Stable Haptic Rendering for Physics Engines Using Inter-Process Communication and Remote Virtual Coupling

Xue-Jian He, Kup-Sze Choi
Centre for Integrative Digital Health, School of Nursing
The Hong Kong Polytechnic University
Hong Kong, PR

***Abstract*— Availability of physics engines has significantly reduced the effort required to develop interactive applications concerning the simulation of physical world. However, it becomes a problem when kinesthetic feedback is needed in the applications since the incorporation of haptic rendering is non-trivial, where fast haptic data update is demanded for stable rendering. In the regard, a framework for integrating haptic rendering into physics simulation engines is proposed. It mediates the update-rate disparity between haptic rendering and physics simulation engine by means of inter-process communication and remote virtual coupling, which fully decouples haptic rendering from complex physical simulation. Experimental results demonstrate that this framework can guarantee fast haptic rendering at 1k Hz even the physical simulation system operates at very low update rate. The remote virtual coupling algorithm shows better performance than the interpolation based methods in terms of stability and robustness.**

Keywords—haptic rendering; physics engine; inter-process communication; virtual coupling.

I. INTRODUCTION

To convey more immersive and realistic experiences in the cyberspace, many virtual reality (VR) systems have integrated kinesthetic feedback with the use of haptic interfaces. It has also been demonstrated that haptic sensation can greatly increase the effectiveness of the VR based simulation systems [1, 2]. With the availability of commercial haptic interfaces, software toolkits, and commercial haptics-enabled applications, haptics-enabled simulation technology is undergoing rapid and exciting growth [3]. However, realistic rendering of haptic sensation in interactive applications is a challenging task. On the one hand, real-time dynamics simulation and efficient collision detection are essential for modeling the physical world and graphics rendering, while high-performance haptic interfaces, stable force rendering algorithms, and a good understanding of the psychophysics of touch and perceptual factors [4] are required for modeling the forces resulting from the interactions in the simulated world.

Take virtual surgery applications as an example. It involves realistic replication of complex behaviors of the physical world, including interactions of surgical tools with multiple deformable objects, bleeding and smoke generation due to cautery procedures, as well as tissue approximation procedures such as suturing and stapling [5]. Although physics engines have been made available to facilitate virtual surgery by offering multi-physics simulation for rigid and deformable bodies, cloth and fluids, it required intensive computation and the high latency incurred is unfavorable for stable haptic rendering. Stable force feedback requires a data update rate of 1k Hz which is much more demanding than the refresh rate needed for smooth graphics rendering, i.e. 30~60 Hz [6]. The computational speed for realistic dynamics simulation of the physical world could even be much slower. It is therefore non-trivial and a trick task to integrate haptic rendering with physics engines, where a good compromise between the wide disparity in data update rate cannot be established easily. This presents a significant barrier for the incorporation of haptic rendering into real-time interactive virtual environments [5]. From a developer's point of view, the integration of haptics and dynamics simulation is a major consideration that would eventually affect the software quality and the development cycle [7].

At the programming level, one common solution to the issue of update rate disparity is to use multi-threading techniques, which allows haptic and graphic rendering to be executed concurrently in separate threads so that each rendering loop can run at its own refresh rate. While the responsiveness and throughput of the applications are improved, the multi-threading technique increases the complexity of software development where additional synchronization of shared resources is needed and the difficulty in debugging is increased. In particular, the error caused by one thread can kill the entire process because the entire memory space could be shared by all threads. Additional measures are thus needed for data synchronization during concurrent operations such as read-and-write memory access in the multi-threaded program. Otherwise, the data race can lead to deadlocks since all the threads are blocked and waiting for the others to release the required data.

An alternative approach to tackle the issue of update rate disparity is to parallelize the simulation processes so that they can operate as independent execution units that contain their own status information, use their own address spaces, while only interact with each other by means of inter-process communication (IPC) mechanisms. In the paper, this mechanism is adopted to develop a framework that uses the

remote virtual coupling based on IPC in order to guarantee haptic rendering to execute at 1k Hz without being affected by physics simulation engines. This framework decouples haptic rendering from physics simulation, thus providing stable haptic feedback even under complex simulation scenarios running at a low update rate.

The rest of the paper is organized as follows. First, the related work of the study is reviewed in section II. The simulation framework is then presented in section III, including system architecture design, abstraction layers and haptic rendering algorithms. In section IV, the benchmark model is described to evaluate remote virtual coupling and interpolation based haptic rendering algorithms. The detailed implementation of the system and the evaluation are presented in section V. Lastly, discussions and conclusions are given in section VI and VII respectively.

II. Related Work

Haptic rendering refers to the process of computing and generating forces in response to user interactions with virtual objects [8]. The forces can be generated based on the penalty depth and coupling distance [9] which are calculated using the pose (position and orientation) of the haptic interface and that of the haptic pointer (rigid body, god-object [10], proxy [11]). To simulate the interactions with dynamic virtual worlds, it is straightforward to update the physics simulations at the same update rate of the haptic simulations. However, because of restrictions caused by computational resources and simulation complexity, it is difficult to ensure the haptic data to be updated at 1k Hz for stable force displaying [6].

To solve this issue, multi-rate techniques were proposed, which has succeeded in improving the stability and responsiveness of haptic rendering systems [12]. The idea was to perform updates of the virtual environment at a low frequency (limited by computational resources and system complexity) while performing high-frequency updates for force feedback by using simplified or approximate force models. These multi-rate techniques [13, 14, 24] were extended from the method of intermediate representation [15] which was proposed by Adachi et al. to enable interactions with a static virtual world by handling the simulation process with two separate threads, namely the collision detection thread and the haptic rendering thread. Each thread was executed at a different update rate and synchronized at the slowest update rate. In the slow collision detection thread, a plane was computed to serve as a unilateral constraint for the force-feedback thread. This technique was later adapted by Mark et al. to interpolate the intermediate representation between updates [14], which enabled higher stiffness values than the approaches that computed the feedback force values at the rate imposed by collision detection. Besides, Hasegawa et al. [13] made use of an impulse rendered in a haptic thread to update and stabilize movement of rigid bodies managed in a physics thread. The multi-rate approach remains a common approach for simulating haptic interactions with deformable models and has been further developed for many other applications [16~18].

