Integrating PhysX and OpenHaptics: Efficient Force Feedback Generation Using Physics Engine and Haptic Devices

[1]Leon Sze-Ho Chan [1,2]Kup-Sze Choi

[1]*School of Nursing, Hong Kong Polytechnic University, Hong Kong*
[2]*Ctr. for Integrative Digital Health, School of Nursing, Hong Kong Polytechnic University, Hong Kong*

Abstract

Haptic feedback plays an important role to further enhance the level of realism of virtual environments where visual and audio feedbacks are only provided. However, realistic haptic rendering depends on its coupling to the underlying physics engine that governs the behavior of virtual objects. This paper presents methods to streamline the generation of haptic feedback with physics engine based on Sensable's OpenHaptics and nVidia's PhysX. Minimal development effort is required to couple these two components . To render the forces due the interactions between virtual objects, the Error-based method and the Contact Plane Collision Response method are proposed to utilize virtual material stiffness and object collision geometry provided by PhysX. The latter method yields more jitter-free output by restricting the haptic interface on one side of the contact plane. While PhysX does not release force information, an Indirect Force Estimation technique is proposed to simulate static or dynamic pulling force by introducing a spring between the haptic interface and the object being pulled. By using Hooke's Law, the pulling force can be estimated indirectly from the elongation of the spring. The use of these methods provides the desired force feedback without significant changes to the developer's codebase.

1. Introduction

An increasing number of interactive computer applications rely on force feedback to enhance user experience. Force feedback provides additional information where visual and audio effects are incapable to offer [1]. One type of such applications is virtual surgery [2, 3]. Virtual surgical simulator emulating actual procedures requires a real-time physics engine to compute the response of virtual objects timely and accurately, and to render the visual and force feedback realistically. From a developer's standpoint, the ability to easily couple the physics and the feedback components is a major consideration that affects the quality of the end result as well as development cycle. However, coupling these two components is non-trivial. For example, computational speed of the physics engine may not cope with the high refresh rate required for realistic haptic rendering. Accessibility to key parameters in the source level, e.g. critical force data, of physics engine is also limited. Haptic rendering with physics engine is thus a tricky task, hindering the software development process. To address these issues, a simulation platform, to be used for virtual surgery, is developed by using nVidia's *PhysX* as the primary physics engine and Sensable's *Phantom Omni* haptic devices as the 3D user interface. Methods to integrate the haptic devices with PhysX are discussed in this paper, with emphasis on minimal development work.

2. PhysX

PhysX is a middleware by nVidia that provides real-time physics simulation. It is available on multiple target platforms and can be hardware-accelerated when appropriate hardware is installed on the target machine. Most notable features of PhysX include collision detection and physics-based simulation of rigid bodies, cloth, soft bodies, and fluid [4]. The increasing adoption of PhysX by the industry and the associated reduction in development time make PhysX a popular choice for physics simulations.

3. Haptic Devices and OpenHaptics

To render immersive experience in virtual environments, intuitive 3D user interface is required to enable realistic manipulation of virtual objects. In the developed simulation platform, a pair of Phantom Omni haptic devices by Sensable was utilized to enable two-handed operations. The stylus of the device was to mimic the handle of surgical tool. Virtual objects were manipulated interactively by maneuvering the stylus. Each of these devices has 6 degrees of freedom in position/orientation input, and 3 degrees of freedom in force feedback output.

The *OpenHaptics* API was used to interface with the haptic hardware. Within the API, there are two implementations for reading the current haptic position and rendering forces. The HDAPI permits direct communication with the hardware for position readings and force renderings [5]. The HLAPI provides a higher level of interaction with a haptic device. The HLAPI reuses OpenGL graphics rendering code to construct a scene graph of virtual objects. The haptic rendering engine then uses these virtual objects to automatically update the position and generate the feedback force within the scene. Although the scene graph rendering technique is widely used in simulations [6], HDAPI was used in the developed simulation platform due to its flexibility in producing various force effects, and its independence with graphics APIs (such as DirectX) employed for application design.

