6

Effective Haptic Rendering Method for Complex Interactions

Josune Hernantes, Iñaki Díaz, Diego Borro and Jorge Juan Gil
CEIT and TECNUN (University of Navarra)
Spain

1. Introduction

The development of haptic technology is allowing the introduction of Virtual Reality systems as teaching and working tools into many fields such as engineering (Howard & Vance, 2007; Savall et al., 2002) or surgery (Basdogan et al., 2004; Li & Liu, 2006).
Haptic devices allow users to interact with a certain environment, either remote or virtual, through the sense of touch, considerably enhancing interactivity. A haptic device is a mechanism that allows users to control the movements of a virtual tool or a real robot and receive tactile and kinesthetic information from the working environment (Fig. 1).
The usability of these systems is conditioned by the quality of the haptic feedback applied to the user. Technologically, the computation of appropriate and realistic haptic stimuli continues to be a complicated issue. The human sensory-motor system demands a fast update rate (at least 1 kHz) for the haptic stimuli applied to the user in order to avoid instabilities in

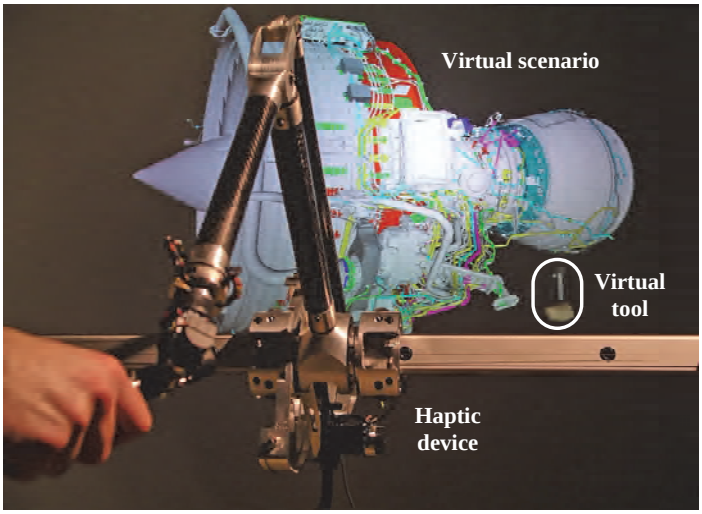


Fig. 1. Haptic interaction with a virtual environment

the system and to present rigid objects with reasonable stiffness (Shimoga, 1992). However, this update rate is often difficult to reach by haptic rendering methods, especially when working in complex environments. One possible solution is to reduce the computational cost of calculating the haptic response by decreasing the accuracy of the method. However, this can result in the emergence of discontinuities in the response. This leads to find a trade-off between the accuracy of the method, which guarantees a smooth and stable haptic response, and the computational cost.

This chapter describes a haptic rendering method to properly compute interacting forces and torques in complex environments, ensuring improved feedback by seeking a compromise between continuity and computational cost. In addition, the proposed method pays particular attention to provide users with comfortable interaction. The method is valid for applications in which the virtual environment is composed of rigid and static objects, excluding deformable objects.

The remainder of this chapter is organized as follows: Section 2 presents an overview of the related research on the area. Afterwards, Section 3 describes the haptic rendering method proposed by the authors, describing in detail all algorithms necessary to render appropriate and stable forces and torques to the user. The proposed method is then evaluated in Section 4 within two different virtual scenarios simulating common collisions during aeronautic maintainability tasks. Aeronautic virtual mock-ups have been selected for algorithms testing due to their high interaction complexity. Finally, conclusions and future directions are drawn in Section 5.

2. Related research

The process of computing and generating forces in response to user interactions with virtual objects is known as haptic rendering (Salisbury et al., 1995). The application of haptic rendering algorithms to complex contact scenarios becomes a challenging issue, due to the inherent cost of collision detection that induces slow force updates. The haptic display of virtual objects has been an active area of research throughout the last decade. Previous research in haptic rendering can be mainly classified within two groups: penalty and constraint-based methods.

When it comes to penalty methods, collision response is computed as a function of object separation or penetration depth. McNeely et al. (1999) proposed point-voxel sampling, a discretized approximation technique for contact queries that generates points on moving objects and voxels on static geometry. This approximation algorithm offers run-time performance independent of the environment's input size by sampling the object geometry at a resolution that the given processor can handle. Renz et al. (2001) adapted this method with several modifications for smoother and more stable haptic feedback. In another research project, Gregory et al. (2000) presented a 6-DOF haptic rendering system that combined collision detection based on convex decomposition of polygonal models, predictive estimation of penetration depth and force and torque interpolation. Kim et al. (2003) attempted to increase the stability of force feedback by using contact clustering, but their algorithm for contact queries suffers from the same computational complexity.

Otaduy and Lin (2003) have presented a sensation preserving simplification technique for 6-DOF haptic rendering of complex polygonal models by adaptively selecting contact resolutions. Later, they have also presented a modular algorithm for 6-DOF haptic

rendering, that provides transparent manipulation of rigid models with a high polygon count (Otaduy & Lin, 2006).
Unlike penalty methods, constraint-based methods do not use the interpenetration between rigid objects to calculate collision response. These methods use virtual coupling techniques (Colgate et al., 1995) and restrict the movement of virtual objects on the surface of obstacles.
Zilles and Salisbury (1995) proposed a constraint-based method for 3-DOF haptic rendering of generic polygonal objects. They introduced the "god-object", an idealized representation of the position of the haptic device that is constrained on the surface of obstacles. At each time step, the location of the god-object minimizes the distance to the haptic device and the difference between the two positions provides the force direction. Ruspini et al. (1997) extended this approach by replacing the god-object with a small sphere as well as proposing methods to smooth the object surface and add friction. Later, Ortega (2006) extended the 3-DOF constraint-based method of Zilles and Salisbury by employing a 6-DOF god-object.
The different haptic rendering methods described above have contributed extensively to a better representation of contact events between virtual objects. However, haptic interactions with multiple contacts, which also include geometrical discontinuities, have not yet been adequately accomplished computing unrealistic or unstable haptic feedback in these cases. These type of situations are very common in real scenarios and therefore it is necessary to compute properly a stable haptic response in order to improve the usability of these systems. The proposed haptic rendering method overcomes limitations from previous approaches in this type of collisions.