Virtual coupling [9] is another robust stable haptic rendering algorithm, which was first proposed by Colgate et al. The algorithm overcomes the instability caused by the artificial addition of energy into the simulation system [19]. A damped spring was introduced between the pose of the haptic device and the simulated pose, where a trade-off between stability and performance was achieved by tuning the spring stiffness. Later, Akahane et al. [20] implemented a haptic display at 10k Hz update rate by interpolating and up-converting the forces generated by virtual coupling to achieve a stiff, high resolution haptic rendering system. Besides, Otaduy et al. [21] developed a haptics-enabled system with a haptic thread of high update rate and the contact thread of low update rate. The haptic thread then calculated the coupling force and simulated the dynamics of the haptic pointer to realize a stable haptic display with a low-mass-value haptic pointer.

To improve the virtual coupling method, a constraint-based coupling algorithm based on the god-object method [10] was proposed by Ortega et al. [22]. Unconstrained and constrained acceleration of haptic pointer were introduced to calculate the feedback forces. The constrained-based coupling technique can suppress the artificial friction or sticking effect occurred in the conventional virtual coupling. Glondu et al. [23] proposed an algorithm called haptic sub-world using a contact graph, which allowed the simulation of rigid bodies with high update rate and enabled the rendering of feedback forces without artifacts. However, this method cannot handle the situation where all rigid bodies are in contact with each other.

However, most of these multi-rate and virtual coupling haptic rendering techniques were implemented in the multi-threading framework, which has some intrinsic shortcomings as mentioned in the previous section. To solve these issues when integrating haptics with physics engines, a remote virtual coupling based on the IPC framework is proposed in this paper.

III. Simulation Framework

In this section, the framework proposed for stable haptic rendering in the integration of haptics and physics simulation is presented. The system architecture, abstraction layers, haptic rendering algorithms involved in the framework will be discussed.

A. Client-Server Architecture Design

By employing the client-server architecture design, the haptics and physics simulation are clearly specified by properly defining the communication protocol. The block diagram in Fig. 1 illustrates the client-server architecture designed for the framework. The server side hosts the physics simulation application that provides the simulation results by utilizing a physics engine. The client side hosts the haptic rendering application mainly responsible for calculating feedback forces at a 1k Hz update rate based on the simulation results transferred from the server. The communication layer operates on a protocol defined to ensure information such as poses of the haptic device and simulation results is exchanged timely and correctly. Notice that the proposed framework only emulates a network. It runs on a single computer without physical connection to other computers.

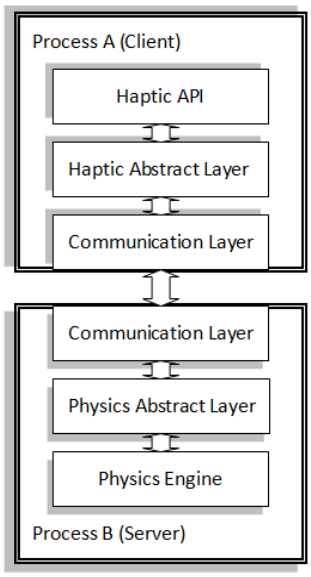

Figure 1. Block diagram of the system architecture design.

In the server side, instead of allowing the physics engine to interface directly with the communication layer, the physics abstraction layer, as an abstraction interface of the physics simulation Application Programming Interface (API), is introduced to improve development flexibility and ease of use.

Here, implementation details of the physics engine are abstracted to provide an organized and clean interface to the developers, where the abstraction layer enables developers to implement their version of physics simulation systems through a unique interface, thus allowing them to switch between different physics engines depending on their needs and performance level required.

In the client side, the Haptics Abstraction Layer (HAL) is an interface defining a common API that provides a robust toolkit for accessing functionalities of haptic devices from different manufacturers. It exposes a set of interfaces that hide the differences among the devices and makes the application device independent. This layer is highly extensible and customizable to cater for diverse types of haptic devices and special needs in individual applications.

The communication layer (CL) is responsible for exchanging information between the server and the client. As shown in Fig. 2, the information include haptic interface pose, velocity, button status, and so on from the client side, and the simulation results such as the virtual tool's proxy pose and velocity from the server side. The client CL runs on a thread at 50 Hz which parcels and sends the spatial information of the haptic device to the server CL.

The server CL will block its own thread until it receives one data packet. The data from the packet will then be extracted by server CL to calculate the force for the proxy in the simulation environment, based on the pose difference between the proxy and the haptic device using a spring-damper model. These events are also described in the sequence diagram in Fig. 3.

It is worth noting that the coordinates of the haptic space might be different from that of the simulation engine and coordinate mapping would then be required.

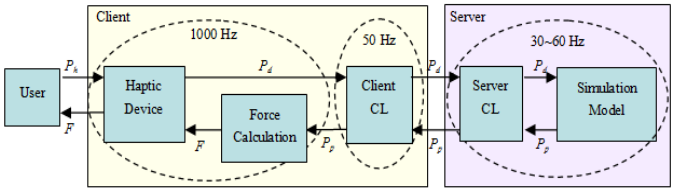

Figure 2.Data flow in the client-server based haptic rendering framework.

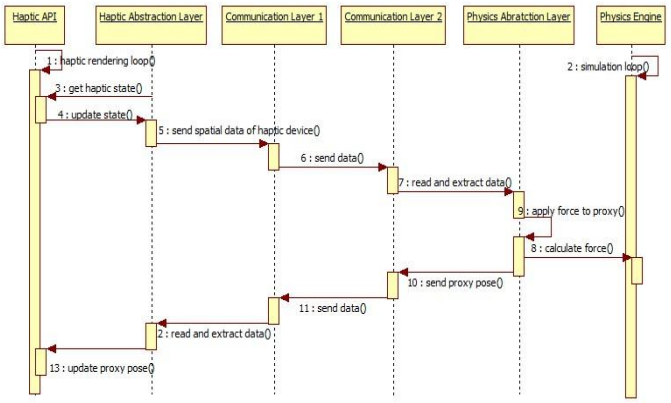

Figure 3.Sequence diagram of the framework.

B. *Haptics In Physics Simulation*

Two issues are concerned for stable haptic rendering in the integration of haptics in physics simulation. First, in most physics engines, dynamic objects are usually directly subject to applied forces instead of poses from external interactions. However, when dealing with a position-controlled impedance-style haptic device, the direct inputs from the user to the haptic device are poses rather than forces. That means the user cannot directly manipulate the objects in the simulation via pose control. It is therefore necessary to converts poses to forces.