4. Force Simulation Layer

A problem of using PhysX with HDAPI is that the forces acting on an object, which are needed to compute the feedback force on the haptic devices, are not accessible. Even when forces are accessible under some conditions, inaccuracy in the solver produces noticeable jitter in the feedback force. The Force Simulation Layer (FSL) is proposed to avoid the above problems by calculating the feedback forces with the geometric data provided by PhysX and rendering the forces on the haptic devices via HDAPI.

The architecture of the simulation software platform as shown in Figure 1 does not significantly differ from a typical PhysX application except with the addition of haptic devices and the implementation of the Force Simulation Layer between PhysX and HDAPI. The FSL acquires the interface position of the currently selected haptic device and transforms this position into the rendering scene coordinate. The object in the scene attached to a haptic interface gets updated with its new position. PhysX uses the new position of the object and advances the physics simulation of the entire scene. Collisions detected by PhysX are reported back to the FSL to compute appropriate resultant force, and to drive the haptic device via the HDAPI.

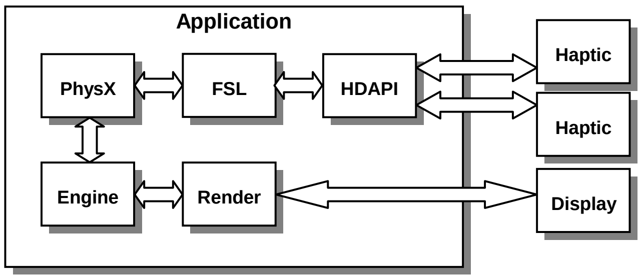


Figure 1. System Architecture

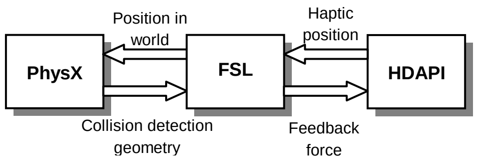


Figure 2. Data flow between PhysX and HDAPI.

Figure 2 is a schematic representation of the interactions between various components. Figure 3 illustrates the sequence of data flow between each component. Since this particular PhysX simulation runs at 60Hz while the haptic interface is required to run at 1000Hz asynchronously, the FSL provides two memory buffers for each haptic device, one for the haptic interface position and the other for storing collision and dynamics data as depicted in Figure 4. These buffers can be safely accessed and modified asynchronously. The processing routine that computes and updates the feedback force runs synchronously with the haptic interface at 1000Hz. All haptic devices are synchronously updated in the same processing routine.

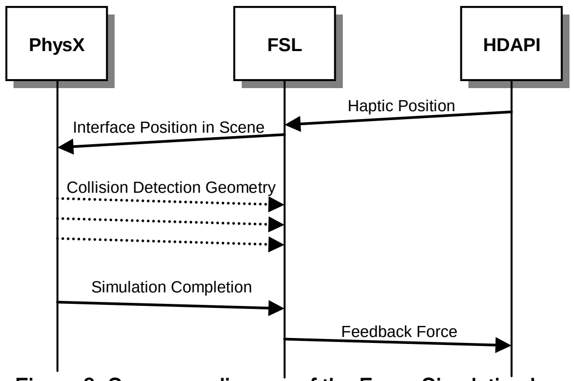


Figure 3. Sequence diagram of the Force Simulation Layer.

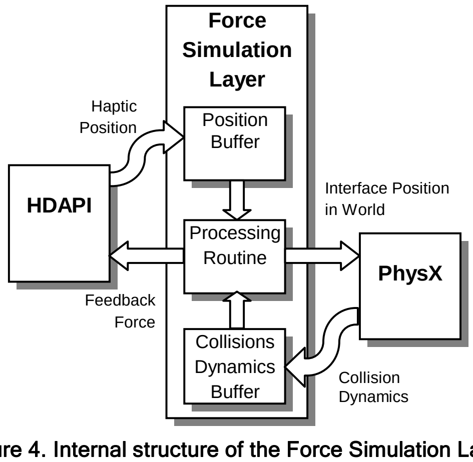


Figure 4. Internal structure of the Force Simulation Layer.