3. Proposed haptic rendering method

The haptic rendering method outlined in this chapter computes the force and torque that result when a collision between two type of objects occurs: a virtual tool (mobile object) manipulated by the user of the haptic device and any object in the simulation (static object).
Three main modules can be identified in the haptic rendering method proposed (Fig. 2): collision detection, collision response and control module.
The complete haptic rendering sequence could be described as follows: firstly, the control module acquires the position ($\boldsymbol{U}_h$) and orientation ($\boldsymbol{R}_h$) of the haptic device and sends it to the collision detection module. With this information, the module checks for collisions between the mobile object and the static environment. If no collisions occur, it waits for new information from the control module. Otherwise, when a collision event occurs, the contact information of both static and mobile objects ($\mathbf{C}_s$, $\mathbf{C}_m$) is sent to the collision response module which calculates the interaction force and torque. This haptic feedback approximates the contact force and torque that would arise during contact between real objects ($\boldsymbol{F}_r$, $\boldsymbol{T}_r$). Finally, the collision response module sends this information to the control module, which applies it to the haptic device ($\boldsymbol{F}_h$, $\boldsymbol{T}_h$), maintaining a stable system behaviour.
A more complete description of each module can be found in the following sections.

3.1 Collision detection

The collision detection method presented in this chapter can handle non-convex objects without modifying the original mesh. A technique based on a spatial partition (voxels) has been chosen. Hierarchical methods like octrees have also been tested since they require less

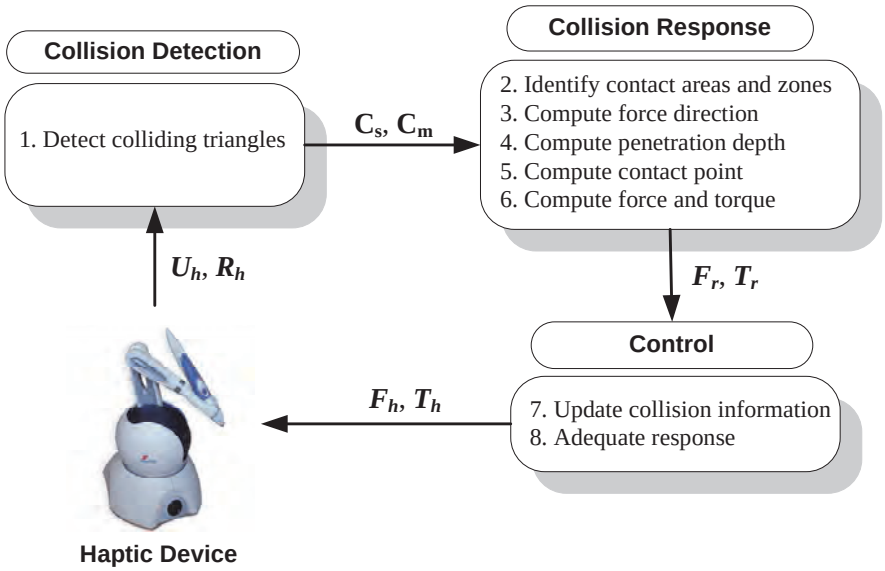


Fig. 2. Diagram of haptic rendering algorithm

memory storage. However, their computation time is higher than that needed for direct access voxel techniques.

It is well-known that algorithms based on voxel techniques have several disadvantages, such as high memory storage requirements and the selection of voxel size. According to previous experiments (Borro et al., 2004), hashing techniques can solve the first problem by reducing memory storage up to 60 % without performance loss, and the choice of an optimal voxel size can be solved by means of an analytical solution based on the algorithm cost function.

The method, in a pre-process, computes a voxel partition from the virtual scene and assigns each triangle of the static environment to its corresponding voxel. This voxel partition is used only for the static object. In addition, each voxel will have a flag identifying it as internal (V_{int}), external (V_{ext}) or boundary (V_{bnd}). The last ones contain the triangles that define the surface of the static object.

Next, at runtime, the partition model is used to detect the set of voxels in collision with the mobile object and to carry out interference checks between triangles.

Fig. 3a shows an example of colliding static triangles (***CST***) and colliding mobile triangles (***CMT***) detected by the method in a virtual collision of a tool with an obstacle. As it can be seen colliding triangles do not provide enough information to delimit the volume that defines the intersection between the objects. Therefore, unlike other existing methods in the literature, the proposed algorithm detects additional triangles in order to calculate the intersection volume correctly (Fig. 3b). The union of the colliding triangles and these additional triangles are referred to as *contact triangles*.

The additional triangles can be classified within three groups: internal mobile (***IMT***), boundary mobile (***BMT***) and boundary static (***BST***). With regard to the mobile object, (***IMT***) are those in contact with $\boldsymbol{V}_{int}$ whereas (***BMT***) are those in contact with $\boldsymbol{V}_{bnd}$, but that do not intersect with static triangles (***CST***) (Fig. 4).

Regarding the static object, ***BST*** set of triangles are those in contact with V_{bnd} (Fig. 5), but that do not intersect with mobile triangles. These triangles are usually in the centre of the contact area surrounded by colliding static triangles.

