Two-handed Haptic Manipulation for CAD and VR Applications

Renaud Ott[1], Frédéric Vexo[2], Daniel Thalmann[3,4]

[1] EPFL-VRlab, renaud.ott@epfl.ch
[2] EPFL-VRlab, frederic.vexo@epfl.ch
[3] EPFL-VRlab and NTU-IMI, daniel.thalmann@epfl.ch

ABSTRACT

In this paper, we propose to increase realism and complexity of haptic applications for Virtual Reality and CAD applications. To achieve this goal, we use a Haptic Workstation which allows to acquire the posture and position of both hands, and to apply forces on the fingertips and wrists. We propose techniques to calibrate and improve the comfort of these kinds of devices in order to integrate them into Virtual Environments. However, a two-handed haptic device does not present only advantages. Indeed, It is much more complicated to compute forces on two hand models, than on a single point or fingertip. For this reason, we propose a framework to optimize this computation.

Keywords: two-handed haptics, hand model, collision detection, force feedback.
DOI: 10.3722/cadaps.2010.125-138

1 INTRODUCTION

To have only one hand able to manipulate objects shows how much the second hand is important. To understand this fact, we refer to Guiard's analysis of human skilled bimanual action [10].

· First, users can effortlessly move their hands relative to one another, but it requires a conscious effort to move a single hand relative to an abstract 3D space. This is the case when users have only one haptic device.
· Second, using both hands takes advantage of the user's existing skills. Most tasks we perform in our everyday lives involve using both hands in asymmetric roles, not one hand in isolation. This is even true for tasks like handwriting.

A key concept of Guiard's model is that the preferred and non-preferred hands (depending of left/right-handed) act together, but not in the same way. This asymmetric division of the tasks allows the hands to work with improved results, comparing to what either hand could achieve by itself. For example, Guiard reports that "the writing speed of adults is reduced by some 20% when instructions prevent the non-preferred hand from manipulating the page". One might also argue that using two hands to operate an interface only adds complexity and makes an interface harder. But there are many compound tasks that uses a single cognitive chunk.

© 2010 CAD Solutions, LLC

Several haptic research include the combined use of 3D graphical and haptic systems for interacting with at least two single-point interfaces. Barbagli et al. present in [2] a multifinger (2 fingers) haptic interface made with 2 Phantom.

Thanks to a two-handed haptic device, we can create Virtual Environments in which an object is graspable and dynamically animated by the laws of physics. When the object is seized by both hands, the haptic rendering engine realistically computes the forces on both exoskeletons. The efficiency of our rendering permits to apply these techniques to complex environments that have a significant number of objects. But the existing visual Virtual Environments are much more detailed than the ones seen in common haptic applications. In this paper, we aim at reducing this gap. One of the problems is that these quality environments usually do not include specific haptic object properties, such as mass or material. We thus propose a software allowing even non-professional to quickly and easily add this information to an environment. Our results show that this haptic rendering engine does not suffer from the large quantity of objects. They demonstrate that we have an efficient framework for integrating a two-handed haptic interface into a generic virtual environment. In the second part, we evaluate the potential of these kinds of Virtual Reality systems in more detail. While most applications can in theory take advantage of haptic devices, the practice shows that it is not always the case. Indeed, with experience, some metaphorical interaction paradigms remain more powerful than realistic ones. We thus present and study the integration of our two-handed haptic interface in CAD and VR applications. Evaluations show that depending on the application, it is not appropriate to reproduce reality: in teleoperation, for instance, simulating a virtual haptic steering wheel is less efficient than providing a force gesture interface. On the other hand, in virtual learning, the power of two-handed haptic manipulation is fully exploited and presents great advantages over standard techniques.

2 DEFINITION OF NEEDS

The role of the hands can be divided into two groups: feeling and interacting. In this paper, we will focus mainly on interaction, i.e. on the action that occurs when one or two hands have an effect upon one or more objects (and vice-versa). To enable the action of the hands on an object, we need:

- A system for acquiring the posture of the hands. Indeed, the posture is really important for manipulation, because it allows specific grasping of particular objects.
- A system for tracking the position and orientation of the hands in the 3D space. The orientation of the wrist is important and could even be considered as part of hand posture. If we cannot orient the hand, some objects are impossible to grasp.
- A workspace allowing to reach any position of the space close to the chest.

To enable the second part of the assertion, i.e. the action of an object on the hands (which is also related to feeling), we need:

- A system for simulating proprioception on the fingers and on the palm. Indeed, the force feedback prevents user's fingers to penetrate into the objects. Moreover, it provides added information about the nature of a virtual object, and finally it eases the grasping by offering the feeling of contact which is difficult to obtain only with visual/audio feedback.
- For the same reasons, we need a system for applying force feedback on the hands.

We believe that these are the minimal hardware requirements for performing realistic two-handed interaction. We use the Haptic Workstation™ (see Figure 1), which is composed of four usual devices of virtual reality:

- A pair of CyberGloves™used for acquiring hand posture.
- A pair of CyberGrasp™ used to add force feedback on each fingers.
- A pair of CyberForce™which is an exoskeleton used to convey force-feedback to both arms.
- A pair of CyberTrack™ encapsulated in the Cyber-Force™ device to get the position and the orientation of user hands.

© 2010 CAD Solutions, LLC

Fig. 1: The Haptic Workstation.

3 SYSTEM ARCHITECTURE

3.1 Software Organization

Our haptic software is composed of several components. Figure 2 presents the general organization of the modules, and shows how they exchange information together. In this section, we explain this diagram, and justify our choice about software organization.