The other issue is the disparity of data update rate as discussed in the previous sections. Only simple simulation applications are able to run at the haptic device's servo loop rate (typically about 1k Hz) while physics simulation is usually only optimized to run at the same rate as the graphics loop (30~60 Hz).

Virtual coupling technique [9] is commonly employed to achieve stable haptic rendering. In this technique, the simulation is stepped in a separate thread and a synchronization mechanism is used to update the positional inputs, as shown in Fig. 4. Each thread samples the respective spring-damper positions. The spring-damper used by the haptic device is attached to a proxy to update its position whenever the simulation is stepped. Similarly, the simulation samples the device position before each simulation step in order to calculate the input force that will be applied to the object being manipulated in the simulated environment.

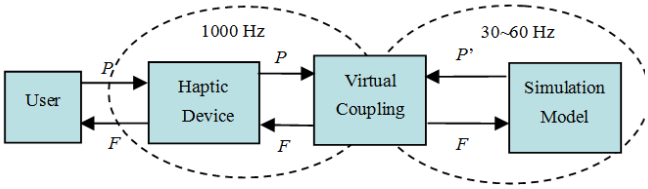

Figure 4.Virtual coupling method for stable haptic rendering.

Note that virtual coupling is usually applied in the same process. However, in our framework, the haptic rendering and physics simulation are executed as two separated processes running at different frequencies in an asynchronous manner. The server takes care of the major processes in the entire simulation, including collision detection and response, soft object deformation, and force calculation, whereas the client one only handles haptic rendering. Therefore, in this paper, the concept of virtual coupling is extended to an inter-process framework, namely remote virtual coupling, which will be explained in the next section.

C. Remote Virtual Coupling

Virtual coupling introduces an intermediate layer between the haptic device and the simulation. It is modeled by connecting a spring-damper system between the simulated body and the device end-effector. The spring-damper system provides a stable mechanism for the haptic device and the simulated body to exchange forces. Fig. 5 depicts the mechanism of the remote virtual coupling. In the client side, the haptic device is linked to a virtual object A with a spring-damper system. The pose of the virtual object is updated by the proxy in the simulation (server side) through the communication channels. The force is calculated at around 1k Hz based on the spring-damper model, whereas the pose of the virtual object is updated at 50 Hz. In the server side, the proxy is linked to a virtual object B based on the spring-damper model. The pose of object B is updated by the haptic device in the client side via the communication channel. The force calculated by the spring-damper model will be applied to the proxy, so that its pose will be updated at 30~60 Hz subject to the interactions between the remote haptic device and the local simulation environment. Different constants are used in the two spring-damper systems to compute the forces to be applied respectively to the device and the manipulated proxy in the simulation environment. The forces can be tuned to achieve a suitable trade-off between stability and smooth haptic feeling without evident sticking artifacts.

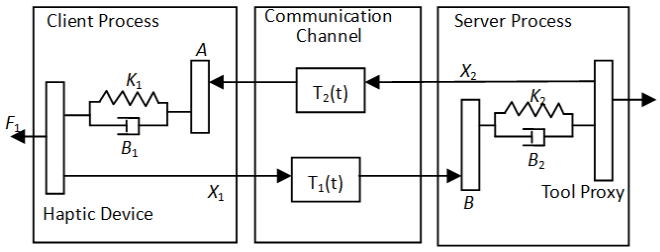

Figure 5.Schematic diagram of remote virtual coupling in IPC.

IV. Benchmark

The proposed remote virtual coupling technique is evaluated by comparing its performance with the interpolation-based haptic rendering method. In the interpolation method, the force is calculated at the server side based on the proxy manipulated by the haptic device and the end-effector pose of the haptic device using a spring-damper model. At the same time, the force is transferred from the server side to the haptic rendering loop in the client side by means of the IPC mechanism as mentioned in the previous sections. Since the force is calculated in server side at a graphic update rate (30~60 Hz) and the force needs to feed haptic rendering at a much higher update rate about 1k Hz, a force interpolation method thus needs to be adopted in order to resolve this disparity issue. Refer to Fig. 6, the haptic force is interpolated as,

$$f_j^h = \alpha[\, f_{i-1}^s + \frac{(f_i^s - f_{i-1}^s)}{N} \times (j\%N)] \qquad (1)$$

$$N = T_h / T_s \qquad (2)$$

$$f_i^s = K \times (P_i^p - P_i^h) - B \times V_i^p \qquad (3)$$

where f_j^h is the interpolated force in the haptic loop at time step j within a time interval T_h, α is the scale factor, f_i^s is the calculated simulation force in the simulation loop at time step i within a time interval T_s as shown in Fig. 6, K is the stiffness constant, B is the damping coefficient, P_i^P is the position of the proxy, P_i^h is the position of the end-effector, and V_i^P is the velocity.

To compensate the delay due to update rate disparity between the physics simulation and haptic rendering process, an improved version of the equation (3) for calculating simulation force is developed by taking the predicted position of the proxy into account, i.e.,

$$f_i^s = K \times (P_{i+1}^p - P_i^h) - B \times V_i^p \qquad (4)$$

where P_{i+1}^P is the predicted position of the proxy at time step, i+1

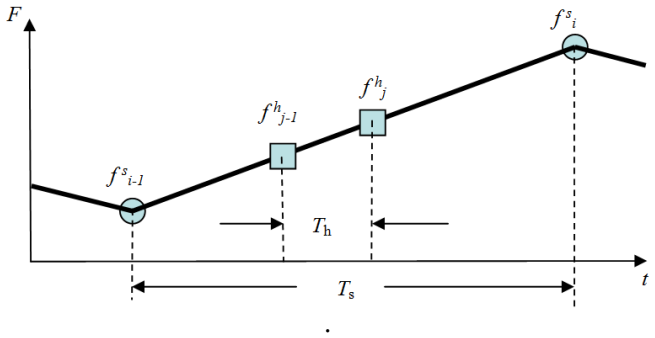

Figure 6.Force interpolation-based haptic rendering.