On the haptic side, the position buffer is refreshed during each HDAPI update loop. The position can be read by calling the hdGetDoublev function with one of the following parameters: HD_CURRENT_POSITION or HD_CURRENT_TRANSFORM [7]. Similarly, a force can be sent to the current haptic device by calling the hdSetDoublev specifying HD_CURRENT_FORCE.

On the PhysX side, collisions are intercepted by using the NxUserContactModify::onContactConstraint function. Whenever a collision is detected on an object that has contact modification enabled, this function will be called where useful geometric information about the collision, such as the contact point, contact normal, and penetration, can be extracted for force feedback computation. Depending on the chosen method of force estimation, the structure of the processing routine and the data received from PhysX would vary.

5. Collision Response

Two methods are proposed to exploit PhysX for handling collision response and generating feedback forces.

5.1 Error-Based Collision Response

Error-based collision response takes advantage of the fact that when a collision between a pair of objects occurred, collision geometry data become available to the application. Specifically, PhysX automatically computes the penetration depth e (or termed Error in PhysX), contact point $\mathbf{p}_C$, and unit contact normal $\mathbf{n}$ of the two colliding object as shown in Figure 5. If the objects are considered to have a combined material stiffness k, then the reaction force can be found according to Hooke's Law:

$$F = ke \qquad (1)$$

Since the penetration depth is along the unit contact normal:

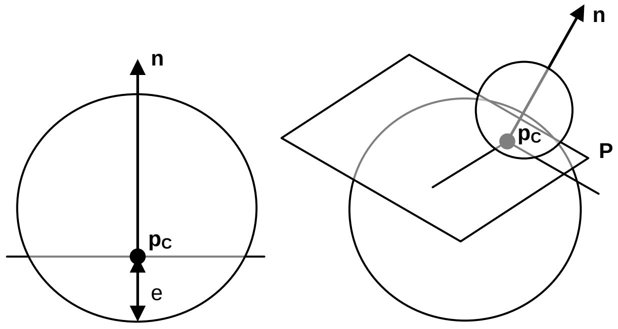


Figure 5. Collision geometries for the error-based (left) and the contact plane (right) collision response.

$$\mathbf{F} = \mathbf{n}ke \qquad (2)$$

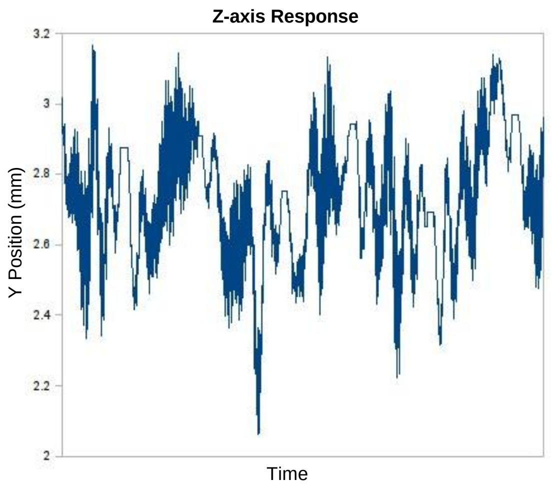
The penetration depth is updated in every PhysX simulation step at 60Hz; however, the haptic device requires a refresh rate of 1000Hz. Therefore, between PhysX simulation steps, a new feedback force $\mathbf{F}$ can be extrapolated by taking the current haptic position $\mathbf{p}_H$ and comparing it to the haptic position in the most recent simulation step $\mathbf{p}_S$:

$$P_n = (\mathbf{p}_H - \mathbf{p}_S) \bullet \mathbf{n} \qquad (3)$$

$$\mathbf{F} = k[P_n + e]\mathbf{n} \qquad (4)$$

The above calculation approximates the feedback force during each haptic update. The function NxUserContactModify::onContactConstraint executes in each PhysX simulation step as long as two objects are in contact with each other. Penetration depth e remains in the equation because $\mathbf{p}_S$ is constant between two PhysX simulation updates.