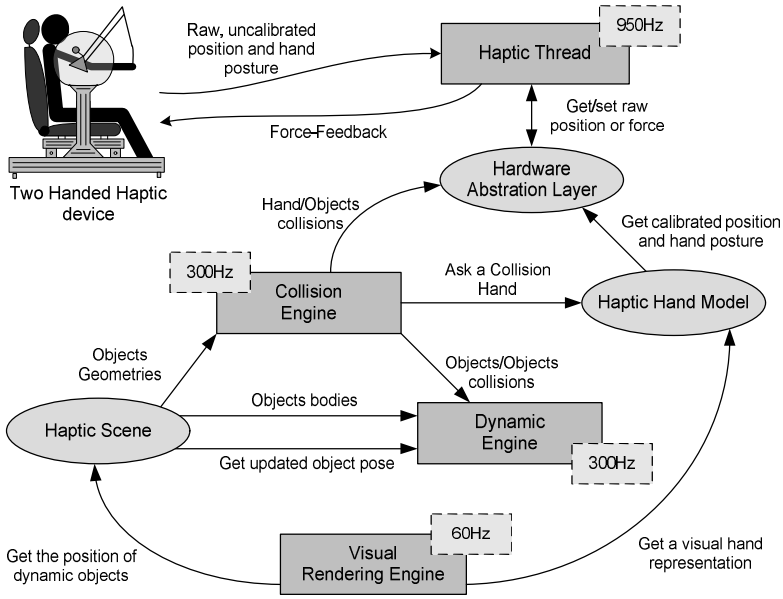

Fig. 2: General Organization of the Haptic software.

On this diagram, the modules are presented in rectangles: the Haptic Thread, the Collision Engine, the Dynamic Engine, and the Visualization Engine. Each module is in fact an infinite loop, running until the application ends. The ovals are representing the data structure that simplifies the communication between the modules. There is the Hardware Abstraction Layer, which guaranties the

© 2010 CAD Solutions, LLC

consistency of the data coming and going to the Haptic Thread. The Haptic Hand Model is one of the most important, as it manage the virtual interacting hands. It will be particularly detailed in section 4. And finally, the Haptic Scene groups the objects of the Virtual Environment that are touchable and manipulable. The arrows present the main messages and exchange of information between these components (modules and data structure). The Haptic Thread embeds the optimization for increasing the refresh rate of the data coming and going to the Haptic Workstation™. Basically, it gathers raw input data (hand posture and position) and copies it into a shared part of the memory. It gives also the force values to the output device. The forces result more or less from the computation of collisions between the hands and the objects. However, these modules do not have a sufficient refresh rate. Thus, we need to update force magnitude even if we do not have updated collision information. Consequently, the only solution is to make it the Haptic Thread. We deal with the force computation in section 4.3. Moreover, at this stage, if activated, there is the User Comfort Improvement function that can add a vertical force component on the CyberForce.

The Collision Engine and Dynamic Engine are working together. It seems difficult to parallelize them because one (the Dynamic Engine) needs the results of the other. They have many roles. First, they detect the collisions between the hands and the virtual objects. This is essential for the force computation and for user's feeling. They also animate the grasped or touched virtual objects, which is essential for interaction. Finally, they allow having a fully dynamic environment where each object is able to act on another, which increases realism and possibly immersion.

The last module is the Visual Rendering Engine. Its role is of course to provide a view of the Virtual Environment. Haptics without visual feedback is often used in the context of object recognition. But, as mentioned in [9], the discrimination of shapes or curvature using a whole-hand (multi-fingered) kinesthetic haptic device is as difficult as with single-point haptic device. Moreover, a two-handed haptic device seems adapted to virtual manipulation or training. This suggests that embedding a visual rendering system into a haptic framework presents advantages.

These modules are not running at the same refresh rate. They are separated into three threads. The first one contains only the Haptic Thread and has a fixed refresh rate around 950 Hz. The second one embeds the Collision and Dynamic Engine. Its refresh rate can be easily adjusted according to the complexity of the Virtual Environment. And finally, the Visual Rendering Engine is in fact not running in a thread, but in the main program which is instantiating the other threads.

3.2 The Haptic Node

As presented on Figure 3, a visual scene graph contains a root, and then different kinds of objects organized in a tree. For instance, only the MVMesh objects should be considered because they can potentially represent a tangible object. On the other side, we have scene graph that contains collision and dynamic objects that do not have a truly visual representation. The role of the haptic node is to provide a link between these two scene graphs. Such link is needed to ease the maintenance of coherence between the two Scene Graphs. For example, if an actor is moved by the Dynamic Engine, the corresponding visual object should be positioned accordingly. For this reason, a Haptic node is a class that contains pointers on a visual mesh and on its dynamic alter ego. This class provides also useful methods described below. The Haptic Scene is simply a collection of nodes, organized in a simple list.

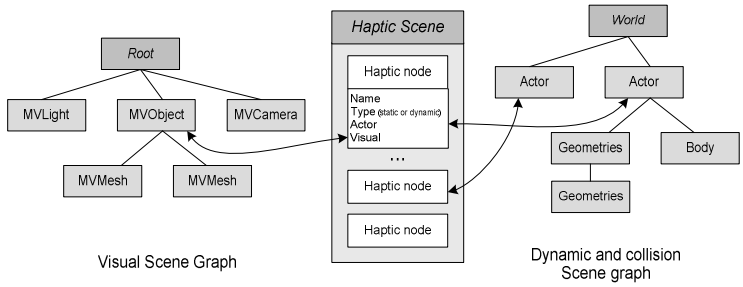

Fig. 3: The Haptic Node links visual meshes and dynamic actors.

