# Multi-Rate Multi-Range Dynamic Simulation for Haptic Interaction

Ikumi Susa* Makoto Sato† Shoichi Hasegawa†

Tokyo Institute of Technology

**ABSTRACT**

In this paper, we propose a technique for a high quality haptic display working with a low update rate rigid body dynamics simulator. The proposed method uses two dynamics simulators. One has a low update rate for the whole virtual world and the other has a high update rate for the neighboring objects of the haptic pointer. In addition, the method calculates accelerance matrices of neighboring objects with regard to the contact forces added to these objects. We carried out a simulation and an experiment to check the effectiveness of the proposed method.

**Index Terms:** I.2.9 [ARTIFICIAL INTELLIGENCE]: Robotics—Kinematics and dynamics I.3.5 [COMPUTER GRAPHICS]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [COMPUTER GRAPHICS]: Three-Dimensional Graphics and Realism—Virtual reality I.6.8 [SIMULATION AND MODELING]: Types of Simulation—Parallel;

## 1 INTRODUCTION

Haptic interaction systems are one way of interacting with virtual worlds. Furthermore, systems combining haptic interfaces and physics simulators enable dexterous manipulation of virtual objects as in the real world. Therefore, it is expected that such systems could be applied to designing, training and entertainment.

Previous studies show us methods of creating haptic interaction systems. Most haptic interfaces have to be controlled at over 1 kHz for stability and display stiffness [9]. Generally, for haptic rendering, some sort of distance (penalty depth, coupling distance [4], *etc.*) is needed. The distance is calculated between the pose of the haptic interface and the pose of the haptic pointer (rigid body, *god-object* [15], *proxy* [14]). Therefore, when interacting with dynamic virtual worlds, physics simulators have to be updated at the same update rate of the haptic interfaces. However, because of limitations in computational resources, it is difficult to update haptic interfaces and physics simulators at the same high update rate.

We propose a system having multi-rate multi-range dynamic simulations to realize stable and stiff haptic display for dynamic virtual worlds filled with rigid bodies. The proposed system works at a computational cost of low update rate dynamics simulators and eliminates artifacts of delay caused by synchronization.

## 2 RELATED WORK

For the above problems, there are many studies [10] [7] which extend an *intermediate representation* [2]. The *Intermediate representation* proposed by Adachi *et al.* [2] enables interaction with a static virtual world with the processing of the haptic interaction system divided into the collision detection thread and the haptic rendering thread. Each thread is executed at a different update rate and synchronized at their slowest update rate. Hasegawa *et al.* [7] use an impulse rendered in a haptic thread to update movement of rigid bodies managed in a physics thread. An impulse makes the updating movement of rigid bodies stable.

In addition, other methods for multi-rate systems are proposed. One uses the *virtual coupling* [4] to connect a haptic interface and a haptic pointer as a rigid body [3][13]. Akahane *et al.* [3] implemented the haptic display of a 10 kHz update by interpolating and up-converting the force which was generated by the *virtual coupling*. They achieved a stiff, high resolution haptic display. Otaduy *et al.* [13] divided the haptic rendering thread into the haptic thread (high update rate) and the contact thread (low update rate). The haptic thread then calculates the coupling force and simulates the dynamics of the haptic pointer to realize a stable haptic display with a low mass value for the haptic pointer.

Another method, which uses the *constraint-based coupling* based on the *god-object* method [15], is proposed by Ortega *et al.* [12]. They introduce unconstrained and constrained acceleration of a haptic pointer to calculate the feedback force. Although the *virtual coupling* allows artificial friction or sticking, the *constrained-based coupling* does not allow this.

These methods enable us to interact with dynamic virtual worlds filled with large numbers of rigid bodies or polygons. However, these methods break the consistency of the time series between the user and the virtual world because different update rates for the threads delay communication. For example, when a user pushes a rigid body on a table via a haptic interface, a haptic thread renders a feedback force. However, the rigid body managed by a physics thread does not start moving until the next update of the physics thread. After the update, the user begins to perceive the movement of the rigid body. This delay makes the user feel a rigid body that is heavier than the configuration value of the mass and inertia of the rigid body. Therefore, these methods display the feedback force with an error caused by the delay.