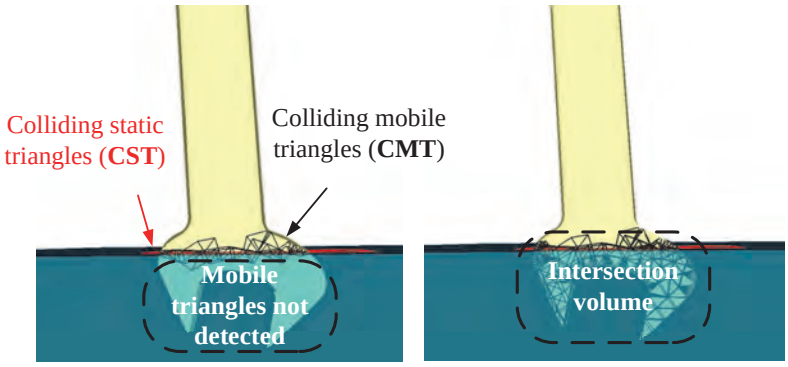


Fig. 3. Colliding triangles detected (a) and additional triangles that define the intersection volume (b)

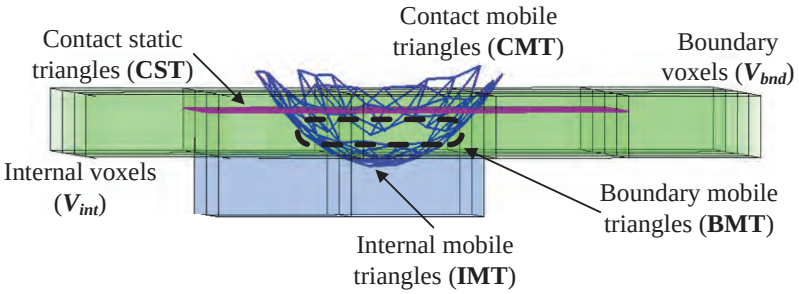


Fig. 4. Internal and boundary mobile triangles

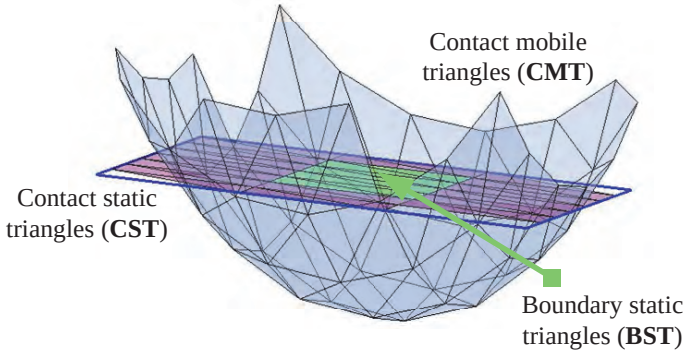


Fig. 5. Static boundary triangles

At the end, the collision detection module yields two sets of contact triangles, one belonging to the static object ($\mathbf{C}_s$) and the other to the mobile object ($\mathbf{C}_m$).

$$\boldsymbol{C}_s = \{\boldsymbol{CST} \bigcup \boldsymbol{BST}\} \tag{1}$$

$$\boldsymbol{C}_m = \{\boldsymbol{CMT} \bigcup \boldsymbol{BMT} \bigcup \boldsymbol{IMT}\} \tag{2}$$

3.2 Collision response

The algorithm proposed follows the well-known penalty methods based on elastic model, thus the haptic response that users feel as a consequence of a collision in the virtual environment is determined by a direction and a penetration value. Both factors have substantial influence on user perception.

The force ($\boldsymbol{F}_r$) and torque ($\boldsymbol{T}_r$) are calculated as follows:

$$\begin{aligned} \boldsymbol{F}_r &= K d \boldsymbol{n} \\ \boldsymbol{T}_r &= (\boldsymbol{cp} - \boldsymbol{gc}) \times \boldsymbol{F}_r \end{aligned} \tag{3}$$

where K is virtual stiffness, d is penetration depth, $\boldsymbol{n}$ force direction, $\boldsymbol{cp}$ the contact point and $\boldsymbol{gc}$ the centre of mass of the virtual tool.

In complex scenarios, it is common to encounter multiple contacts. The lists of triangles ($\mathbf{C}_s$ and $\mathbf{C}_m$) obtained by the collision detection module do not give a priory information about the number of different contacts. Therefore, the method divides these sets of triangles into different contact areas considering their spatial proximity. The forces and torques of each area are then added up to compute the net force and torque.

A certain amount of static triangles form a contact area when they are adjacent. In other words, they share at least one edge. Once the different contact areas have been calculated from $\mathbf{C}_s$, it is necessary to identify the mobile triangles in contact that are associated with those areas. To facilitate this process, a sphere is created in each contact area that covers the bounding box defined by its triangles. Finally, each triangle of $\mathbf{C}_m$ is associated with the contact area depending on the sphere which contains it. Fig. 6 shows an example of the division of contact areas.

The difficulty in calculating an appropriate and stable haptic response increases when the geometry has sharp edges, since haptic instabilities often appear due to abrupt force direction or penetration depth changes. In order to detect these cases, each contact area is sub-divided into different contact zones that provide information about the nature of the geometry in collision.

The collision response module subdivides a collision area into different contact zones taking surface C^1 discontinuities into account. When two triangles share an edge and the angle between their normal vectors is lower than a fixed value, the edge is designated as "smooth". Triangles in a contact zone must be interconnected and all the shared edges must be smooth. There will be as many contact zones as necessary to satisfy the condition of smooth connectivity (Fig. 7).

