfaces for communication with the core application and a set of components for the creation, display and manipulation of virtual scenes. One of the communication interfaces allows writing simulations that can be run from the core application. This interface has three main methods: one that is responsible for the initialization of the scene, one that drives one simulation step, and one for the cleanup. The state of the virtual scene is accessible from these methods. Each implementation of this interface has to be compiled as a dynamic library that will be loaded by the application. The application will automatically call the methods for initialization, simulation and cleanup.

---

[1] Avatar Multimodal Environment Libraries and Integrative Framework

The simulator we developed under this framework is built upon four modules. Since each module has an abstract interface, each of these modules is independent of the implementation of the other components. Some of them have their own representation of the virtual scene, which is updated by reading data from the core virtual scene. These modules are:

- the *dynamics algorithms*: its main role is to implement the computation of accelerations for each body and joint of the scene as a function of joint torques and external forces and moments applied on the bodies. We implemented the Featherstone algorithm described in section 2.2 in this module.
- the *collision detection*: its purpose is to detect collisions and to provide their description on demand to the other modules. Each collision is described by the coordinates of collision point, as well as the normal and one tangent vector to one of the colliding triangles. Now, this module encapsulates the PQP library for the collision detection.
- the *collision handler*: this component interacts with the collision detection to compute constraint-based forces resulting from the contact between bodies, and uses the dynamics algorithms to compute the resulting acceleration of the bodies.
- the *dynamic simulator*, which is a black-box that encapsulates the dynamic simulation: it uses these three modules to compute at each step the accelerations of each body and of each joint. It then integrates these accelerations to update the state of the virtual scene.
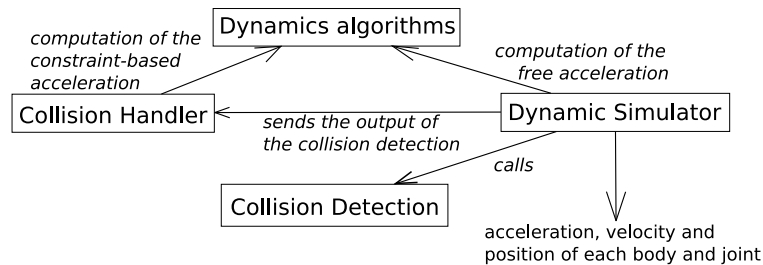

Fig. 2. Program structure.

3   Haptic feedback interfacing

Our simulator is centered on haptic interaction with virtual environments and with virtual avatars. An example of such a haptic interaction is the realization of a collaborative manipulation task with haptic feedback. For this purpose, we need to integrate a haptic device to our simulator. We interfaced the

PHANTOM© Omni™ device commercialized by SensAble Technologies[2]. This device has six degrees of freedom with a three degree-of-freedom force feedback. The hardware limits of velocity we can apply are given by the maker.

For the time being, we consider two different ways of interacting: touching or dragging objects (including avatars). In both cases, interaction will add an external force $\mathbf{f}_e$ to the dynamics of the objects:

$$\ddot{\mathbf{q}} = \mathbf{A}^{-1}\left(\boldsymbol{\Gamma}(\mathbf{q}) - \mathbf{b}\left(\mathbf{q},\dot{\mathbf{q}}\right) - \mathbf{g}(\mathbf{q})\right) + \mathbf{A}^{-1}\mathbf{J}_c^T\mathbf{f}_c + \mathbf{A}^{-1}\mathbf{J}_e^T\mathbf{f}_e \qquad (19)$$

When an object is touched, the feedback force is obtained via the Sensable OpenHaptics™ Toolkit. This library provides a collision detection algorithm that computes the feedback force due to contact between the virtual tip of the device handle and the virtual environment, as well as the coordinates of the contact point. The interaction force at the contact point is computed by weighting the feedback force with an arbitrary chosen coefficient.