Several experiments are carried out to compare the performance of the remote virtual coupling technique with the interpolation-based haptic rendering algorithms. Here, the passivity and stability of two methods are compared. In the benchmarking experiments, the haptic device PHANTOM® DesktopTM from the Sensable Technologies is used. It has 6 degree-of-freedom positional input and 3 degree-of-freedom force output. The proximal end of the device is fixed with the secondary arm, as shown in Fig. 7.

The end-effector of the haptic interface is initially lift vertically upwards in the air at the position y=20 mm with zero initial velocity.

When the stylus is released, it drops in free fall due to gravity until it hits the virtual horizontal wall at the position, y= 0 within the horizontal X-Z plane. When it hits the virtual wall, the force calculated by a spring-damper model pushes the stylus vertically upwards. First, the proxy-based method is used, i.e. no special technique is used to streamline haptic rendering. Therefore, the trace of the end-effector in Y direction over time is shown in Fig. 8, which is in the form of a damping curve. It can be seen from the figure that the end-effector of the stylus bounces for about 0.7 second, and finally stays at the position, y=-2 mm.

The experiment is repeated using the two haptic rendering methods and the force-time curves are measured. With K=100 N/s, B=0.01 Ns/m and using equation (3) for force interpolation, it can be seen from Fig. 9 that the force oscillates for more than 4 seconds, which is too long and not favorable for rendering smooth haptic sensation. When the prediction of proxy position is considered, with equation (4), it is found that the force can quickly converge within 0.4 second, as shown in Fig. 10. When a higher stiffness constant is applied, say K=500 N/m, the force generated by the interpolation-based method diverges very quickly even though the prediction technique is used, while the force generated by remote virtual coupling can still converge as shown in Fig. 11. These experiments demonstrate that the remote virtual coupling technique outperforms the interpolation methods in terms of stability and robustness. The improved version of force interpolation method with proxy position prediction is a better approach than the conventional force interpolation method for stable haptic rendering.

Figure 7.The haptic device, PHANTOM® DesktopTM used in the simulation system. The proximal end of the stylus is fixed with the secondary arm using a rubber band

V. Performance Evaluation

Two simulation scenarios, involving interactions of rigid and soft objects respectively, are implemented using the proposed framework to investigate the performance of the remote virtual coupling on the IPC platform and the interpolation-based haptic rendering algorithms. The experiments are performed on an Intel i7-2600 (3.4G Hz) personal computer running Windows 7, with 4 GB RAM, an NVIDIA Quadro 4000 graphics card, and a PHANTOM® Desktop as the haptic device.

A. Software Tools

Several software packages are used to implement the proposed haptic rendering framework, including Bullet (http://bulletphysics.org), Physics Abstraction Layer (PAL) (http://www.adrianboeing.com/pal), and Haptik (http://sirslab.dii.unisi.it/haptiklibrary/). Bullet is an open source physics engine which features 3D collision detection, soft body dynamics, fluid simulation, and rigid body dynamics. PAL is a high-level interface for low-level physics engines used in games, simulation systems, and other 3D applications. It also supports a number of dynamic simulation methodologies. Haptik is a component-based lightweight library that provides a hardware abstraction layer for access to haptic devices. Hardware from different manufactures can be easily accessed in a uniform way, enabling applications to be conveniently developed regardless of dependencies on specific APIs, hardware and drivers. The IPC is implemented with named pipes. A duplex pipe is employed for communication between the pipe server and one or more pipe clients. In Windows, the design of named pipes is based on client-server communication, working in a way like sockets.

B. Rigid Bodies

The dynamics of rigid objects is simulated by using the rigid body simulation package provided by the physics engine Bullet. The objects are manipulated with the haptic interface as shown in Fig. 12. In this experiment, the performance of the framework is evaluated by simulating 125, 250 and 500 rigid cubes in the virtual environment. The results in Fig. 13 show that the average update time per step is about 10 ms (100 Hz) for 125 cubes, 23 ms (about 43 Hz) for 250 cubes, and 57 ms (about 18 Hz) for 500 cubes. In the last case, the simulation update rate is less than 20 Hz, indicating that visual discontinuity can be perceived. As the simulation engine occupies a lot of computation, high update rate required for smooth haptic rendering cannot be guaranteed if haptics and physics simulation run on the same process.

When the multi-processing framework proposed in this paper is applied, haptic rendering and physics simulation are executed on separate processes. The computation resources of the former will not be deprived by that of the latter; even the physics engine may occupy a large amount of CPU time and memory resources. This is demonstrated with the experiments where the proxy is controlled by the haptic device to push through (along axis Z) a stack of 5×5×5, 10×5×5 and 10×10×5 cubes respectively, as illustrated in Fig. 14. The haptic and graphic update rates with regards to different simulation complexity are listed in Table 1. The haptic update rates are calculated based on the average update time in the haptic

rendering loop. The sampling time lasts for about 20 seconds. It can be seen from the table that while the update rate of the simulation engine drops down from 100 Hz to 18 Hz as the number of rigid cubes increases from 125 to 500, the haptic update rate remains relatively stable around 1k Hz.

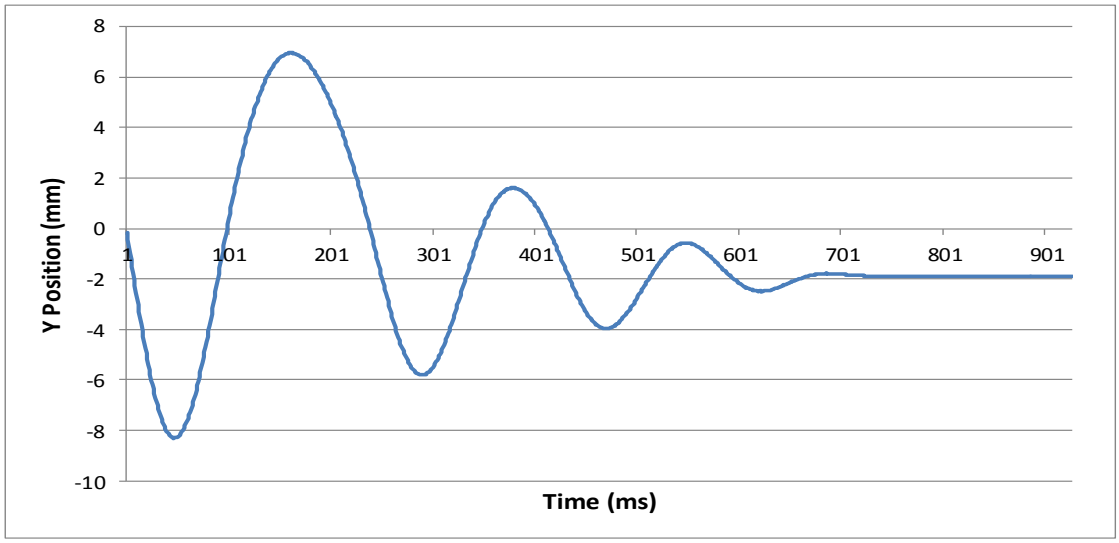

Figure 8.Position curves obtained with simple proxy-based method, which is executed in the haptic loop at 1k Hz) (K=200 N/m, B=0.02 Ns/m).