Once the contact areas and zones have been detected, the method computes the contact normal vector (force direction) of each area ($\mathbf{n_c}$). If the contact area has a unique contact zone, the contact normal is computed as the normalized sum of all normal vectors of the static triangles of that zone. Otherwise, when a contact area has two or more contact zones, the

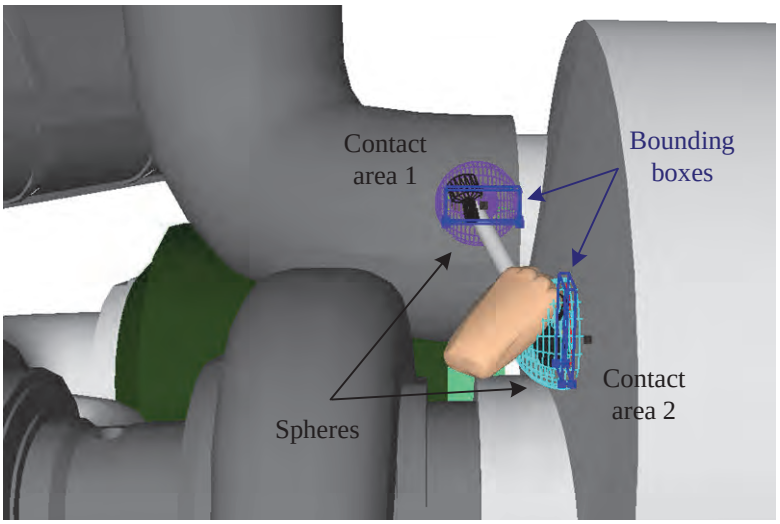


Fig. 6. Example of two contact areas

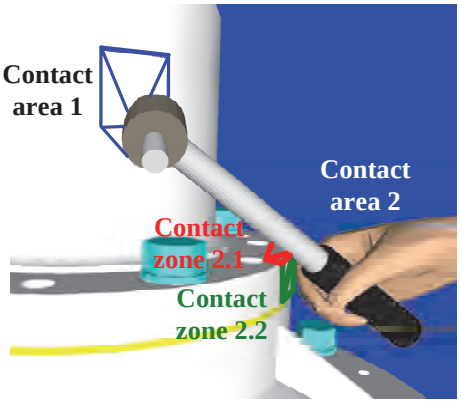


Fig. 7. Example of two contact areas. The second one has two contact zones

contact has occurred in an area of the static object that is not a continuous surface. In this case, static triangles do not provide enough information to obtain a suitable force direction. Therefore, $\mathbf{n_c}$ is computed as the normalized sum of all normal vectors of the mobile triangles of that area. This solution enables smoother direction transitions when interacting with sharp edges.

After computing the force direction in each area, the following step is the calculation of the minimum distance required to separate the two objects, known as penetration depth (d). For that purpose, the method samples the volume of intersection measuring heights throughout this volume to determine the penetration between two objects. These heights are determined by tracing a ray from the centroid of each static triangle in $\mathbf{n_c}$ direction (Fig. 8). If this ray intersects with a mobile triangle (Möller & Trumbore, 1997), the height is defined as the

distance between the centroid and the intersection point. The final value of penetration for each area is computed as an average of all the computed heights.

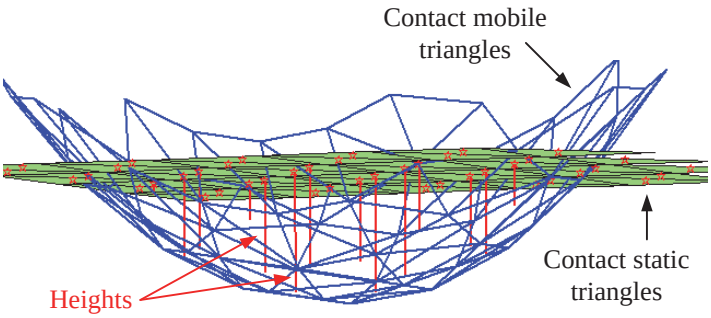


Fig. 8. Rays traced to compute penetration depth

The last parameter necessary to compute collision torque is the contact point (**cp**). Selecting an inappropriate point such as the most penetrating one for each contact area might lead to non-continuous changes in the haptic feedback (Hasegawa & Sato, 2004). To avoid this effect, the proposed solution is to choose a representative contact point for each contact area. This point is calculated as the average of the midpoints of the boundary voxels associated with each contact area.

3.3 Control module

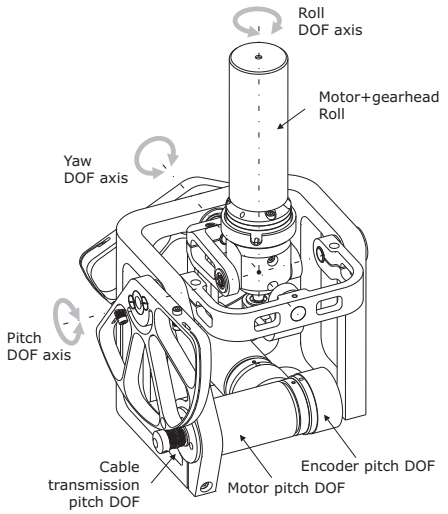
The control module receives the ideal interaction force and torque calculated by the collision response method, adapts them to the device's capabilities and applies them to the user at a 1 kHz sampling rate. Since the control loop runs faster than the collision module, several strategies must be implemented in order to avoid abrupt changes in contact force and torque when collision information is not available (Savall et al. (2002)).

There are two main problems that should be taken into account. The first one is the delay that exists in the collision-related information. This information, calculated by the response module, is valid for a previous user position, but not the actual one. The second problem resides in the existence of some control loops without collision-related information, since the response module is slower than the control loop. In order to deal with these problems, a strategy based on intermediate representations (Adachi et al. (1995)) is implemented.