Dragging objects is useful for tasks like pick-and-place, pushing or collaborative tasks (with avatars). As the user moves the device, the object behaves accordingly while the user feels its weight. To achieve this, the most common way used in interactive simulations is to model a virtual coupling intermediary between the device and the object, which is a spring-damper model:

$$\mathbf{f}_e = k_p(\mathbf{x}_{\text{device}} - \mathbf{x}_{\text{contact}}) + k_v(\dot{\mathbf{x}}_{\text{device}} - \dot{\mathbf{x}}_{\text{contact}}) \qquad (20)$$

where $\mathbf{x}_{\text{device}}$ and $\mathbf{x}_{\text{contact}}$ are the positions/orientations of the haptic device and of the object, respectively. k_v is chosen so that $k_v = \sqrt{2mk_p}$, with m the mass of the object.

4   Experiments

We ran simulations on a 1.66GHz notebook PC running under Windows. The time step of the simulation is 1msec and we use a simple Euler integration. The avatar is either the 32-DOF HRP-2 robot or a human avatar with spherical joints. In these simulations, we place an object on a table. Once the avatar takes it, we interact with the avatar via the object that we pull and push using the haptic device. For these examples, the avatar is not constrained to its initial position and follows the lead given by the user. Also, all bodies are position-controlled, but the compliance is only applied from the body touched by the haptic device to the waist. Thus we can move in a co-operative way the object from a place to another on the table.

For these simulations, the virtual avatar was position-controlled using the following PD law:

$$\boldsymbol{\Gamma} = k_p(\mathbf{q}_d - \mathbf{q}) - k_v\dot{\mathbf{q}} \qquad (21)$$

where k_p and k_v are positive coefficients. To make the robot compliant to the interaction force with the haptic device, the desired joint vector $\mathbf{q}_d$ is defined by

---

[2] www.sensable.com

the following equation:

$$\mathbf{M}\ddot{\mathbf{q}}_d + \mathbf{B}\dot{\mathbf{q}}_d = \mathbf{J}_e^T \mathbf{f}_e \qquad (22)$$

where $\mathbf{M}$ and $\mathbf{B}$ are diagonal positive matrices corresponding to a virtual inertia and a virtual damping, $\mathbf{f}_e$ is the interaction force with the haptic device, and $\mathbf{J}_e$ is the jacobian relating joint velocities to Cartesian velocities and angular velocities for the kinematics chain that contains all the compliant bodies. When the object handled by the avatar is grabbed with the haptic device, the additional force is transported to the wrists frames (where the force sensors are located on the real robot). The desired joint positions $\mathbf{q}_d$ are computed from these forces. Screenshots of these manipulations are on figure 3. The thin orange bar represents the haptic probe linked with the force feedback device.
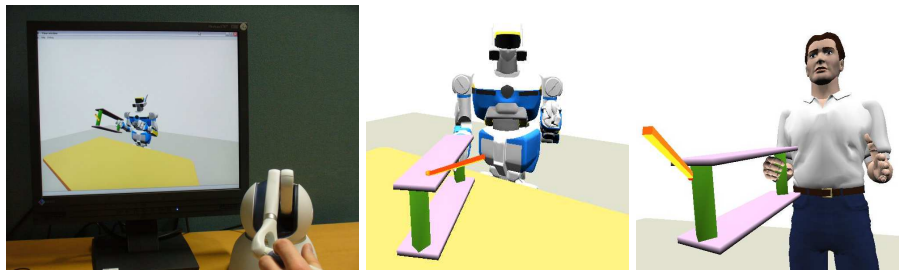

Fig. 3. Collaborative haptic manipulation with the HRP-2 and a human avatar.

The simulation with the human avatar raises a problem: the computation time. Indeed, due to the important number of collision points (more than 100), it took 300sec to simulate 10sec of simulation. For the HRP-2 with 32 collision points, it only takes 92sec. The number of contact points, at each foot, is big enough so that the simulation is slowed down, which does not allow a real-time motion of the avatar.