Tests were conducted by implementing the error-based method and testing it on a virtual flat plane y = 3 with stiffness k = 1.0 and motions along the x- and z-axis



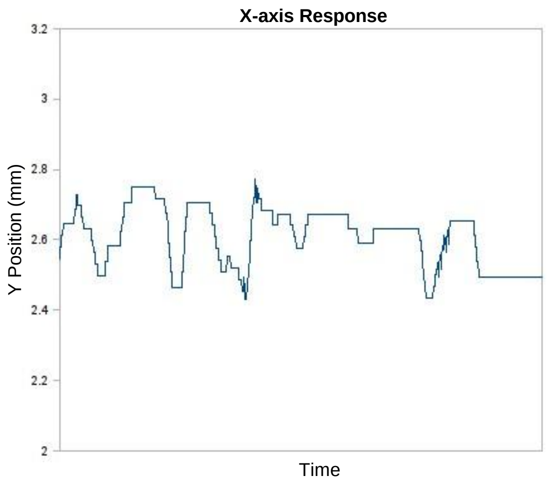


Figure 6. Motion jittering along the z and x axes when the error-based method is used.

of the scene. The position variations of the haptic interface for motions along the x- and z-axis are shown in Figure 6.

An unexpected observation is that motion along the z-axis has a less stable output than along the x-axis. Comparing the above figures, the z-axis motion produces plenty of jitters on the haptic device. Attempts have been made by using other sets of haptic devices to identify the problem but it was suspected that PhysX has different solver accuracy along different axes when performing simulations.

5.2 Contact Plane Collision Response

An alternative method to compute the feedback force is thus constructed by using PhysX's collision data. Instead of using the penetration depth, a contact plane is constructed from the contact point and contact normal. The haptic interface is then restricted to motion on one side of the contact plane. Any penetration through the plane will be resisted by a force according to the depth of penetration and the material stiffness.

Refer to Figure 5, the equation of the contact plane **P** can be obtained from the contact normal **n** and contact point $\mathbf{p}_C$:

$$Ax + By + Cz = D, \qquad (5)$$

where A, B, and C correspond to the x-, y-, and z-component of **n** as given by PhysX. D can be calculated by substituting the contact point into the plane equation and solving for D. It can be created readily with PhysX's plane creation routine NxPlane.

Once the contact plane is defined, the haptic interface position can be tested against this plane for penetration during each haptic update. This is done by finding the distance e from the haptic position $\mathbf{p}_H$ to the contact plane:

$$e = \frac{Ap_{H_x} + Bp_{H_y} + Cp_{H_z} - D}{\sqrt{A^2 + B^2 + C^2}}, \qquad (6)$$

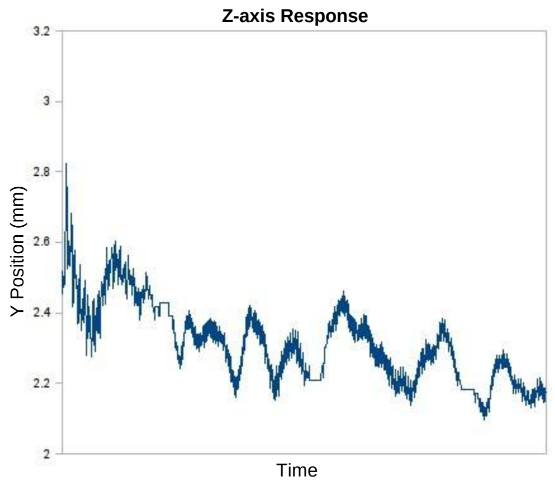
which can be obtained by calling the NxPlane::distance function in PhysX. Whenever the penetration depth becomes negative in the direction of the contact normal, the feedback force is updated using Hooke's Law. Using the same set of test conditions as the error-based method, the obtained position variations are shown in Figure 7.

The axis-dependent vibration issue discussed in the error-based method also exists in this case. However, with the avoidance of using the penetration depth supplied by PhysX, the overall motion achieved with the contact plane method becomes more stable than error-based method.