Let $\boldsymbol{M}_h$=($\boldsymbol{F}_h$, $\boldsymbol{T}_h$) be the six-dimensional vector of the force and torque that we want to apply to the user and $\boldsymbol{X}_h$=($\boldsymbol{U}_h$, $\boldsymbol{R}_h$) be the six-dimensional configuration of the haptic device ($\boldsymbol{U}_h$ represents the position, and $\boldsymbol{R}_h$ the axis angle representation of the rotation). First, the control method updates the force and torque ($\boldsymbol{M}_r$) computed by the response module at a previous sampling period j, to the current one i (K represents virtual stiffness):

$$\boldsymbol{M}_h(i) = \boldsymbol{M}_r(i-j) - K(\boldsymbol{U}_h(i) - \boldsymbol{U}_h(i-j)) \tag{4}$$

These forces and torques might vary quite abruptly if they are applied to the user every time they are updated by the collision response method. Therefore, the control module restores



Degrees of freedom	
Sensing input	3 DOF (Pitch-Yaw-Roll)
Feedback output	3 DOF (Pitch-Yaw-Roll)
Workspace	
Pitch-Yaw	±70 º
Roll	Unlimited
Continuous output level	
Pitch-Yaw	0.7 Nm
Roll	0.2 Nm
Peak output level	
Pitch-Yaw	1.6 Nm
Roll	0.9 Nm
Actuators	
Pitch-Yaw	DC MaxonRE35, 90 W + Cable Transmissions 13.5:1
Roll	DC MaxonRE-max21, 6 W + Maxon Planetary GP22C 29:1
Encoders	
Pitch-Yaw	Quantum Devices QD145, 5000 ppr
Roll	MaxonMR TypeM, 512 ppr

Fig. 9. 3-DOF torque feedback device used for the experiments

them in n subsequent sampling periods:

$$\Delta \boldsymbol{M}_h(i) = \frac{\boldsymbol{M}_r(i-j) - K(\boldsymbol{U}_h(i) - \boldsymbol{U}_h(i-j)) - \boldsymbol{M}_h(i-1)}{n}$$
$$\boldsymbol{M}_h(i) = \boldsymbol{M}_h(i-1) + \Delta \boldsymbol{M}_h(i) \tag{5}$$

In the next sampling period $(i+1)$, the method also takes the new movements performed by the user into account:

$$\boldsymbol{M}_h(i+1) = \boldsymbol{M}_h(i) + \Delta \boldsymbol{M}_h(i) - K(\boldsymbol{U}_h(i+1) - \boldsymbol{U}_h(i)) \tag{6}$$

This will continue until the collision response method updates the collision-related information. To determine n, the optimal value should approximate the number of control sampling periods that the collision module needs to compute the response. Since this number can vary significantly depending on the number of triangles in collision, a conservative number can be set. However, if n is very high, the method may excessively filter the signal. Moreover, n can have a fixed value during the entire task or can be modified by means of the average collision response delay in previous sampling periods.

4. Implementation and results

Two different virtual scenarios have been designed to test the effectiveness and stability of the proposed haptic rendering algorithm on complex interactions. A 3-DOF torque feedback device has been used for the experiments. The mechanism was designed and built at CEIT and inspired by past research (Angerilli et al., 2001). Fig. 9 shows the device and its main specifications.

The system is controlled by a dSPACE DS1104 board that reads encoder information, processes the haptic control loop and outputs torque commands to the motors. Graphic rendering and

collision detection are performed on a PC running the Windows XP operating system with a Pentium Dual Core 6600, 2GB memory and an NVIDIA GeForce 8600 GT.

4.1 Analysis of multiple contacts and geometrical discontinuities

The first scenario is composed of two parallelepipeds, one central cylinder and a virtual tool similar to a clamp (see top of Fig. 10). The virtual scenario consists of 25,000 triangles while the clamp is composed of 1,500 triangles. The aim is to analyze the behavior of the method when multiple contacts and geometrical discontinuities are involved. For that purpose, a sequence of different collisions has been simulated:

1. Collision with the central cylinder in which two contact areas are involved.
2. Collision with the corners of the two parallelepipeds (C^1 discontinuities).
3. Collision with four contacts simultaneously, combining the previous cases.

Fig. 10 shows the torque feedback computed in all three axis by the collision response method during the sequence. The figure also shows the number of contact triangles detected and the computational cost in each frame.
Notice in the figure that the torque feedback applied to the user during collision is quite smooth and does not offer abrupt changes or discontinuities that may degrade the user's perception of contact. The last figure shows that the collision response method computes the haptic feedback at an average of 2.5 ms, which is not far from the optimal computation time (1 ms) necessary for a realisic haptic experience. This allows the control module to compute the real forces and torques at 1 kHz using a low number of n transitions (described in Section 3.3), and thereby maintaining a stable system behaviour. Specifically, for this experiment, the number of transitions n for the control algorithm was set to 5.

4.2 Simulation of a disassembly task

The task designed for the second experiment is similar to the extraction of a clamp from a pipe, which frequently appears in engine disassembly tasks in aeronautics maintenance. Once the clamp is unfastened, the exit path is established by following the spatial trajectory laid out by the pipe itself. Along said path, the curves and bends of the pipe and other obstacles force the clamp to rotate in space. Therefore, in order to accomplish this task properly, it is important that the haptic feedback restored to the user is realistic. The virtual engine mock-up is defined by 100,000 triangles while the clamp is composed of 2,000 triangles.
The torque feedback device used in the previous experiment does not allow any translation. Thus, to be able to displace the clamp within the virtual environment, it is necessary to provide the mechanism with a translational DOF. For that purpose, a linear actuator designed and built at CEIT (Savall et al., 2008) is used. The displacement along this linear DOF is mapped into a displacement of the clamp along the axial direction of the pipe, and forces along this DOF are also displayed when collisions occur. Fig. 11 shows both the designed virtual environment and the haptic device used for the experiments.
A virtual path from right to left along the route of extraction was performed. During this process, different types of collisions occurred between the clamp and the environment. Fig. 12 shows an example of the main possible stages during the extraction:

1. Initial position.

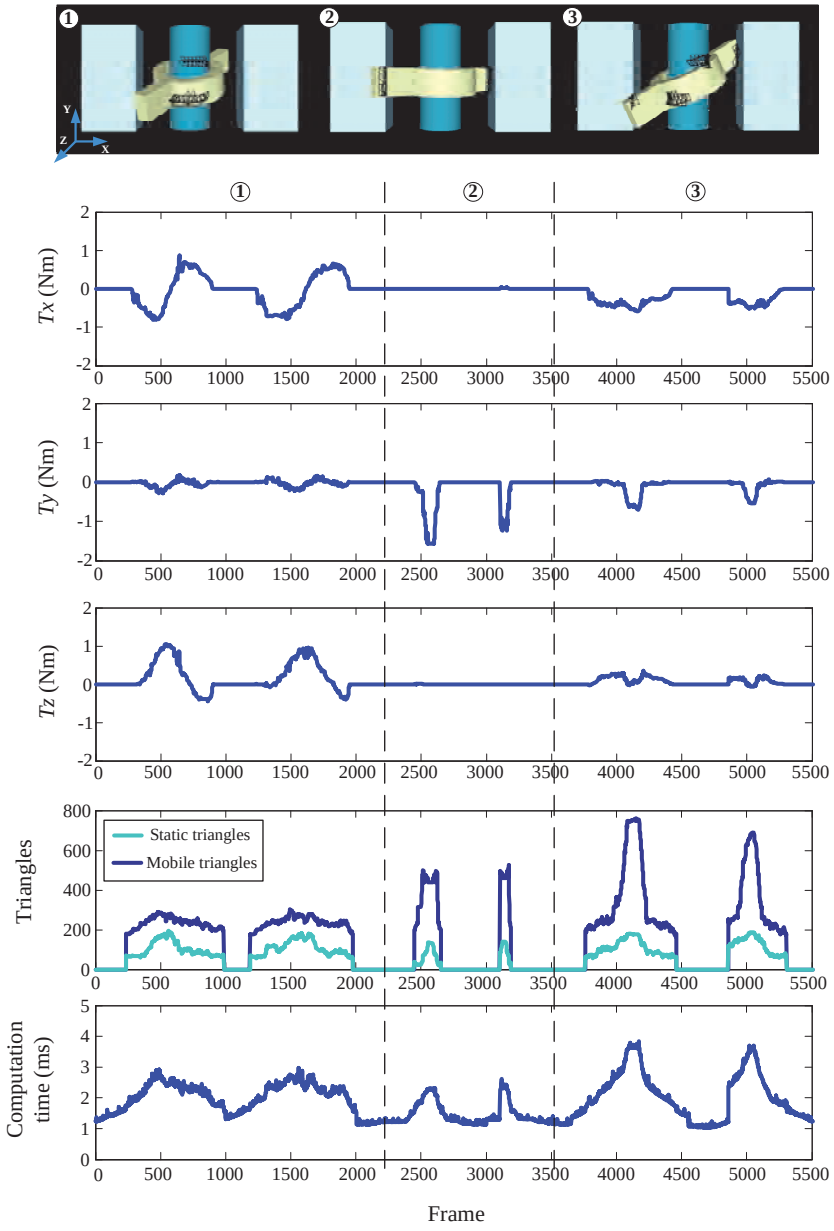


Fig. 10. Performance of our approach in multiple contacts and geometrical discontinuities

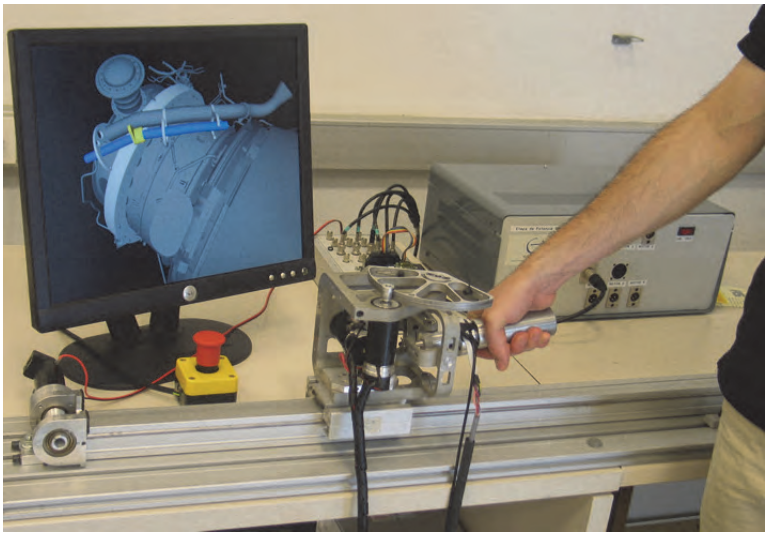


Fig. 11. Haptic system used for the experiment

2. Collision of the clamp with the first obstacle. In this case, the linear actuator restores a force in the X axis to avoid translational movement.

3. The clamp is inside the obstacle and, when rotating along the X axis, it collides with the upper part of the obstacle.

4. Collision of the clamp with the pipe when rotating along the Z axis.

5. Collision of the clamp with the pipe resulting in complex multi-axis torque.

6. Final position. The clamp is disassembled.

Fig. 12 shows the torque applied to the user in each axis (T_{r_x}, T_{r_y} and T_{r_z}) and the force exerted by the linear actuator (F_{r_x}) computed by the collision response method. It also indicates the number of triangles in collision, in addition to the computational cost at each frame. Notice that in this figure data are shown from right to left according to the movement of the tool.

Haptic feedback obtained gives a realistic perception of collision events and allows to perform the task properly. Unlike the previous scenario, designed to study the behavior of the method in situations with multiple collisions, in this case the aim is to simulate a real task. For this reason, although the complexity of the environment is higher, the number of simultaneous contacts decreases because the user corrects trajectory when a collision is detected.