© 2010 CAD Solutions, LLC

3.3 Collision Detection

Collision detection is a broad topic dealing with a quite simple problem. It consists in determining if two or more objects are intersecting. Moreover, we may also want to know when and where the collision happens in order to properly adapt the force feedback response.

As shown in the previous subsection, directly testing the geometry of two objects for collision against each other could be expensive. To minimize this cost, object bounding volumes could be tested for overlap before the geometry intersection test. The choice of the bounding volume characteristics is important. As shown in Figure 4, it is a trade-off between intersection test cost and tight fitting of the volume. A large bounding volume could result in a false positive collision and thus, reduce the performances. Our method uses axis-aligned bounding boxes because they are easy to compute, easy to intersect and require less memory. In case of false positive collisions, it does not strongly reduce the performances because we made the choice to approximate the objects with simple primitives.

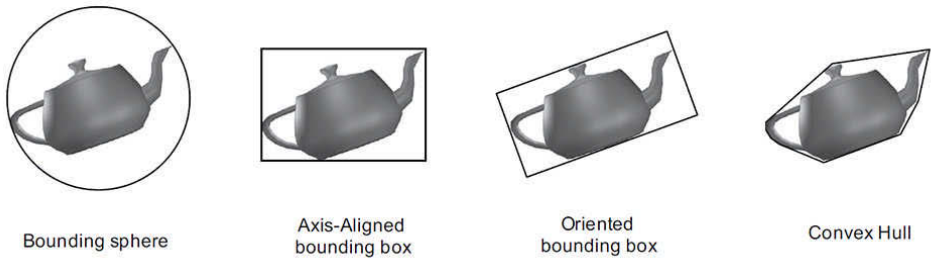

Fig. 4: Types of bounding volumes.

Second optimization is really simple. It comes directly from the fact that, in a normal realistic life scene, only few objects are moving. Thus, every pair of "non-moving" objects that are not colliding at a time t could not be colliding at a time $t + \Delta t$. Thus, during the animation process, we mark each object that does not move. And then when we parse the pairs of objects, we do not even call the static inference test if they are both disabled. This optimization is very efficient because it replaces the static inference test, which can be complex by two simple bit tests. However, this method does not reduce the number of pairs of object tested.

The third optimization is related to the fact that two relatively small and distant objects could not intersect. So it is not useful to test these kinds of pairs. Two main approaches exist for classifying the objects. First approach consists in organizing the objects in a Binary Space Partition Tree (BSP-trees) [11]. This approach is efficient when the scene can be preprocessed. But, in our case, it is not possible because a scene can possibly contain a lot of dynamic objects. We use a Spatial Partitioning. It consists in dividing the space into regular regions and to test only the objects belonging to the same region.

3.4 Dynamic Animation

People are also used to diverse reactions of the objects when submitted to conditions toward the violation of the non-penetration law. Some objects bounce on each other, some deform, some others break; they generally produce sound when colliding, sometimes heat. In Virtual Reality, modeling objects reproducing such phenomena is a big issue on increasing the realism of a scene. Moreover in a context of realistic manipulation, it is absolutely necessary to provide such mechanism.

In our system, we use Forward and Inverse Dynamics these two kinds at the same time.

A Virtual Environment is composed of moving objects defined by 3 main parameters:

- the mass m, the center of gravity G, and the inertia tensor I, a 3×3 matrix, which represents the angular moment of inertia.

© 2010 CAD Solutions, LLC

- Forces acting on a body tend to move it (displacement and orientation). To calculate the resulting change of position and orientation, we use the Newton-Euler laws to :

$\sum F = m\ddot{x}_G$, and $\sum T = I_A \ddot{\alpha}$ where F and T are the forces and torque applied on the rigid body and x and α are position and orientation of the rigid body.

To solve the equations 3.1, we can distinguish the approximate methods or the analytic methods. One of the most frequently used in computer animation is the Euler Integrator which is explained in [1]. In [12], Otaduy and Lin present a good review of these techniques when applied to haptic rendering.

4 HAPTIC HAND MODEL

In the two previous sections, we presented a way to physically animate the virtual objects. The main advantage of using this method is that when we exert external forces on virtual objects, we do not need to provide specific behaviors (using scripts). But it means also that the virtual hands should be modeled with respect to the same approach.

In this section, we present a powerful model of virtual hands in the context of haptic manipulation. Several approaches have been primarily tested based on different methodologies. In this subsection, we present the two techniques that were implemented and tested.

4.1 Direct Mapping

It consists in positioning a virtual interaction point at the same position than the device itself (hard link). This is the most trivial solution, and seems also to be the easier to implement (but we will show later that it is finally not the case). The calibrated position, rotation and posture of the hands are directly mapped onto the virtual hand. The hand model is thus composed with collision geometries and with a skeletal animated visual mesh that uses skinning techniques for increasing realism. The idea behind this technique is to compute collisions between the hands and the virtual objects and to apply a realistic force-feedback for avoiding the collisions.

As already mentioned, the collision detector returns if and where two objects collides. This second information is used to compute the force feedback. For example, if the fingertip enters into a table, two points are returned:

- The deepest point of the phalanx into the table (P_D in Figure 5)
- Another point P_n laying on the table surface that is the closest as possible of P_D

The vector $P_D P_0$ determines the direction of the force feedback, and its norm is used to compute the magnitude of the reaction force. This method gives the impression that an elastic with a zero rest length is link between P_n and the fingertip resulting in a force $F = -k\Delta$. As for every virtual elastics, it is possible to change the spring constant k. In order to avoid the resonance, due to the fact that the user's fingertip is oscillating between a "inside-object"/"outside-object" state, it is also possible to add a damping factor d for smoothing the force magnitude: $F = -(k\Delta + d\Delta')$. In fact, damping is critically important, due to its role in counteracting the energy generation from errors introduced by sensing and discrete time.