6. Indirect Force Estimation

The above collision response methods are suitable for simulating *pushing* forces due to collision contacts. However, generation of force feedback when pulling an object with a haptic device requires a different approach. From classical mechanics, a system can be isolated with equal but opposite forces. By isolating the haptic interface from the rest of the system, the internal force becomes exposed as illustrated in Figure 8.

Assuming that the haptic interface is massless, the reaction force would equal the pulling force in magnitude with an opposite direction, which can be used to provide force feedback to the haptic device. Unfortunately, PhysX does not make the reaction force available to the application. A simple solution to this problem is to estimate the force indirectly with a spring introduced



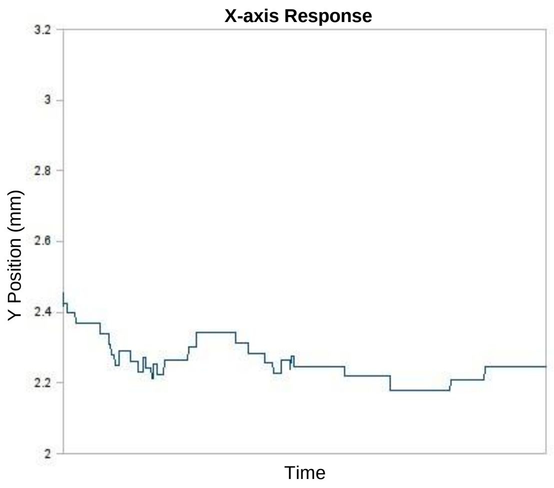


Figure 7. Motion jittering along the z and x axes when the contact plane method is used.

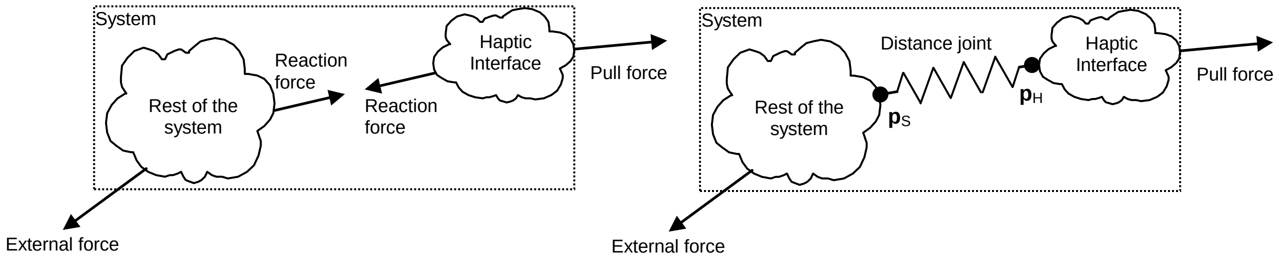


Figure 8. Indirect force estimation with a spring joining the system and the haptic interface.

between the haptic interface and the rest of the system. By measuring the elongation of the spring, the reaction force can be deduced by using Hooke's Law. PhysX provides *distance joint* which can be used to link the haptic interface and the rest of the system while acting as a spring with constant stiffness k.

During simulation, the length of the joint and the attached location of the joint on each object $\mathbf{p}_S$ and $\mathbf{p}_H$ are updated and available for query. The reaction force (and hence feedback force) **F** can be obtained using the equation below:

$$\mathbf{F} = k(\mathbf{p}_S - \mathbf{p}_H). \qquad (7)$$

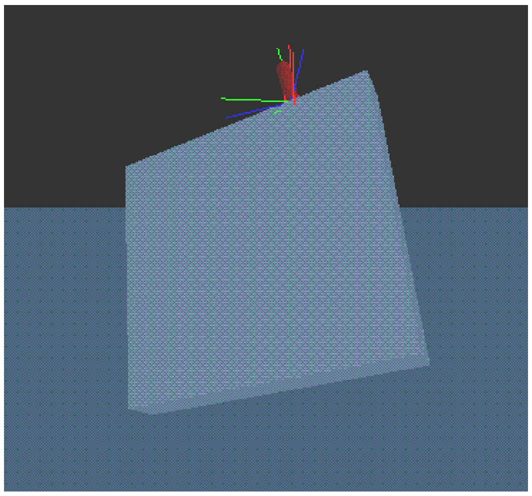
Figure 9 shows the virtual setup used for testing an implementation of the indirect force estimation method. A massive virtual plate was lifted up using the haptic device. The captured feedback force rendered to the haptic device shows that the plate oscillated while approaching a steady state.

7. Conclusion

Two approaches to approximating the feedback force from collision data supplied by PhysX were illustrated. For both cases, inaccuracy in PhysX's solver and collision handling contributed to the difference in responses with motions along the x- and z-axis, from which response generated along the x-axis exhibited lower variation. Also, by further eliminating the variation in the PhysX-calculated penetration depth within the error-based collision response method, the contact plane response that restricts through-plane motions yielded a more stable feedback response.

While force data is not provided by PhysX, the Indirect Force Estimation method showed one way of deducing the pulling force between the haptic interface and the rest of the system. A distance joint with constant spring stiffness was introduced between the haptic interface and the object being pulled to compute the pulling force by measuring the elongation.

Haptic devices are often employed in demanding virtual reality applications, e.g. surgical simulation, where complex 3D interactions, accurate position data and force feedback are required. The proposed simulation platform provides a simple and direct technique to couple PhysX and OpenHaptics with minimal development time. Interactive 3D virtual surgical training applications will be developed on top of this platform in the future.



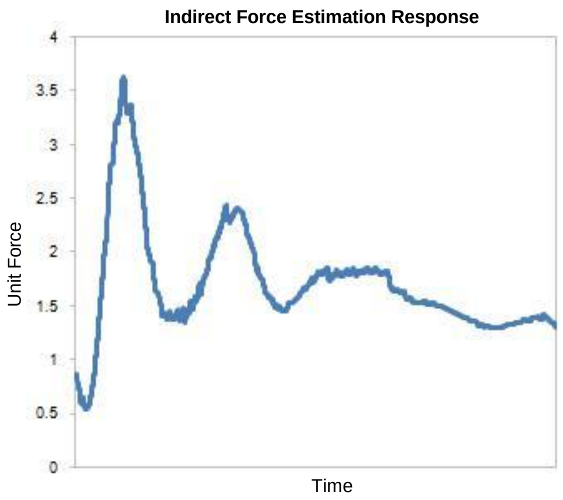


Figure 9. Lifting of a plate (left) and the corresponding feedback force computed with indirect force estimation (right).

Acknowledgement

This project is supported in part by the Research Grants Council of the HKSAR (Project No. PolyU5145/05E and No. PolyU5147/06E) and the Hong Kong Polytechnic University (Block Grant, Project Code 1-ZV2U).

References

[1] S. Garbaya and U. Zaldivar-Colado, "The Affect of Contact Force Sensation on User Performance in Virtual Assembly Tasks," Virtual Reality, 11:287-299, 2007.

[2] T.P. Grantcharov et al, "Randomized Clinical Trial of Virtual Reality Simulation for Laparoscopic Skills Training," British Journal of Surgery, 11(2): 146-150, 2003.

[3] C. Basdogan et al, "Haptics in Minimally Invasive Surgical Simulation and Training," Haptic Rendering-Beyond Visual Computing, IEEE Computer Society, March/April, pp 56-64, 2004.

[4] PhysX SDK 2.8. NVIDIA Corporation, 2008.

[5] OpenHaptics Toolkit Programmer's Guide, version 2. Sensible Technologies, 2005.

[6] A. Fischer, J.M. Vance, "PHANToM Haptic Device Implemented in a Projection Screen Virtual Environment," Ninth Eurographics Workshop on Virtual Environments, 2003.

[7] OpenHaptics Toolkit API Reference, version 2. Sensible Technologies, 2005.