For this problem, Glondu *et al.* [6] proposed the *haptic sub-world using a contact graph*. It allows a simulation of selected rigid bodies with high update rate and enables to render feedback force without artifacts produced by interpolation. However, the *haptic sub-world* is limited and it is difficult to include all rigid bodies which are in contact with each other. Although some methods for deformable objects with multi-rate approach have been proposed [11], these are not suitable for rigid bodies.

## 3 OVERVIEW OF THE PROPOSED METHOD

If the physics thread update rate is in the order of the haptic thread rate (1 kHz-), the problem of delay is solved. However, the physics simulator cannot complete the calculation for the whole virtual world in such a short period. Reflecting the user's input to the physics simulation without waiting for synchronization of the threads, we propose a method that simulates part of a virtual world with the update rate of the haptic thread like [6] Simulating part of the virtual world enables the user to reflect the physics simulation without synchronizing threads. This multi-threaded simulation realizes similar force feedback as well as single one. This method uses penalty based haptic rendering, so that the positions of the haptic pointers and the haptic interfaces are the same. Furthermore, the method is currently available for a rigid body dynamics simulator with convex collision and a 3-DoF haptic display.

*e-mail: susa@hi.pi.titech.ac.jp

†e-mail: (msato, hase)@pi.titech.ac.jp

IEEE World Haptics Conference 2011
21-24 June, Istanbul, Turkey
978-1-4577-0297-6/11/$26.00 ©2011 IEEE

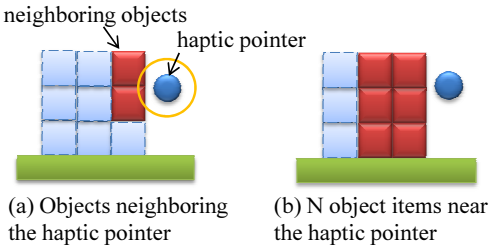


Figure 1: Range for applying local dynamics simulation

### 3.1 Range of the Local Dynamics Simulation

A dynamics simulation which is run in a haptic thread covers a part of the virtual world regarding the limitation of computational resources. Let us consider the range of local dynamics;

- Rigid bodies which are going to be in contact with a haptic pointer (Figure 1.(a)). We call these the objects neighboring the haptic pointer.
- The range from the haptic pointer to the N th rigid body. (Figure 1.(b)).

However, these ranges cause some problems. For example, in Figure 1.(a), objects neighboring the haptic pointer are simulated by the local dynamics simulator running in the haptic thread. Thus, it ignores the contact forces generated between neighboring objects and objects which are simulated by the global dynamics simulator in the physics thread. Because contact forces are not considered, neighboring objects will fall down due to gravity. Furthermore, in Figure 1.(b), the local dynamics simulator handles N rigid body items. However, when more rigid bodies come into contact with these N rigid bodies, the local dynamics simulator cannot calculate the feedback forces associated with this increase in rigid bodies. Even if this limitation is removed, the computation volume will restrict real time processing.

Therefore, we adopt the method of using neighboring objects for the haptic pointer (Figure 1.(a)) as the range for the local dynamics simulation. To consider the contact forces generated between neighboring objects and others, we introduce the accelerances of neighboring objects. The accelerance is a matrix which transforms force to acceleration. After the global dynamics simulation, the physics thread calculates the accelerance of the neighboring objects by carrying out another dynamics simulation (we call it a testing simulation) which simulates a further step in the global dynamics simulation. Finally, we send the accelerances of the neighboring objects to the haptic thread and simulate the movement of neighboring objects based on the accelerances and forces rendered by haptic rendering. This method considers all the contact forces which are added to neighboring objects and achieves the correct force feedback.

### 3.2 Architecture of the Proposed System

In this section, we describe the architecture of the proposed system (Figure 2).

- Physics thread

1. **Global dynamics simulation**
   Update the state of all rigid bodies.
2. **Find the neighboring objects of the haptic pointer (Section 4.1)**
3. **Calculate the accelerances of the neighboring objects (Section 4.2 and 4.3)**
   Run the testing simulation and calculate the accelerances of the neighboring objects from the differential of velocities.
4. **Synchronize with the local dynamics simulation (Section 6.1)**

- Haptic thread

a. **Update the states of the haptic pointer**
b. **Haptic rendering (Section 5.1)**
c. **Local Dynamics Simulation (Section 5.2)**
   Update the states of the neighboring objects of the haptic pointer based on the rendered forces and accelerances.
d. **Run a.-c. while the physics thread ends its step**
e. **Synchronize with the global dynamics simulation (Section 6.2)**
   Reflect the nonlinear forces (which are added to the neighboring objects) to the local dynamics simulation.