The first difficulty with this method is to deal with user's movement. Figure 5 presents three particular cases. The first one 5a, shows the resulting force feedback when the user displaces his hand on the surface of a spherical object. We can observe that the force seems to be smooth and the variation of direction gives the impression that the object is a sphere. However, the two other examples presenting a hand moving on the surface of a table (see figure 5b and 5c) are showing force continuity breaks at particular points P_i. They are due to the fact that the collision detector returns the closest point to P_i laying on the object surface. We can imagine many solutions to avoid this specific problem, but in fact, each solution will present an inconsistency in a particular context.

© 2010 CAD Solutions, LLC

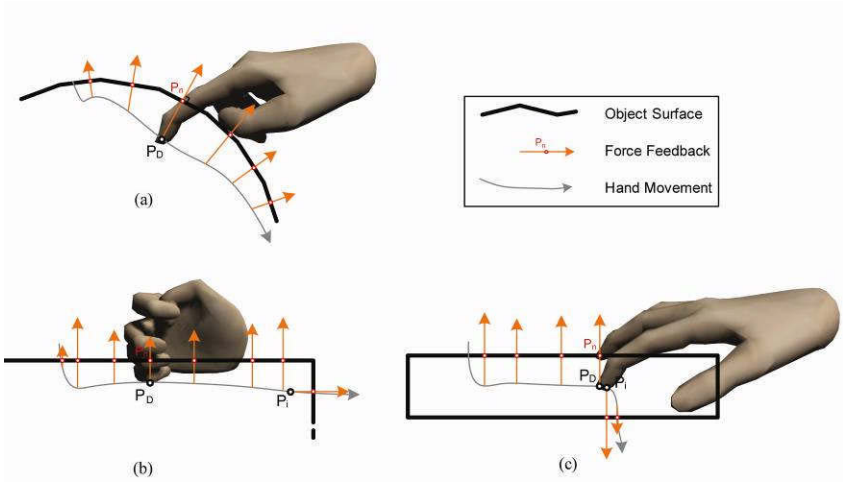

Fig. 5: Force feedback computation based on penetration distance.

The main source of problem with this solution comes from the Haptic Workstation itself. The break sensation in force continuity increases with the penetration depth (because force magnitude is greater), and our haptic device is not powerful enough to counterbalance the weight of an arm resting on a table (or barely). Thus, penetration distances can become great and the perception of a break increases.

Another problem to mention which is related to the Haptic Workstation™ appears when the force feedback could be applied on several actuators at the same time. Figure 6 presents an example when it happens. User's fingers are in contact with a cube. When the fingers penetrate into the cube, a resistive force has to be applied. But, it raises a question: Where should we apply the force feedback? On the CyberGrasp (at finger level), or on the CyberForce (at wrist level)?

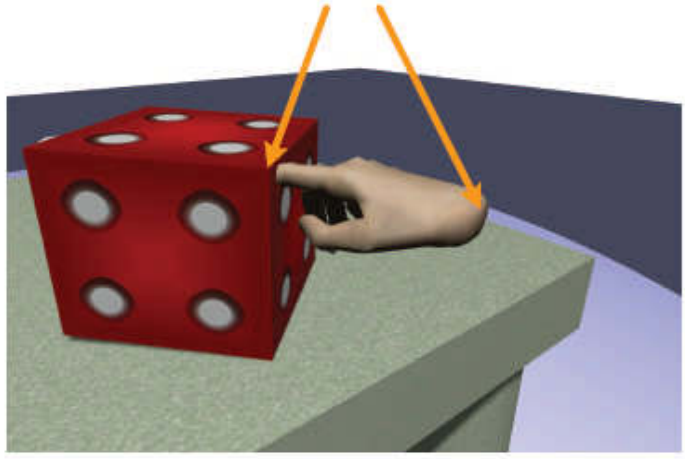

Fig. 6: Where should we apply the force feedback?

In fact, it is impossible to answer to this question by examining a single frame at a single time. The reason is that it depends of the muscular tension of the wrist and fingers. If the user locks his wrist and that finger muscles are resting, the force feedback should be mainly sent to the CyberGrasp™ (and opposingly of course). If every muscles rest, force feedback should be applied on the two devices at the same time. The same problem appears when two hands are grasping the same dynamic object. In this case, it is not trivial to compute the force magnitude on the CyberForce™, because each hand

© 2010 CAD Solutions, LLC

transmit force through the virtual object to the other hand. As our Haptic Workstation™ is a passive device, we only know the position of the hands, and not the forces applied by the hands on the system. This is the reason why it is impossible to answer to the question using a single state of simulation. However, by examining many consecutive frames, we can get an approximation of hand's speed and acceleration. Then, because of the relation $\Sigma F = ma$, it is at least possible to approximate the force divided by the mass.

Finally, with the direct mapping method, there are visual inconsistencies due to the interpenetration of hands and object resulting in a break in presence [14]. It does not satisfy the law of non-penetration. Here again, the limited force of the Haptic Workstation™ is the source of the problem. Moreover, it does not ease the computation of the force feedback, and some specific examples shows that it could introduce discontinuities in force direction. We present another solution in the next paragraph. We choose it for avoiding these two problems.