5   Conclusion and future work

In this paper, we presented a dynamic simulator centered on virtual avatars and haptics. This simulator is based on fast dynamics computation and constraint-based forces with friction without discretization. This simulator has been developed under an integrative framework called AMELIF, which provides components and generic interfaces for algorithm prototyping.

As mentioned in the previous section, simulations slow down as the number of contact points increases, thus penalizing haptic interaction. We will improve this aspect by optimizing the number of contact points (e.g. working with collision zones) and by computing the resulting contact forces per body rather than forces per contact point. The integrated simulator has been used to demonstrate haptic interaction with virtual avatars. The virtual avatar was enhanced with a low

level behavior, implemented as a compliant motion control law, to comply with external forces applied by an operator with a haptic device. In the future, higher level behaviors will be integrated, based on semantics that will be associated to patterns of interaction force between avatars and their environment. These behaviors will be based on an optimization of task sequences, allowing smoother and faster motions and thus more realistic haptic interaction with virtual avatars.

References

1. J. Hodgins and W. Wooten, "Animating human athletes," *Robotics Research*, pp. 356–367, 1998.
2. A. Shapiro, D. Chu, B. Allen, and P. Faloutsos, "The dynamic controller toolkit," in *The 2nd Annual ACM SIGGRAPH Sandbox Symposium on Videogames*, San Diego, CA, August 2007.
3. W. Son, K. Kim, and N. M. Amato, "A generalized framework for interactive dynamic simulation for multirigid bodies," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 34, no. 2, pp. 912–924, April 2004.
4. D. C. Ruspini and O. Khatib, "Collision/contact models for dynamics simulation and haptic interaction," in *Int. Symp. of Robotics Research*, 1999.
5. L. Sentis, "Synthesis and control of whole-body behaviors in humanoid systems," Ph.D. dissertation, Stanford University, July 2007.
6. H. Hirukawa, F.Kanehiro, and S.Kajita, "Openhrp: Open architecture humanoid robotics platform," in *Int. Symp. Robotics Research*, 2001.
7. C. Duriez, F. Dubois, A. Kheddar, and C. Andriot, "Realistic haptic rendering of interacting deformable objects in virtual environments," *IEEE Trans. on Vizualization and Computer Graphics*, vol. 12, no. 1, pp. 36–47, January-February 2006.
8. J.-R. Chardonnet, S. Miossec, A. Kheddar, H. Arisumi, H. Hirukawa, F. Pierrot, and K. Yokoi, "Dynamic simulator for humanoids using constraint-based method with static friction," in *IEEE Int. Conf. Robot. and Biomimetics*, December 2006.
9. R. Featherstone, *Robot dynamics algorithms*. Kluwer Academic Publishers, 1987.
10. B. Mirtich, "Impulse-based dynamic simulation of rigid body systems," Ph.D. dissertation, University of California at Berkeley, 1996.
11. F. S. Grassia, "Practical parameterization of rotations using the exponential map," *J. Graph. Tools*, vol. 3, no. 3, pp. 29–48, 1998.
12. D. H. Eberly, *Game Physics*. New York, NY, USA: Elsevier Science Inc., 2003.
13. A. L. Schwab, "Quaternions, finite rotation and euler parameters," 2002. [Online]. Available: http://audiophile.tam.cornell.edu/~als93/quaternion.pdf
14. O. Khatib, "A unified approach for motion and force control of robot manipulators: the operational space formulation," *IEEE J. Robot. and Autom.*, vol. RA-3, no. 1, pp. 43–53, 1987.
15. T. Liu and M. Y. Wang, "Computation of three-dimensional rigid-body dynamics with multiple unilateral contacts using time-stepping and gauss-seidel methods," *IEEE Trans. on Automation Science and Engineering*, vol. 2, no. 2, pp. 19–31, January 2005.