The timing of the synchronization is linked to each end of the physics thread's steps. For example, let us assume that the update rate of the haptic thread is $n$ times as fast as the physics thread. The synchronization is executed after $n$ haptic thread steps in parallel

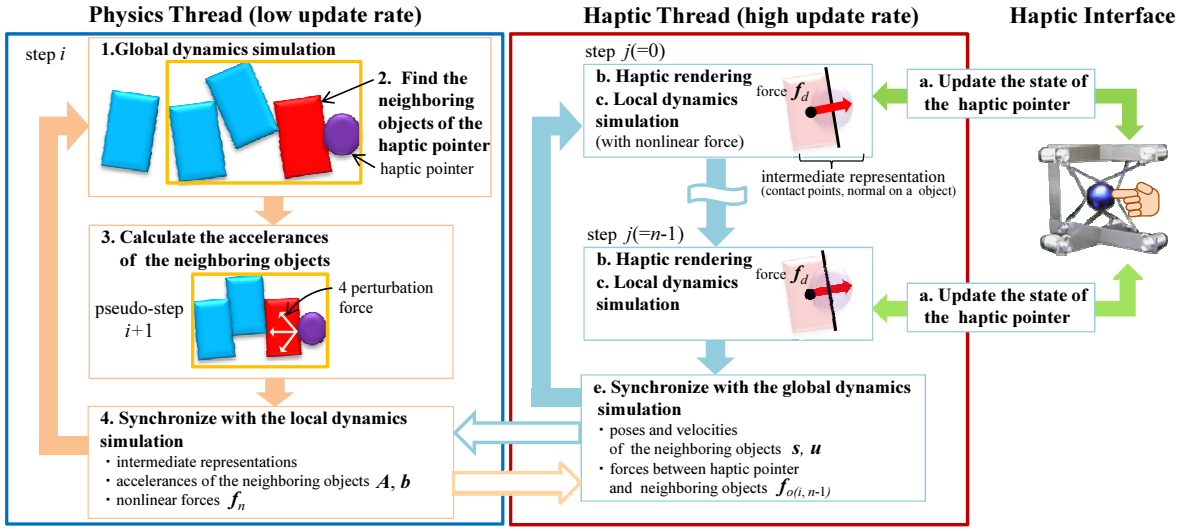


Figure 2: Architecture of proposed system

with a single physics thread step.

### 3.3 Notational Convention

We define characters that are used in the physics and haptic threads as follows.

- $G$: index that represents the notation in the physics thread
- $L$: index that represents the notation in the haptic thread
- $i$: step count of the physics thread
- $i, j$: step count of the haptic thread. While the physics thread runs the $i$ th step, the $j$ th step of the haptic thread is run. ($0 \leq j < n$. After synchronizing with the physics thread, we let $j = 0$.)

For instance, in the case that the physics thread runs the global dynamics simulation and updates a pose $\boldsymbol{s}^G_{(i-1)}$ of a rigid body at the ($i$)th step, the pose will be $\boldsymbol{s}^G_{(i)}$. In addition, if the haptic thread runs the local dynamics simulation and updates the pose $\boldsymbol{s}^L_{(i,j-1)}$ of a neighboring object at the ($i, j$)th step, it will be $\boldsymbol{s}^L_{(i,j)}$. In the next section, we describe in detail the processing of the threads.

## 4 Processing of the Physics Thread

As we described in Section 2.2, we note that the contact forces added by the other objects are necessary for simulating neighboring objects correctly. To take account of the contact forces, we introduce the analogy of accelerance. The accelerance is a matrix that transforms force to acceleration. We calculate the accelerances of each neighboring object. The accelerances of neighboring objects are calculated in the physics thread and sent to the haptic thread. To obtain the accelerances, we run a testing simulation after the global dynamics simulation in the physics thread by adding the perturbation force assumed from the feedback force. The testing simulation temporarily forwards a step of the global dynamics simulation for calculating the derivation of the current and the next velocities of the neighboring objects. The accelerances are then calculated from the perturbation forces and the derivation of velocities. In this section, we describe an algorithm for calculating the accelerances of the neighboring objects.