4.2 Mass Spring Hand

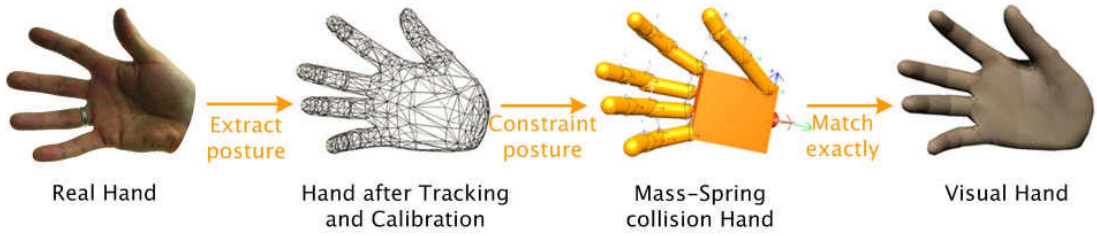

Fig. 7: The three hand models.

This technique consists in using a proxy-based method [15]. A proxy is weakly linked to the position of the device, i.e. the Haptic Workstation™. Basically, it is based on three hand models (see figure 7):

- The Tracked Hand (shown in wireframe on the figure). It is in fact the virtual hand skeleton created after calibration. It is supposed to be the exact representation of the real hand position orientation and posture into the Virtual Environment. It is of course not the case, but we assume that the matching is correct.
- The Proxy (Volumes shown on the figure), which is a mass-spring-damper system that has the shape of the hands. Each phalanx and the palm is composed of a collision geometry, and has also dynamic properties. These elements are linked together with motorized joints parameterized with springs and damping coefficient.
- The Visual Hand. This is the hand that is visually rendered and the only one visible. It is easily created using the visual rendering engine [13].

For each hand, the idea is to couple a proxy hand to the tracked hand using a set of virtual linear and angular springs. As a result of the dynamic simulation, the spring-hand tends to follow the tracked-hand. The visual hand displayed to the user reflects the spring-hand configuration.

This approach follows the "proxy" method proposed for the Phantom (a single-point interaction device), extending it to the whole hand. It has been firstly described by Borst et al. [3]: they applied it to the CyberGrasp™ force feedback device which is also a component of the Haptic Workstation™.

It solves the problem of interpenetration between the visual hands and the environment because the spring-hands adapt their pose on the surfaces of the objects. Spring-Hands have basically two constraints:

- A soft constraint which is to match the best as possible the configuration of the tracked hands. This is achieved by applying specific force and torques on the linear and angular springs (Figure 8a).
- A hard constraint which is to avoid penetration within virtual objects. This is achieved simply by activating the collision detection between the phalanxes/palm rigid bodies and the objects of the Virtual Environment (Figure 8b).

© 2010 CAD Solutions, LLC

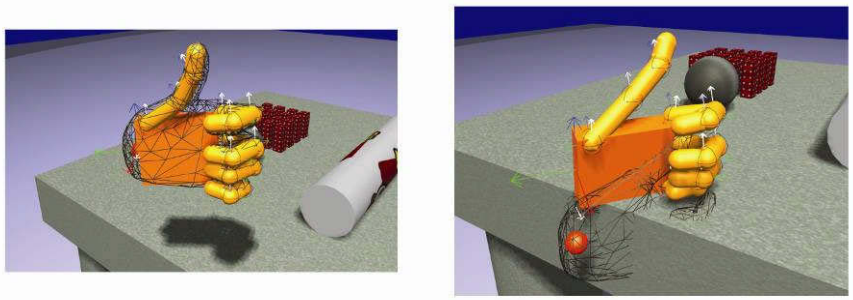

Fig. 8: (left) soft constraint, (right): hard constraint.

The spring-hands are created using dynamic rigid bodies. To match the posture of the hands, we use two kinds of geometrical primitives. For each hand, the phalanxes are approximated with capsules and the palm with a flat box. Then, these rigid bodies are linked together using spherical joints (3 angular DOF). Some of them could be replaced by hinge joints (only 1 angular DOF) though we did not notice any problem with this method. By linking the geometries together with those joints, we get an articulated hand. Finally, an angular spring is attached to each joint. These springs will give a torque to the phalanxes according to the angles of the tracked hand. As a result, the fingers of the spring-hand will follow the fingers of the tracked-hand. The torque τ applied by the spring on the articulation made of an angular joint is computed as follow:

$$\tau = k(\alpha_s - \alpha_t) - d(\omega_s - \omega_t) \tag{4.2}$$

where k, d are the spring and damping of the angular joint, α_t , α_s are respectively the angles of the tracked hand and of the spring-damper hand, ω_s, ω_t are angular velocities of the phalanxes on the tracked hand, and on the spring-damper hand.

The spring constant defines how stiff the torque is applied to the fingers. An high value will provide a more reactive behavior but will suffer from vibrations. The damping constant allows the torque to be reduced according to the respective angular velocities of the joints. It avoids the vibrations of the fingers but provides a smoother reaction.

By the same manner, to allow the translation and the rotation of the hands in the virtual world, one linear spring (3 DOF) and one angular spring (3 DOF) are attached to the base of each hand. Therefore, when the user moves the wrists, these two springs respectively apply a force and a torque, allowing the base of the spring-hand to follow the base of the tracked-hand. The linear spring provides a linear force F as follow:

$$F = k_t(p_s - p_t) - b_t(v_s - v_t) \tag{4.3}$$

where k_t , b_t are the spring and damping constants, p_t , p_s are positions of the tracked and spring hands, v_s, v_t are velocities of the spring and tracked models.

The torque that is applied by the angular spring is more complex as it depends on the direction of the hands. As a result the spring torque vector and the damping torque vector usually do not point in the same direction.