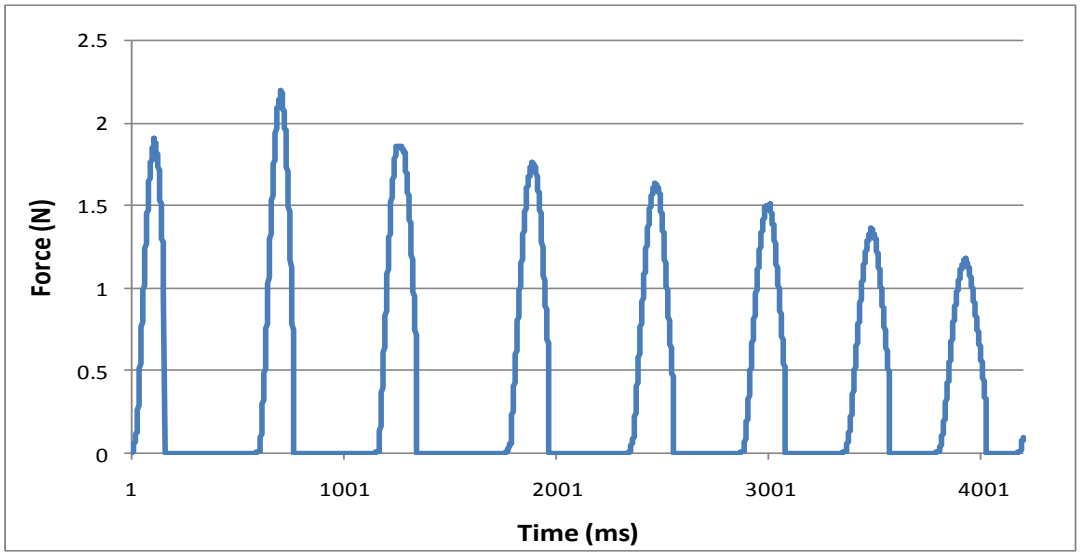

Figure 9.Force profile obtained using the interpolation-based method without prediction, with K=100 N/m, B=0.01 Ns/m

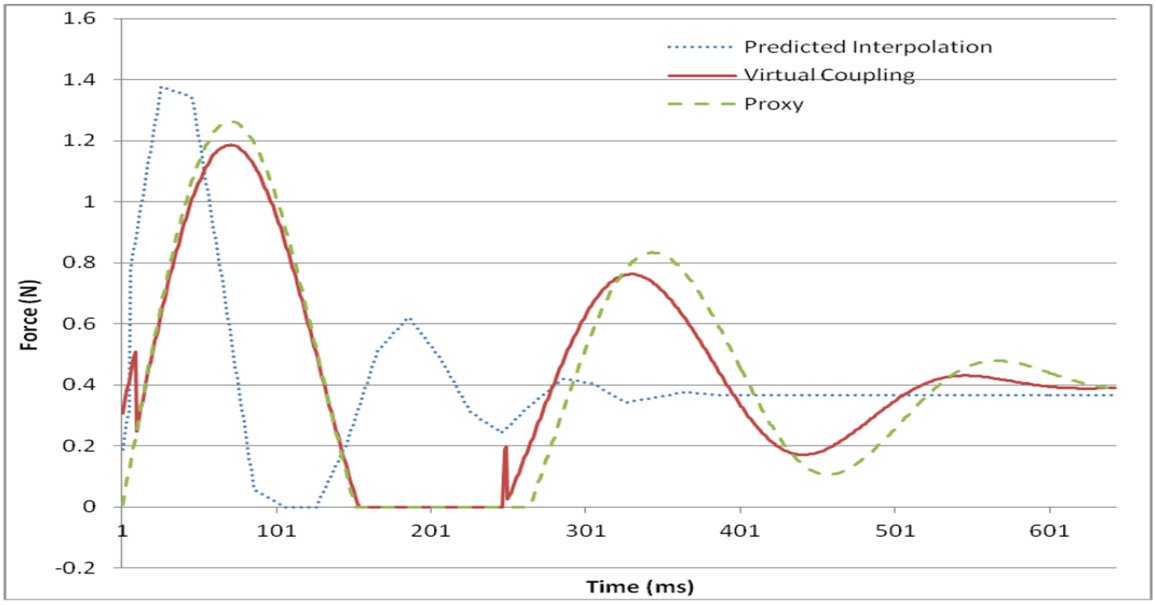

Figure 10.Force curves obtained with virtual coupling and the interpolation-based method with prediction (K=100 N/m, B=0.01 Ns/m). The green dotted line, referring to the force curve obtained by proxy-based haptic rendering at 1k Hz update rate, is provided as a reference