### 4.1 Finding the Neighboring Objects of the Haptic Pointer

The proposed method can use any collision detection algorithms which achieve information on nearest points and normals on the neighboring objects; e.g. *GJK* algorithm [5]. We put the information into *intermediate representations* [2] for haptic rendering (Section 5.1) and send them to the haptic thread at the synchronization term. The purpose of the above approach is to simplify the geometric model for haptic rendering and restrain the amount of data of convex shapes.

### 4.2 Accelerances of the Neighboring Objects

Because of the interposition of a user, the force $\boldsymbol{f}_o$, which is added to the neighboring objects of a haptic pointer, is not clear. Consequently, we consider the relationship between movement of the neighboring objects and the force $\boldsymbol{f}_o$. If the relationship is represented as a linear model, the movement of a neighboring object will be formulated as

$$\begin{aligned} \boldsymbol{M}\dot{\boldsymbol{u}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{K}\int \boldsymbol{u}dt + \boldsymbol{f}_e &= \begin{pmatrix} \boldsymbol{f}_o \\ \boldsymbol{r}\times\boldsymbol{f}_o \end{pmatrix} \\ &= \boldsymbol{J}_h\boldsymbol{f}_o. \end{aligned} \tag{1}$$

Where

- $\boldsymbol{M}, \boldsymbol{B}, \boldsymbol{K} (\in \mathbb{R}^{6\times 6})$: mechanical impedance matrix
- $\boldsymbol{u} (\in \mathbb{R}^6)$: velocity and angular velocity of a neighboring object
- $\boldsymbol{f}_o (\in \mathbb{R}^3)$: force added by the haptic pointer
- $\boldsymbol{f}_e (\in \mathbb{R}^6)$: external force except the force $\boldsymbol{f}_o$
- $\boldsymbol{r} (\in \mathbb{R}^3)$: point of application of the force $\boldsymbol{f}_o$
- $\boldsymbol{J}_h (\in \mathbb{R}^{6\times 3})$: matrix that transforms $\boldsymbol{f}_o$ into force and torque.

In this formula, $\boldsymbol{u}$ and $\int \boldsymbol{u}dt$ are not varied even if $\boldsymbol{f}_o$ changes. Thus, putting these constants together we transform (1) as

$$\begin{aligned} \dot{\boldsymbol{u}} &= \boldsymbol{M}^{-1}\boldsymbol{J}_h\boldsymbol{f}_o - \boldsymbol{M}^{-1}(\boldsymbol{B}\boldsymbol{u} + \boldsymbol{K}\int \boldsymbol{u}dt + \boldsymbol{f}_e) \\ &= \boldsymbol{A}\boldsymbol{f}_o + \boldsymbol{b}. \end{aligned} \tag{2}$$

Where $\boldsymbol{A} (\in \mathbb{R}^{6\times 3})$ is the accelerance of a neighboring object, which features mass and inertia, and $\boldsymbol{b}$ is the acceleration term derived from an external force such as gravity. In addition, $\boldsymbol{b}$ is not varied by the force added by the haptic pointer.

While a neighboring object is in contact with the other objects, connected with links or spring-dampers, the assumption of a linear model is valid as long as the situation continues. Though the cases that are transitions of the friction states or changes in the number of contacts, are not valid linear models, these cases occur only in the global dynamics simulation. Therefore it is difficult to take into account nonlinear changes in the local dynamics simulation.

Consequently, to maintain consistency between the global dynamics simulation and the local dynamics simulation, the global dynamics simulation calculates the nonlinear forces. The nonlinear forces are then sent to the haptic thread and reflected to the local dynamics simulation. This technique is described in Section 6.2.

### 4.3 Calculation of the Accelerance

To calculate the accelerance $\boldsymbol{A}$ and the acceleration term $\boldsymbol{b}$ of equation (2), the physics thread runs the testing simulation 4 times. The testing simulation forwards a single step to the global dynamics simulation. The reason that we selected 4 times of testing simulation is one for determining the acceleration term $\boldsymbol{b}$ and three for the accelerance $\boldsymbol{A}$ which is determined by 3 linear independent forces.