It is possible to directly attach linear and rotational springs to the joints. The spring constant k and the damping constant b are both parameterizable. It is preferable to adopt this solution than computing the forces and torques and apply them manually to the phalanxes. The reason is that it

© 2010 CAD Solutions, LLC

may be quickly instable for stiff spring and forces due to a limitation of the numerical integration used to advance time from one time step to the next. Internally, the joints use a drive constraint in which the springs are implicitly integrated within the solver. This is the best way to model stiff behavior for stable simulation. We followed this recommendation as the spring constants are necessarily very high to ensure that the spring-hands follow the user's moves as close as possible.

In fact, it implies that the forces and the torques are not applied directly as presented previously. Instead, we give angular orders to the drives (motors) of the joints and the Physics Engine integrates these values to internally produce the forces and the torques. However, the model presented here is still perfectly valid in our case and represents the real behavior of our simulation.

4.3 Force-feedback Computation

The mass spring hand model provides an elegant way to compute the force feedback. When a collision with the hand occurs, it is indeed easily possible to compute the distance between the position of the tracked hand and the position of the mass-spring system. Then, we send these two positions to the Hardware Abstraction Layer, especially in the CyberForce and CyberGrasp data structure. Then, the Haptic Thread get these data at a high refresh rate, and is able to compute a force for making the tracked hand to match the position of the mass-spring hand. The force magnitude is of course proportional to the distance between the two models. In Figure 9, we can see the difference between the two positions of the tracked and mass spring hands. The resulting forces are represented by the arrows. The advantage of the method is that the forces are refreshed according to the lastly updated positions, near 1 KHz, whatever the speed of the collision engine. The disadvantage is that during the real hand movement we can miss some collisions. This occurs only with small objects: for a hand speed of 1ms−1, and at 300 Hz, the detector checks collisions every 3 mm. This is the concept of force feedback computation.

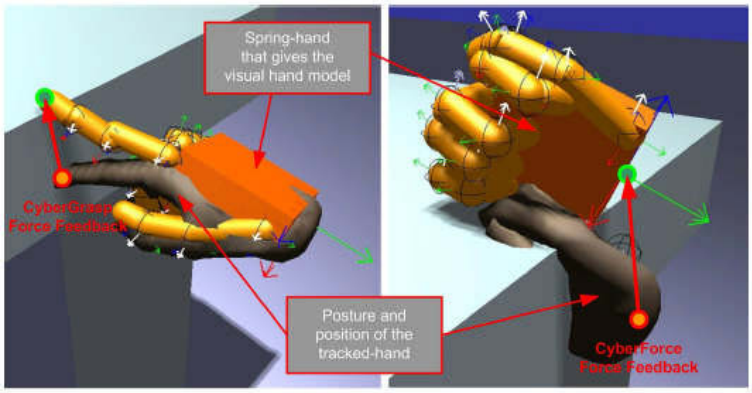

Fig. 9: The computation of the forces for the CyberForceTM, and CyberGrasp™.

The CyberGrasp is an unilateral device. Thus some collisions could not be managed. To solve this problem, when we compute the difference vector between the tracked and mass spring hands, we also get the vector that is normal to the nail (the nail's normal vector is aligned with the string of the Cybergrasp). Then, if the scalar product of these two vectors (after normalization) is positive, it means that the CyberGrasp should produce force.

4.4 Benefits for Manipulation using Two-hands

In this section, we will show that the mass-spring hands present many advantages in the context of two-handed haptic interaction.

The first advantage concerns the visual feedback. We have seen that when using this system, the visual hand do not penetrate anymore into the virtual objects. However, it induces a problem often called visual-proprioceptive discrepancy. This problem occurs when the virtual hand is for example stopped by a table. It is then possible that the real hand is not at the same position that the virtual one

© 2010 CAD Solutions, LLC

(Position discrepancy). It is also possible that the real hand moves but the visual one does not (Motion discrepancy). This is of course something that the user could notice. The question is then to know which anomaly is noticed the first. In [4], Burns et al. investigate users detection threshold for visual interpenetration and visual-proprioceptive discrepancy. Conclusions show that we are much more sensitive to interpenetrations. We can thus conclude that the spring-hand model present an advantage over the direct mapping technique in terms of presence.

The second benefit that we have presented concerns the computation of the force feedback. In the case of interaction through a single-point device using for example a Phantom the force feedback computation is greatly improved due to the proxy. As mentioned earlier, it removes the strange behavior that usually happens when we compute forces according to the penetration distance.

Finally, this method has also a third advantage. It implicitly computes the forces applied by the user on the virtual object, as illustrated in Figure 10.

These improvements strongly increase the realism and thus the immersion of the user into the virtual environment. We try to simulate the best as possible the reality and its physic. This implies a better learning curve of the manipulation system, and requires less adaptation from the user. The efficiency of many applications is improved because users could focus on the simulated task only, and not on the way to manage the simulation itself.

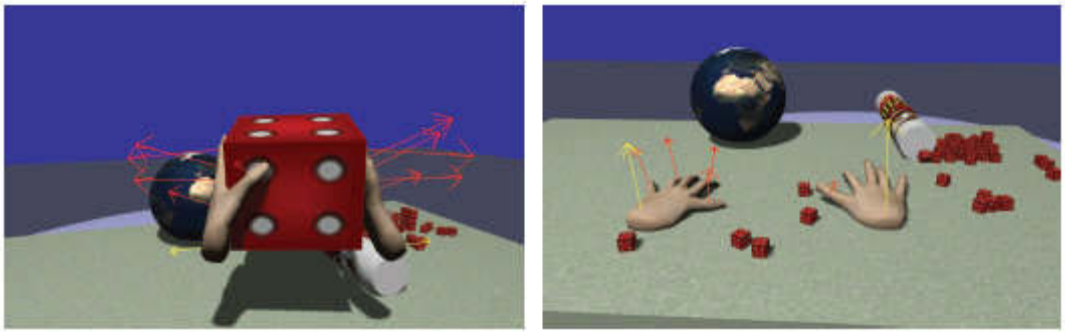

Fig. 10: The implicit force estimation process done by the mass-spring system.