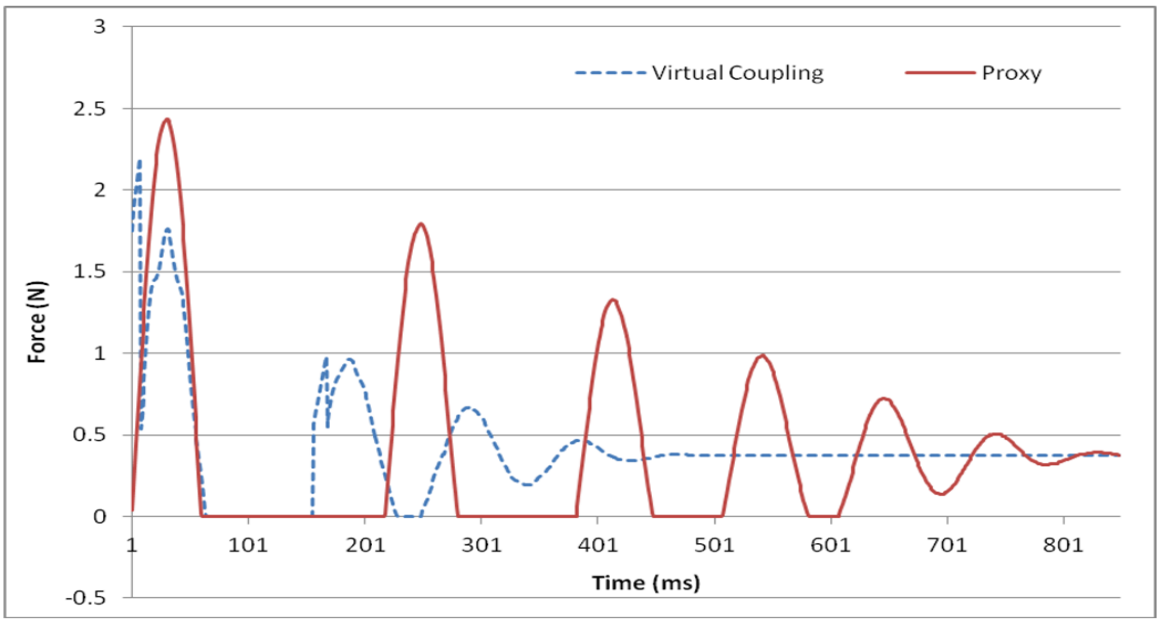

Figure 11.Force curves obtained using virtual coupling (K=500 N/m, B=0.05 Ns/m). The red solid line, referring to the force curve obtained by proxy-based haptic rendering at 1k Hz update rate, is provided as a reference.

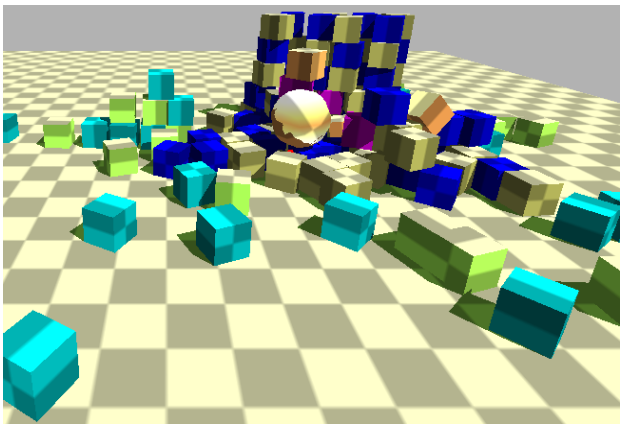

Figure 12.Screenshot of the simulated interactions of dynamic rigid cubes. The proxy (sphere) is manipulated by the haptic device to interact with the cubes.

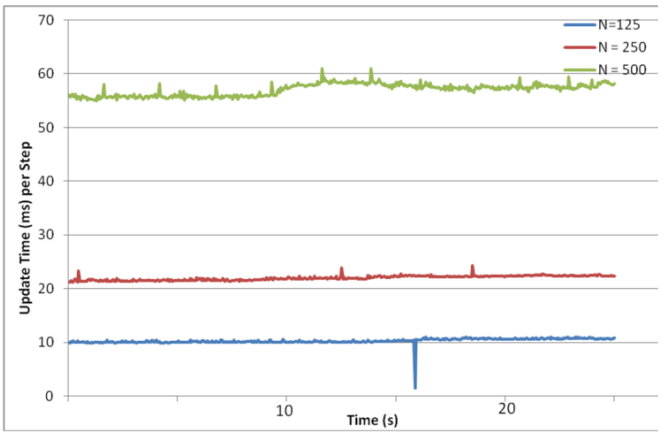

Figure 13.Time taken for one simulate step when the number of cubes N in the scene is 125, 250 and 500 respectively.

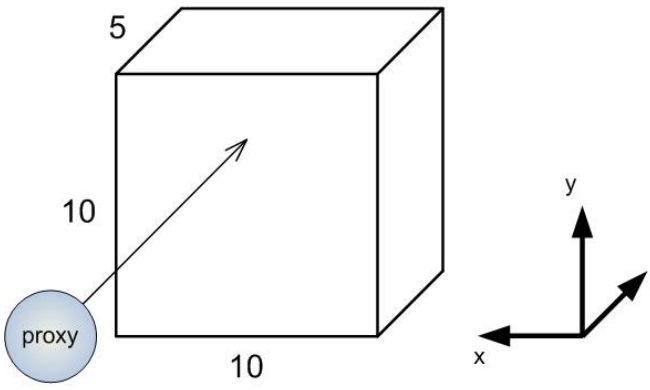

Figure 14.Pushing the proxy toward a stack of 10□10□5 cubes.

TABLE I. UPDATE RATES MEASURED FROM THREE SIMULATION SCENARIOS.

Number of Rigid Cubes	125	250	500
Simulation Engine Update Rate (Hz)	100	43	18
Haptic Rendering Update Rate (Hz)	1008	1003	1003

In the experiments, the corresponding pushing forces are recorded and plotted as shown in Fig. 15 and 16. Two different haptic rendering algorithms, force interpolation and remote virtual coupling are applied in the experiments. It can be seen from the figures that the forces calculated by the both algorithms are stable. While the force vibration can be perceived visually and kinesthetically in the interpolation-based haptic rendering algorithm, the forces generated by remote virtual coupling have relatively smoother profiles.

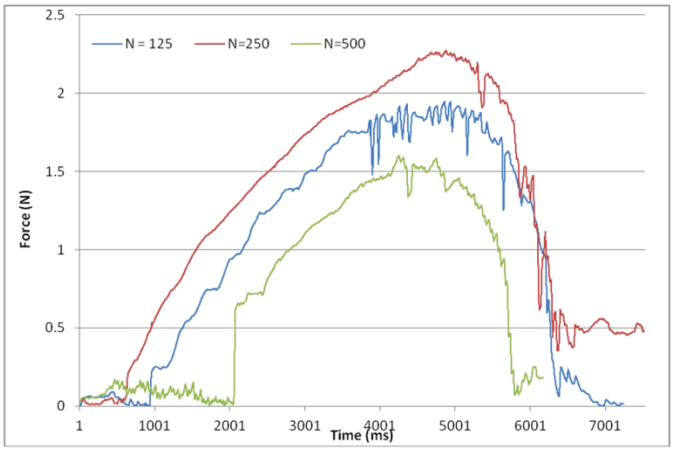

Figure 15.Force profiles measured when the proxy is moved toward a stack of 125, 250 and 500 cubes. The interpolation-based haptic rendering method is used in the experiment.