Let us show the calculation of the accelerance at the ($i$)th steps of the physics thread. First, the physics thread runs the testing simulation with a perturbation force of $\boldsymbol{f}^G_{p0} = (0\ 0\ 0)^T$ and achieves updated states of the rigid bodies on pseudo $(i+1)$th-step. The velocity of the neighboring object $\boldsymbol{u}^G_{0(i+1)}$ is then obtained. Here, we denote $\Delta t^G$ as a time step of the global dynamics simulation and transform equation (2) to a difference equation:

$$\boldsymbol{u}^G_{(i+1)} = \boldsymbol{u}^G_{(i)} + \left\{\boldsymbol{A}_{(i+1)}\boldsymbol{f}^G_{o(i+1)} + \boldsymbol{b}_{(i+1)}\right\}\Delta t^G. \tag{3}$$

Where $\boldsymbol{u}^G_{(i)}$ and $\Delta t^G$ are known, and substituting $\boldsymbol{u}^G_{0(i+1)}$ and $\boldsymbol{f}^G_{p0}$ for $\boldsymbol{u}^G_{(i+1)}$ and $\boldsymbol{f}^G_{o(i+1)}$, the acceleration term $\boldsymbol{b}_{(i+1)}$ is obtained. Secondly, we define 3 linear independent perturbation forces $\boldsymbol{f}^G_{p1}$, $\boldsymbol{f}^G_{p2}$ and $\boldsymbol{f}^G_{p3}$ based on the norm of the force $\boldsymbol{f}^L_{(i-1,n-1)}$, which is added by the haptic pointer at the $(i-1, n-1)$th step of the haptic thread, as

$$\begin{aligned} \boldsymbol{f}^G_{p1} &= (||\boldsymbol{f}^L_{o(i-1,n-1)}||, 0, 0)^T \\ \boldsymbol{f}^G_{p2} &= (0, ||\boldsymbol{f}^L_{o(i-1,n-1)}||, 0)^T \\ \boldsymbol{f}^G_{p3} &= (0, 0, ||\boldsymbol{f}^L_{o(i-1,n-1)}||)^T \end{aligned}. \tag{4}$$

Then the physics thread runs the testing simulation with each of the forces $\boldsymbol{f}^G_{p1}$, $\boldsymbol{f}^G_{p2}$ and $\boldsymbol{f}^G_{p3}$ to obtain the next update velocity of the neighboring object($\boldsymbol{u}^G_{1(i+1)}$ , $\boldsymbol{u}^G_{2(i+1)}$ and $\boldsymbol{u}^G_{3(i+1)}$). The reason that

we use the force $\boldsymbol{f}^L_{(i-1,n-1)}$ for the perturbation force is to obtain a more precise accelerance corresponding to current situation of haptic interaction. Then the equation (3) is transformed as

$$\begin{aligned}
\boldsymbol{A}_{(i+1)}\boldsymbol{f}^G_{p1} &= (\boldsymbol{u}^G_{1(i+1)} - \boldsymbol{u}^G_{(i)})\Delta t^{G^{-1}} - \boldsymbol{b}_{(i+1)} \quad (\equiv \boldsymbol{y}_1) \\
\boldsymbol{A}_{(i+1)}\boldsymbol{f}^G_{p2} &= (\boldsymbol{u}^G_{2(i+1)} - \boldsymbol{u}^G_{(i)})\Delta t^{G^{-1}} - \boldsymbol{b}_{(i+1)} \quad (\equiv \boldsymbol{y}_2) \\
\boldsymbol{A}_{(i+1)}\boldsymbol{f}^G_{p3} &= (\boldsymbol{u}^G_{3(i+1)} - \boldsymbol{u}^G_{(i)})\Delta t^{G^{-1}} - \boldsymbol{b}_{(i+1)} \quad (\equiv \boldsymbol{y}_3)
\end{aligned} . \quad (5)$$

Then combining equation (5) together and multiplying both sides of the equation by $[\boldsymbol{f}^G_{p1}\ \boldsymbol{f}^G_{p2}\ \boldsymbol{f}^G_{p3}]^{-1}$ to determine the accelerance $\boldsymbol{A}$:

$$\boldsymbol{A}_{(i+1)} = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \boldsymbol{y}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{f}^G_{p1} & \boldsymbol{f}^G_{p2} & \boldsymbol{f}^G_{p3} \end{bmatrix}^{-1}. \quad (6)$$

Finally, the physics thread sends the accelerance $\boldsymbol{A}$ and the acceleration term $\boldsymbol{b}$ to the $(i+1,0)$th step of the haptic thread and uses them in the local dynamics simulation (Section 5.2) until $j = n-1$.