Fig. 13 is an augmentation of the third collision stage of Fig. 12 (frames 3450–3650) for torque feedback in the x axis, with and without applying the control algorithms. It can be seen that torque computed by the collision response is smooth and avoids abrupt changes. In addition, control algorithms improve the continuity of the feedback signal and apply it to the user at a sampling rate of 1 kHz. As in the previous example, in this case the number of transitions for the control algorithms is also 5.

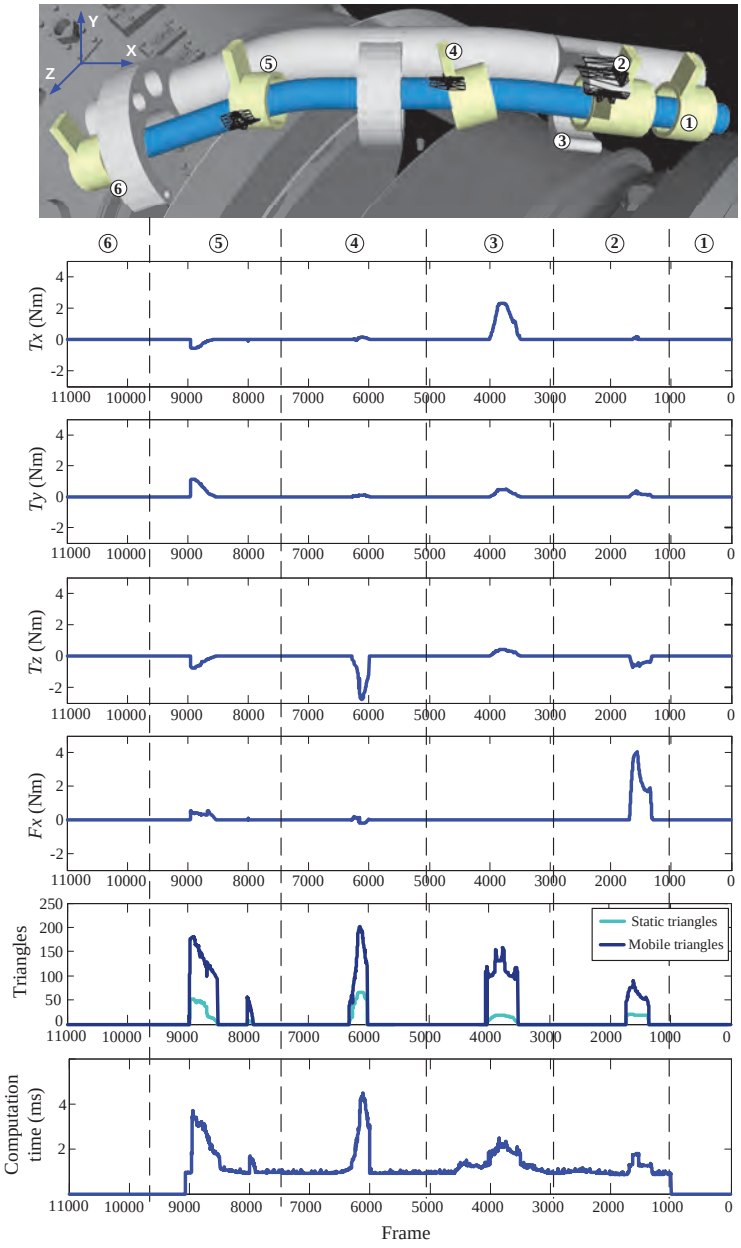


Fig. 12. Extraction of a clamp from a pipe

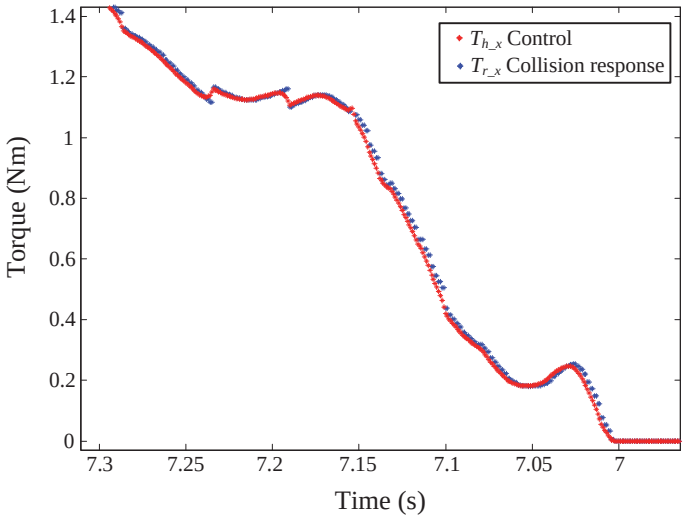


Fig. 13. An augmentation of the third collision stage for torque feedback in the X axis, with and without applying the control algorithms

5. Conclusions and future research

The real-time computation of the forces and torques in a virtual environment is a complicated task but a key point for the effectiveness of haptic systems. It is known that non-realistic or inappropriate haptic feedback has negative effects on the usability and leads to frustration when manipulating haptic systems. Therefore, it is very important to guarantee smooth and realistic haptic feedback.

This chapter outlines a haptic rendering method that computes a proper haptic response in complex environments. It ensures improved feedback by seeking a compromise between continuity and computational cost. The method avoids abrupt changes in the haptic force direction and magnitude, thereby improving the overall stability of the haptic system.

In order to validate the proposed method, two different scenarios containing complex collision examples, such as multiple contacts and geometrical discontinuities, have been used. The yielded results validate the applicability of the method in these types of interactions.

In terms of future research, the authors plan to analyze the performance of the method from a perceptual perspective carrying out studies of human factors to improve the responsiveness. The authors also hope that the research included in this chapter will provide a better understanding of the many phenomena that challenge the development of improved haptic rendering methods able to display adequate force and torque feedback while preserving stability, and thereby improve performance of current haptic interfaces.