4.5 Haptic Scene Creator

When creating various applications that have an extensive use of haptic features, the programmers (creators of the application) often face an important problem: the lack of haptic information of the 3D models. The common editing tools for creating Virtual Environments do not always provide the adequate functionalities for adding this kind of information.

In this context, it appears to be necessary to give the opportunity to the haptic programmer to augment the visual Virtual Environment using an authoring tool, the Haptic Scene Creator. The complexity of a visual mesh requires its decomposition in low level primitives in order to speed up the collision detection. Obviously, this decomposition cannot be done automatically, because it strongly depends on the targeted application and desired level of detail. For this purpose, the Haptic Scene Creator application is a graphical tool that supports the haptic augmentation of any Virtual Environment loadable in our visual rendering engine. It is very intuitive, simple to learn and provides useful automated features to simplify and accelerate the task.

The designer should be able to select visual objects in order to “augment” them. Once selected, the user can manipulate information which is relevant for our dynamic engine and collision detection system.

The dynamic engine needs information related to the “mass” of an object. We remind that the “mass”, or body, includes the real mass (or density), the center of gravity and the inertia tensor. Of course the mass/density is easily parameterizable in the editor, but we preferred to hide the inertia tensor to the designer to keep the tool intuitive and accessible. In fact, the center of gravity and the tensor matrix can be computed knowing the size, the position and the density of the geometries

© 2010 CAD Solutions, LLC

linked to the object. Unless some special effects are desired, it provides a reasonable approximation. By automating these computations, the designer will also be able to focus on higher level tasks.

The collision detection system needs to know the shape of an object. Of course, the shape is clearly defined in its visual description. However, using this information only is by far too complex for the computation of the collisions [8]. Thus, objects should be approximated with simple geometric primitives (boxes, spheres, capsules). Later on we add more complex geometries such as convex meshes and penetration maps. These geometries are not displayed during the simulation, but it is necessary to visualize them in the editor. Moreover, for each collision geometry linked to an objet, its material properties must be parameterizable. These parameters are the static and dynamic friction coefficients and the coefficient of restitution (bounciness).

As shown previously, the Haptic Scene Creator includes many features aiming at simplifying the task of a Haptic application designer. To evaluate it, a designer augmented a visual Virtual Environment. The test scene is a quite complex. It represents a four-rooms house (kitchen, bathroom, bed, office and living-room), and contains 398 visual nodes. It is presented in Figure 11. The main goal is to use the Haptic Scene Creator to augment every touchable object with a good level of approximation in order to manipulate objects easily. This task took almost three hours, time to create 612 geometries for 167 static objects and 114 dynamic objects. The advanced functionalities (copy-pasting, vertex selection, PCA) were of course extensively used. We do not have performed the same procedure on the full scene without using the advanced functionalities (and neither using a simple text editor to write by hand the XML file). But, to give an idea of the improvements, augmenting a chair without advanced functionalities took five (tedious!) minutes, whereas it took around 1 minute using the PCA.

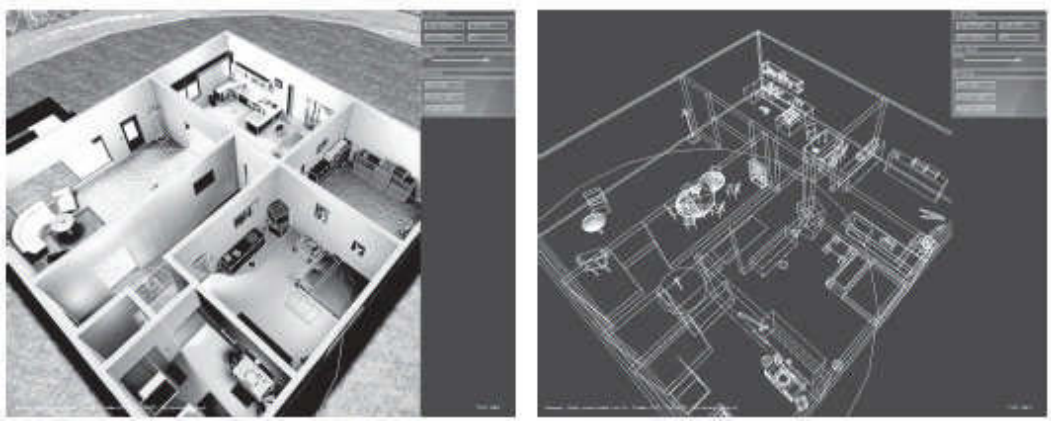

Fig. 11: The augmentation of the 3D House.

4.6 Interaction with Complex Virtual Environment

The PC running the simulation is a based on an Intel Quad-Core processor (2.4 GHz) and a NVIDIA 8800GTX graphics card. When loading the simulation, we remark that between 75% and 100% of the microprocessor is occupied, which is as expected. At the beginning during few second, the program lags. The reason is that every dynamic object is in the “moving state”, and that most of them are colliding. Then, once they stop, the refresh rate is stable. The Haptic Thread is running around 900 Hz. The Physics Engine has been limited to 300 Hz, and the display is also refreshed around 300 Hz. The visual result is presented in Figure 12.

© 2010 CAD Solutions, LLC

Fig. 12: Manipulation and Interaction with a haptic house.