## 5 Processing of the Haptic Thread

### 5.1 Haptic Rendering

To obtain the feedback force, we use a spring-damper model based on the depth of a haptic pointer, which penetrates the neighboring object represented as the *intermediate representation*[2]. For haptic rendering, we define the notations below.

- $k$, $d$: coefficient of the spring and damper.
- $\boldsymbol{x}_p$, $\dot{\boldsymbol{x}}_p$: position and velocity of a haptic pointer.
- $\boldsymbol{x}^L_o$: position of a contact point on a neighboring object which corresponds to a haptic pointer.
- $\dot{\boldsymbol{x}}^L_o$: velocity of a contact point on a neighboring object.

Then the feedback force $\boldsymbol{f}_d$ is represented by

$$\boldsymbol{f}_d = k(\boldsymbol{x}^L_o - \boldsymbol{x}_p) + d(\dot{\boldsymbol{x}}^L_o - \dot{\boldsymbol{x}}_p). \quad (7)$$

In addition, following Newton's third law, the force $\boldsymbol{f}^L_o$, which is added to the neighboring object, is given by $\boldsymbol{f}^L_o = -\boldsymbol{f}_d$.

### 5.2 Local Dynamics Simulation

The local dynamics simulation in the haptic thread simulates the movements of the neighboring objects. To simulate the movement of the neighboring object, we use the force $\boldsymbol{f}^L_o$, the accelerance $\boldsymbol{A}$ and the acceleration term $\boldsymbol{b}$, which are described in Section 4.2, 4.3 and 5.1. Then the movement of the neighboring object is expressed as

$$\dot{\boldsymbol{u}}^L = \boldsymbol{A}\boldsymbol{f}^L_o + \boldsymbol{b}. \quad (8)$$

For instance, the velocity and angular velocity of the neighboring object $\boldsymbol{u}^L_{(i,j)}$, which are updated at the $(i,j)$th step of the haptic thread, are represented as the difference equation with the time step of local dynamics simulation $\Delta t^L$:

$$\boldsymbol{u}^L_{(i,j)} = \boldsymbol{u}^L_{(i,j-1)} + \left\{ \boldsymbol{A}_{(i)}\boldsymbol{f}^L_{o(i,j)} + \boldsymbol{b}_{(i)} \right\} \Delta t^L. \quad (9)$$

Moreover, the updated pose of the neighboring object is given by

$$\boldsymbol{s}^L_{(i,j)} = \boldsymbol{s}^L_{(i,j-1)} + \boldsymbol{S}\boldsymbol{u}^L_{(i,j)}\Delta t^L. \quad (10)$$

Here, $\boldsymbol{s}$ represents the position $\boldsymbol{x}$, which is a rectangular coordinate, and the orientation $\boldsymbol{q} = (q_w\ q_x\ q_y\ q_z)^T$ which is a quaternion, as $\boldsymbol{s} = (\boldsymbol{x}\ \boldsymbol{q})^T$. $\boldsymbol{S}(\in \mathbb{R}^{7\times 6})$ is a matrix that transforms the angular velocity to a quaternion:

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{E} & 0 \\ 0 & \boldsymbol{Q} \end{pmatrix}, \boldsymbol{Q} = \frac{1}{2}\begin{pmatrix} -q_x & -q_y & -q_z \\ q_w & -q_z & q_y \\ q_z & q_w & -q_x \\ -q_y & q_x & q_w \end{pmatrix}. \quad (11)$$

Where $\boldsymbol{E}(\in \mathbb{R}^{3\times 3})$ is a unit matrix.

Finally, the haptic thread is executed $n$ times with the above processes. The haptic thread then sends poses and velocities of the neighboring objects and the rendered force $\boldsymbol{f}^L_{o(i,n-1)}$, which are for the testing simulation, to the $(i+1)$th step of the physics thread.

## 6 Synchronization between the Global and Local Dynamics Simulation

### 6.1 Reflecting the Results of the Local Dynamics Simulation to the Global Dynamics Simulation

The reflection is to replace the pose and velocity of neighboring objects, which are simulated by the global dynamics simulation, with neighboring objects, which are simulated by the local dynamics simulation. To reflect the results of the local dynamics simulation, the synchronization overwrites the velocities and poses of neighboring objects in the global dynamics simulation by the results of the local dynamics simulation.