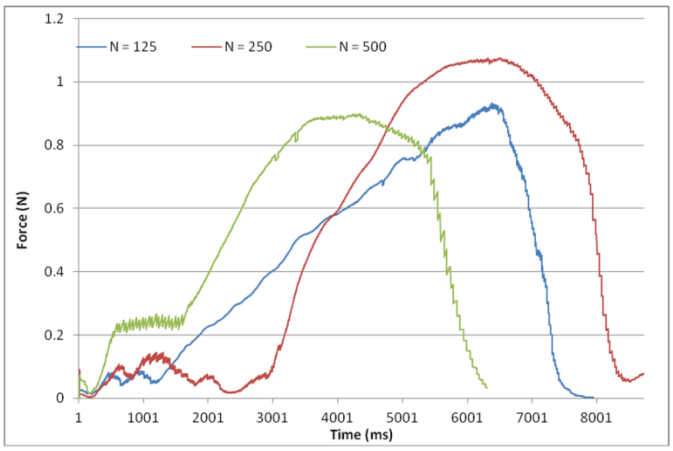

Figure 16.Force profiles measured when the proxy is moved toward a stack of 125, 250 and 500 cubes. The remote virtual coupling technique is used in the experiment.

C. Soft Objects

In this experiment, five soft bunny models are simulated in the virtual environment. As shown in Fig. 17. The bunnies can be touched by the proxy manipulated by the haptic device. Each bunny model has 450 nodes, 1353 edges and 902 faces. The contact forces generated by the interpolation-based method and the remote virtual coupling technique are plotted respectively in Fig. 18 and 19. It can be seen that the force profile of the remote virtual coupling is smoother than that of the interpolation-based method.

VI. DISCUSSIONS

While physics engines have been developed to reduce the effort required to simulate the physical world, they do not have much support for haptic rendering. In this study, research is conducted to facilitate the integration of haptic feedback into interactive applications developed with physics engines. To deal with the problem of update-rate disparity among the visualization, physics and haptics simulation processes, the proposed framework attempts to bridge the gap by making use of the techniques of remote virtual coupling technique and inter-process communication.

Notice that the purpose of the study is to propose a stable haptic rendering algorithm based on the framework rather than improving the computational performance or visual appearance of existing systems. Experiments have been carried out to evaluate the feasibility of the proposed method with the interpolation-based method. The results demonstrate that remote virtual coupling has better stability and robustness and that physics engines can also be used to support stable haptic rendering although this is not primarily designed for that purpose.

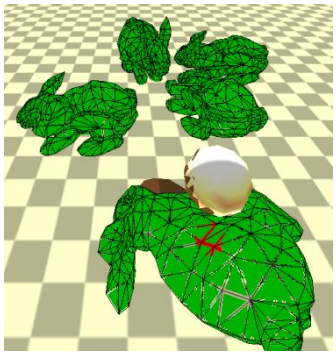

Figure 17.Screenshot of the interactions between the proxy and the soft bunnies.

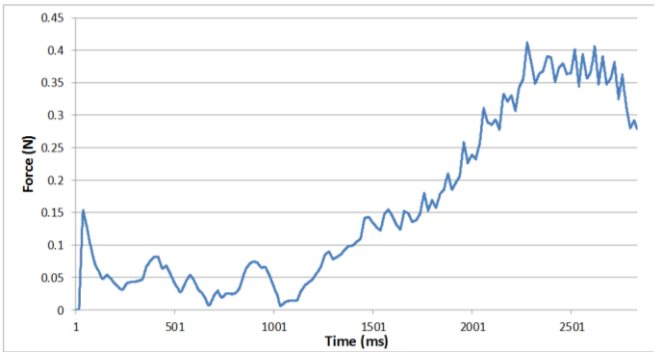

Figure 18Force profile recorded with interpolation-based haptic rendering.

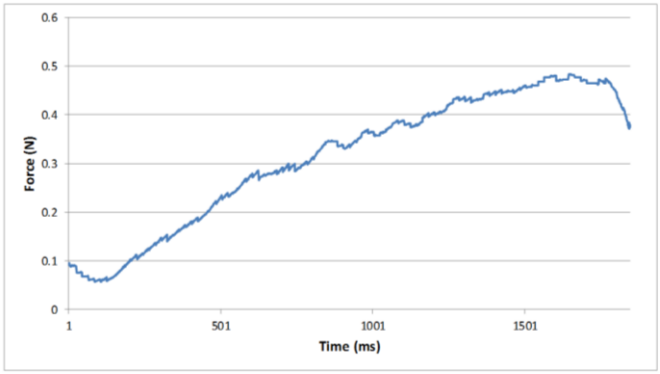

Figure 19.Force profile recorded with remote virtual coupling.

However, some latency and sticking force effects are observed even the parameters of the two spring-damper models, in the server and client side respectively, have been tuned carefully for remote virtual coupling in IPC. One possible solution is to use a massless proxy instead of the spring-damper method in server side. The poses of the proxy can be manipulated by the remote haptic device directly. However, objects in most physics engines are usually controlled by forces and velocities rather than poses.

As illustrated in section V.B, the remote virtual coupling has relatively smoother force profiles in the cube-pushing experiment. It is worth noting that virtual coupling possibly filters out the useful vibration forces caused by the interactions between the proxy and the dynamic objects. This filtering effect is related to the setting of coupling parameters, i.e. stiffness and damping coefficients. Therefore, these parameters need to be tuned carefully based on the simulated body density, stiffness, size, friction and other physical parameters.

Although the framework based on the client-server architecture design can be possibly extended to networked haptics-enabled systems, it essentially runs on a single computer. To extend it for networked applications, it is necessary to deal with instability due to the stochastic nature of computer networks, e.g. time delay, jitter and packet loss, which are ignored in the communication protocol of the proposed framework.