When immersed into a large scale Virtual Environment, the user is not able to reach every object. As (s)he is seated in the Haptic Workstation™, it seems difficult to propose a realistic displacement method. Thus, a metaphor should be used to let him/her move aground the Virtual Environment. In [5], authors presented an evaluation of 3 different techniques to achieve the interaction with objects that appear bigger than the workspace of the haptic device. We used a method similar to the "bubble" technique described in [6]. When the user moves his arms near the workspace limits, he enters into an area that displace the virtual camera: arms to the front moves the camera forward, arms to one side turns the camera. In addition, force feedback is applied to tell the user that he is entering to this area. This convenient method is intuitive, and the displacement is really controllable. However, we remark that a calibration has to be done to adapt the method to the user's morphology. Indeed, the far limit of the workspace is not at the same position for everyone.

The force feedback is efficient with most of the static objects. We were surprised that even the thin tables produce a convincing resistive force due to the fact that the hand does not go through. The main limitation comes from objets augmented with pMaps. Even the static ones produce unexpected behaviors. It also occurs occasionally with dynamic objects that use convex meshes. Despite these issues, the feeling of being there (presence) seems increased, mainly because of the 3D interaction capabilities of the two-handed haptic device.

5 CONCLUSION

We believe that two-handed Haptic feedback is a wide topic which is still under exploration. When Guiard analyzed human bimanual action, he proved that the vast majority of human manual acts involve two hands acting in complementary roles [10]. However, in Haptics, the majority of undergone experiments involve only one hand, or even one finger. In this thesis, we made preliminary studies on two-handed haptic feedback focusing, on the one hand, on the realistic rendering for manipulation, and on the other hand, on its applications.

As mentioned earlier in this conclusion, we believe that two-handed Haptics has a promising future. The intuitiveness and efficiency resulting from the use of such technologies for the manipulation of Virtual Environments allow us to think that two-handed devices will become the new standard of Haptics in few years. We are conscious that, in 2008, such devices have a very high cost.

© 2010 CAD Solutions, LLC

However, situation was the same for the Phantom R 10 years ago, and today, some devices that imitate it are commercially available for less than 300 dollars.

However, if we had to choose only one improvement of the Haptic Workstation™, it would be the addition of a device stimulating the palm. This is one of the most common remarks that we had from the dozens of users of the device. It is true that the grasping state usually provokes a pressure on a part of the palm. We believe that a study on the importance of the palm pressure feeling could provide interesting results.

Finally, in the same state of mind, the adjunction of tactile devices on the fingertips would greatly increase the discrimination of shapes. It is really challenging to combine these two haptic perceptions, but it has already be done on a smaller scale with success.

ACKNOWLEDGMENT

This work was supported by the Swiss National Research Foundation.

REFERENCES

[1] Baraff D.; Witkin A.: Physically based modeling: Principles and practice, Online Siggraph '97 Course notes. ACM Press, 1997.
[2] Barbagli F.; Salisbury J.K.; Devengenzo R.: Enabling multi-finger, multi-hand virtualized grasping, Proc. ICRA'03. IEEE International Conference on Robotics and Automation, 1(1), 2003, 809–815.
[3] Borst C. W.; Indugula A.P.: Realistic virtual grasping, Proc. 2005 IEEE Conference on Virtual Reality (VR'05), IEEE Computer Society, 2005, 91-98.
[4] Burns E.; Razzaque S.; Panter A.T.; Whitton M.C.; McCallus M.R.; Brooks J.F.P.: The hand is more easily fooled than the eye: Users are more sensitive to visual interpenetration than to visual proprioceptive discrepancy, Presence: Teleoperators and Virtual Environments, 15(1), 2006, 1–15.
[5] Buxton W. A. S.: Chunking and phrasing and the design of human-computer dialogues, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, 494-499.
[6] Dominjon L.; Lécuyer A.; Burkhardt J.; Andrade-Barroso G.; Richir S.: The "bubble" technique: Interacting with large virtual environments using haptic devices with limited workspace, Proc. First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WORLDHAPTICS'05).
[7] Dominjon L.; Richir S.; Lécuyer A.; Burkhardt J.: Haptic hybrid rotations: Overcoming hardware angular limitations of force-feedback devices, Proc. IEEE conference on Virtual Reality (VR '06), 2006, 164-174.
[8] Ericson C.: Real-Time Collision Detection. Morgan Kaufmann, 2005.
[9] Frisoli A.; Solazzi M.; Salsedo F.; Bergamasco M.: A Fingertip haptic display for improving curvature discrimination, Presence: Teleoperators and Virtual Environments, 17(6), 2008, 550–561.
[10] Guiard Y.: Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model, Journal of Motor Behavior, 19, 1987, 486-517.
[11] Naylor B.; Amanatides J.; Thibault W.: Merging bsp trees yields polyhedral set operations, Proc. SIGGRAPH, 17th annual conference on Computer graphics and interactive techniques, ACM NY, USA, 1990, 115-124.
[12] Otaduy M. A.; Lin M. C.: High Fidelity Haptic Rendering, chapter 6 DOF Haptic Rendering Methodologies. Morgan and Claypool Publishers, 2006, 23-34.
[13] Peternier A.; Thalmann D.; Vexo F.: Mental vision: a computer graphics teaching platform. In proceedings of the 2006 Edutainment Conference, 2006, 223–232.
[14] Slater M.; Steed A.: A virtual presence counter, Presence: Teleoperators and Virtual Environments, 9(5), 2000, 413–434.
[15] Zilles C. B.; Salisbury J. K.: A constraint-based god-object method for haptic display, Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, 3, 1995, 146–151.

© 2010 CAD Solutions, LLC