For example, the reflection from the local dynamics simulation at the $(i,n-1)$th step of the haptic thread to the global dynamics simulation on the $(i)$th step of the physics thread, results in the replacements

$$\boldsymbol{u}^G_{(i)} = \boldsymbol{u}^L_{(i,n-1)}, \quad (12)$$

$$\boldsymbol{s}^G_{(i)} = \boldsymbol{s}^L_{(i,n-1)}. \quad (13)$$

Because the different update rate between the physics and the haptic thread makes different accuracy between global and local dynamics simulation, these replacements cause discontinuities in global dynamics simulation and visual display such as penetrations between neighboring objects and other objects. However, such penetrations are little and can be solved in global dynamics simulation gradually.

### 6.2 Reflecting the Nonlinear force to the Local Dynamics Simulation

The local dynamics simulation can determine the contact forces between the neighboring objects and other objects from accelerances $\boldsymbol{A}$ in equation (8). On the other hand, it cannot determine nonlinear forces such as impulsive and friction forces caused by contact and friction state changes. Without these forces, results of the local dynamics simulation replace some states of the global dynamics simulation such as Section 6.1, the neighboring objects will penetrate the other objects and the user cannot feel the impulsive forces which are added to the neighboring objects.

Therefore, to establish consistency between the global and local dynamics simulations, we calculate the nonlinear force $\boldsymbol{f}^G_n$ in the global dynamics simulation and reflect it in the local dynamics simulation at the first step of the haptic thread after synchronization. For example, when the nonlinear force $\boldsymbol{f}^G_{n(i)}$ is added to a neighboring object in the global dynamics simulation at the $(i)$th step of the physics thread, our method reflects $\boldsymbol{f}^G_{n(i)}$ to the local dynamics simulation at the $(i+1,0)$th step of the haptic thread:

$$\begin{aligned}
\boldsymbol{u}^L_{(i+1,0)} &= \boldsymbol{u}^L_{(i,n-1)} + \left\{ \boldsymbol{A}_{(i+1)}\boldsymbol{f}^L_{o(i+1,0)} + \boldsymbol{b}_{(i+1)} \right\} \Delta t^L \\
&\quad + \boldsymbol{M}^{-1}\boldsymbol{J}^G_{n(i)}\boldsymbol{f}^G_{n(i)}\Delta t^G.
\end{aligned} \quad (14)$$

Where $M^{-1}(\in \mathbb{R}^{6\times 6})$ is an inverse mass-inertia matrix and $J^G_n(\in \mathbb{R}^{6\times 3})$ is a matrix which transforms the force $\boldsymbol{f}^G_n$ to a vector of force and torque. The first and second terms on the right side of equation (14) correspond to equation (9). We then add the nonlinear force $\boldsymbol{f}^G_{n(i)}$ as the third term, to the right side of equation (14). This procedure establishes consistency between the global dynamics simulation and the local dynamics simulation.

## 7 Evaluation of the Proposed Method

To check the effectiveness of the proposed method, we did two evaluations. One is a haptic rendering with a computer simulation to see amount of rendered force. The other is measuring the computation time of the simulation.

### 7.1 Experimental System

The implemented system consists of an Intel(R) Core(TM)2 Duo CPU 2.33 GHz processor PC with Microsoft Windows Xp x86, a haptic interface SPIDAR-G6 [8] and a physics simulator Springhead2 [1] based on analytical method. In addition, we implemented 3 methods of haptic rendering. Method 1 updates simulation with high cycle (physics thread 1 ms, haptic thread 1 ms). Method 2 is proposed one which uses local dynamics simulation (physics thread 50 ms, haptic thread 1 ms). Method 3 is conventional one which uses impulse communication [7] (physics thread 50 ms, haptic thread 1 ms). The time steps of the simulation are the same as the cycle of the thread.

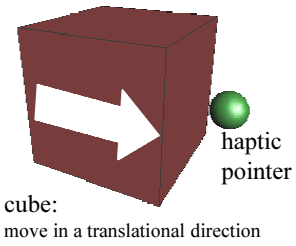


Figure 3: The virtual world for simulation

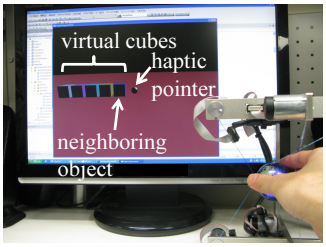


Figure 4: Measuring computation times of simulations

### 7.2 Evaluation of the Haptic Rendering

This Evaluation compares the feedback forces from the 3 methods and checks the effectiveness of the proposed method. To check effectiveness, we ran a computer simulation.