VII. CONCLUSIONS

In advanced interactive applications, e.g. virtual surgery, haptic rendering plays an important role alongside graphics rendering and physics simulation. The use of physics engines is able to facilitate physically realistic simulation and improve the visualization of the simulation results, but it is yet to be fully compatible with haptic rendering.

The proposed framework fully decouples the haptic rendering from physics simulation by means of IPC and the remote virtual coupling techniques. It is demonstrated experimentally that this framework can guarantee stable haptic rendering even when the physics simulation runs at a low update rate. It also has the potential to reduce development time and effort by exploiting the benefits of physics engine.

Future work will be conducted to further demonstrate the flexibility of the proposed framework by employing other physics engines, e.g. PhysX (http://www.geforce.com), and hapitc devices, e.g. Novint Falcon (http://www.novint.com). Convenient integration of haptic rendering and physics simulation is anticipated since the abstraction layers for the respective processes shall provide convenient interfaces for switching the underlying physics engines and haptic interfaces.

ACKNOWLEDGMENT

This work was supported in part by the Research Grants Council of Hong Kong SAR (Project No. PolyU 5134/12E, 5152/09E) and the Hong Kong Polytechnic University (Project Account Code 87RF).

REFERENCES

[1] Basdogan, C., et al., Haptics in minimally invasive surgical simulation and training. Computer Graphics and Applications, IEEE, 2004. 24(2): p. 56-64.
[2] Lah, T., THUMP: an immersive haptic console for surgical simulation and training. Medicine Meets Virtual Reality 12: Building a Better You: The Next Tools for Medical Education, Diagnosis, and Care, 2004. 98: p. 272.
[3] Salisbury, K., F. Conti, and F. Barbagli, Haptic rendering: introductory concepts. Computer Graphics and Applications, IEEE, 2004. 24(2): p. 24-32.
[4] Otaduy, M.A. and M.C. Lin. Introduction to haptic rendering. in International Conference on Computer Graphics and Interactive Techniques: ACM SIGGRAPH 2005 Courses: Los Angeles, California. 2005.
[5] Maciel, A., et al., Using the PhysX engine for physics‐based virtual surgery with force feedback. The International Journal of Medical Robotics and Computer Assisted Surgery, 2009. 5(3): p. 341-353.
[6] Love, L.J. and W.J. Book, Contact stability analysis of virtual walls. 1995.
[7] Choi, K.S., et al., Haptic Rendering in Interactive Applications Developed with Commodity Physics Engine. Journal of Multimedia, 2011. 6(2): p. 147-155.
[8] Salisbury, K., et al. Haptic rendering: Programming touch interaction with virtual objects. in Proceedings of the 1995 symposium on Interactive 3D graphics. 1995: ACM.
[9] Colgate, J.E., M.C. Stanley, and J.M. Brown. Issues in the haptic display of tool use. in Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on. 1995: IEEE.
[10] Zilles, C.B. and J.K. Salisbury. A constraint-based god-object method for haptic display. in Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on. 1995: IEEE.
[11] Ruspini, D.C., K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. in Proceedings of the 24th annual conference on Computer graphics and interactive techniques. 1997: ACM Press/Addison-Wesley Publishing Co.
[12] Susa, I., M. Sato, and S. Hasegawa. Multi-rate multi-range dynamic simulation for haptic interaction. in World Haptics Conference (WHC), 2011 IEEE. 2011: IEEE.
[13] Hasegawa, S., et al., Inter-process communication for force display of dynamic virtual world. ASME DYN SYST CONTROL DIV PUBL DSC., 1999. 67: p. 211-218.
[14] Mark, W.R., et al. Adding force feedback to graphics systems: Issues and solutions. in Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. 1996: ACM.
[15] Adachi, Y., T. Kumano, and K. Ogino. Intermediate representation for stiff virtual objects. in Virtual Reality Annual International Symposium, 1995. Proceedings. 1995: IEEE.
[16] Astley, O.R. and V. Hayward. Multirate haptic simulation achieved by coupling finite element meshes through norton equivalents. in Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on. 1998: IEEE.
[17] Cavusoglu, M.C. and F. Tendick. Multirate simulation for high fidelity haptic interaction with deformable objects in virtual environments. in Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on. 2000: IEEE.
[18] Duriez, C., C. Andriot, and A. Kheddar. A multi-threaded approach for deformable/rigid contacts with haptic feedback. in Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS'04. Proceedings. 12th International Symposium on. 2004: IEEE.
[19] Adams, R.J. and B. Hannaford, Stable haptic interaction with virtual environments. Robotics and Automation, IEEE Transactions on, 1999. 15(3): p. 465-474.
[20] Akahane, K., et al. A proposal of a high definition haptic rendering for stability and fidelity. in Artificial Reality and Telexistence--Workshops, 2006. ICAT'06. 16th International Conference on. 2006: IEEE.
[21] Otaduy, M.A. and M.C. Lin, A modular haptic rendering algorithm for stable and transparent 6-DOF manipulation. Robotics, IEEE Transactions on, 2006. 22(4): p. 751-762.
[22] Ortega, M., S. Redon, and S. Coquillart. A six degree-of-freedom god-object method for haptic display of rigid bodies. in Virtual Reality Conference, 2006. 2006: IEEE.
[23] Glondu, L., M. Marchal, and G. Dumont, A new coupling scheme for haptic rendering of rigid bodies interactions based on a haptic sub-world using a contact graph. Haptics: Generating and Perceiving Tangible Sensations, 2010: p. 51-56.
[24] Yasrebi, N. and D. Constantinescu. Wave filter bank for high fidelity passive multirate haptic interaction with slowly updated virtual environments. in Haptics Symposium (HAPTICS), 2012 IEEE.

AUTHORS PROFILE

Xue-Jian He received his Ph.D. degree in mechanical engineering from the University of Hong Kong. He is currently a research associate at the School of Nursing, the Hong Kong Polytechnic University. His research interests include haptic modeling, human-computer interaction, medical simulation, virtual reality in medicine and health care.

Kup-Sze Choi received his Ph.D. degree in computer science and engineering from the Chinese University of Hong Kong. He is currently an assistant professor at the School of Nursing, the Hong Kong Polytechnic University. His research interests include computer graphics, virtual reality, physically based simulation, computational intelligence, and their applications in medicine and health care.