6. Appendix: List of notation

- x, y, z : Reference displacement axis
- $\mathbf{U}_h$: User displacement of the haptic handle
- $\mathbf{R}_h$: User rotation of the haptic handle in angle-axis notation

- $\mathbf{X}_h$: User displacement and rotation of the haptic handle
- $\mathbf{F}_r$: Collision force computed by the collision response method
- $\mathbf{T}_r$: Collision torque computed by the collision response method
- $\mathbf{M}_r$: Collision force and torque computed by the collision response method
- $\mathbf{F}_h$: Force feedback applied to the user
- $\mathbf{T}_h$: Torque feedback applied to the user
- $\mathbf{M}_h$: Force and torque feedback applied to the user
- $\mathbf{V}_{int}$: Internal voxels
- $\mathbf{V}_{ext}$: External voxels
- $\mathbf{V}_{bnd}$: Boundary voxels
- $\mathbf{C}_s$: List of static triangles in collision
- $\mathbf{C}_m$: List of mobile triangles in collision
- **cp** : Collision contact point
- **gc** : Centre of mass of the virtual tool
- K : Virtual object stiffness
- d : Penetration of the mobile tool within a static object
- **n** : Force direction
- $\mathbf{n}_c$: Force direction of each contact ares
- n : Number of transitions for the control algorithm

7. References

Adachi, Y., Kumano, T. & Ogino, K. (1995). Intermediate representation for stiff virtual objects, *IEEE Virtual Reality Annual International Symposium* pp. 203–210.

Angerilli, M., Frisoli, A., Salsedo, F., Marcheschi, S. & Bergamasco, M. (2001). Haptic simulation of an automotive manual gearshift, *10th IEEE International Workshop on Robot and Human Interactive Communication*, pp. 170–175.

Basdogan, C., De, S., Kim, J., Muniyandi, M., Kim, H. & Srinivasan, M. (2004). Haptics in minimally invasive surgical simulation and training, *IEEE Computer Graphics and Applications* 24(2): 56–64.

Borro, D., García-Alonso, A. & Matey, L. (2004). Approximation of optimal voxel size for collision detection in maintainability simulations within massive virtual environments, *Computer Graphics Forum* 23(1): 13–23.

Colgate, J., Stanley, M. & Brown, J. (1995). Issues in the haptic display of tool use, *IEEE International Conference on Intelligent Robot and Systems*, Vol. 3, Pittsburgh, PA, USA, pp. 140–145.

Gregory, A., Mascarenhas, A., Ehmann, S., Lin, M. & Manocha, D. (2000). Six degree-of-freedom haptic display of polygonal models, *IEEE Conference on Visualization*, Salt Lake City, Utah, United States, pp. 139–146.

Hasegawa, S. & Sato, M. (2004). Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects, *Computer Graphics Forum* 23(3): 529–538.

Howard, B. M. & Vance, J. M. (2007). Desktop haptic virtual assembly using physically based modelling, *Virtual Reality* 11(4): 207–215.

Kim, Y. J., Otaduy, M. A., Lin, M. C. & Manocha, D. (2003). Six-degree-of-freedom haptic rendering using incremental and localized computations, *Presence: Teleoperators and Virtual Environments* 12(3): 277–295.

Li, M. & Liu, Y.-H. (2006). Haptic modeling and experimental validation for interactive endodontic simulation, *IEEE International Conference on Robotics and Automation*, Orlando, Florida, USA, pp. 3292–3297.

McNeely, W. A., Puterbaugh, K. D. & Troy, J. J. (1999). Six degree-of-freedom haptic rendering using voxel sampling, *ACM SIGGRAPH - Computer Graphics*, Los Angeles, California, USA, pp. 401–408.

Möller, T. & Trumbore, B. (1997). Fast, minimum storage ray-triangle intersection, *Journal of Graphics Tools (JGT)* 2(1): 21–28.

Ortega, M., Redon, S. & Coquillart, S. (2006). A six degree-of-freedom god-object method for haptic display of rigid bodies, *IEEE Virtual Reality Conference*, pp. 191–198.

Otaduy, M. A. & Lin, M. (2006). A modular haptic rendering algorithm for stable and transparent 6-dof manipulation, *IEEE Transactions on Robotics* 22(4): 751–762.

Otaduy, M. A. & Lin, M. C. (2003). Sensation preserving simplification for haptic rendering, *ACM Transactions on Graphics* 22(3): 543–553.

Renz, M., Preusche, C., Pötke, M., Kriegel, H.-P. & Hirzinger, G. (2001). Stable haptic interaction with virtual environments using an adapted voxmap-pointshell algorithm, *Eurohaptics*, Birmingham, UK, pp. 149–154.

Ruspini, D. C., Kolarov, K. & Khatib, O. (1997). Haptic interaction in virtual environments, *IEEE International Conference on Intelligent Robots and Systems*, Vol. 1, Grenoble, France, pp. 128–133.

Salisbury, J. K., Brock, D. L., Massie, T., Swarup, N. & Zilles, C. (1995). Haptic rendering: Programming touch interaction with virtual objects, *ACM Symposium on Interactive 3D Graphics*, Monterey, California, USA, pp. 123–130.

Savall, J., Borro, D., Gil, J. J. & Matey, L. (2002). Description of a haptic system for virtual maintainability in aeronautics, *IEEE International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, pp. 2887–2892.

Savall, J., Martín, J. & Avello, A. (2008). High performance linear cable transmission, *Journal of Mechanical Design* 130(6).

Shimoga, K. B. (1992). Finger force and touch feedback issues in dextrous telemanipulation, *Fourth Annual Conference on Intelligent Robotic Systems for Space Exploration*, pp. 159–178.

Zilles, C. & Salisbury, J. (1995). A constraint-based god-object method for haptic display, *International Conference on Intelligent Robots and Systems, Human Robot Interaction, and Cooperative Robots*, Vol. 3, pp. 146–151.