#### 7.2.1 Contents of the Simulation

The contents of the simulation are given by a virtual cube moving in a translational direction and this collides with a haptic pointer (Figure 3). While the virtual cube and the haptic pointer collide and rebound, the forces which are added to the virtual cube are recorded. To avoid unnecessary influences on the haptic renderings, the virtual cube is moved in one direction without rotating or the influence of gravity. In addition, the haptic pointer is fixed and does not move. We performed the above simulation using 3 methods with a virtual cube of mass 60 kg and velocity 0.017 m/s.

#### 7.2.2 Results of the simulation

Figure 5 shows the results of the simulation. Comparing the above 3 methods, Method 3 (using impulse communication) resulted in a longer contact time and a larger force. On the other hand, Method 1 (updated simulation with a high cycle) and Method 2 (using the local dynamics simulation) followed the same trajectory. In addition, Methods 1 and 2 had the same impulse value. This shows that the problem caused by the delay is solved in Method 2.

### 7.3 Evaluation of the Computation Time

The proposed method contains the testing simulation in addition to the global dynamics simulation. Therefore, we measured the computation time of the simulations and compared them with other methods.

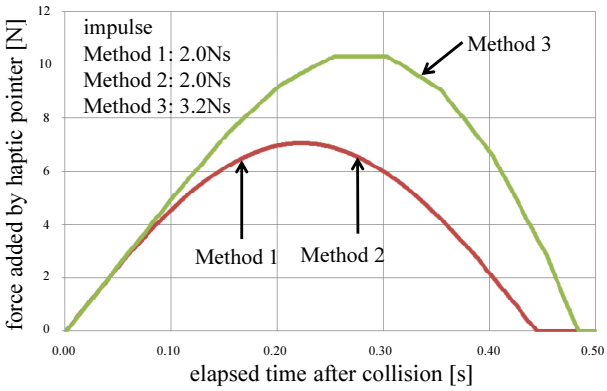


Figure 5: Results of the simulation

#### 7.3.1 Contents of the Experiment

We measured the computation time for each single step of the simulation, while pushing the virtual cubes with the haptic pointer (Figure 4). We used a CPU clock to measure the computation time. To observe changes in the computation time on number of virtual cubes and collisions, we measured two situations for each of the three methods. The first pushes a single virtual cube and the second pushes 10 virtual cubes. When pushing 10 virtual cubes, the cubes are not in contact with each other in the initial condition. The cubes progressively come in contact with each other once pushed by the haptic pointer.

#### 7.3.2 Results of The Experiment

Figure 6 and 7 shows the results of the experiment. The vertical axes of the graphs are computation duty factors. The computation duty factor is determined by dividing the measured computation time for a single step by the period of a simulation step. A computation duty factor beyond 1 means a simulation was not complete within a simulation time step.

Referring to the Figure 6 and 7.(a), the computation duty factor of each approach increased as the number of virtual cubes increases. Referring to Method 1, when the number of virtual cubes is 1, the computation duty factor is below 0.15, but when the number of virtual cubes is 10, the computation duty factor goes beyond 1 and Method 1 could not complete the calculation within 1 ms. The computation took 3 times the time step of Method 1. The force feedback was in fact unstable in Method 1 when there were 10 virtual cubes. On the other hand, with the other methods, the computation duty factors were less than 0.25 and the force feedbacks were steady.

Referring to the Figure 7.(b) which is the enlarged trajectory of Methods 2 and 3, Method 2 has the global dynamics and testing simulations. Comparing Method 2 (without testing simulation) and Method 3, there are no noticeable differences. On the other hand, the computation duty factor of Method 2 (including the testing simulation) was about 3-4 times as large as for Method 3. In fact, Method 2 ran the testing simulation 4 times when calculating the accelerances and acceleration terms of the neighboring objects. In addition, because the collision detection phase was not included in the testing simulation, the computing time of Method 2 did not reach 5 times that for the global dynamics simulation. Thus, the result for Method 2 is reasonable.

In conclusion, Method 2 is good at simulating the virtual world on a large scale compared to Method 1 and calculates more precise feedback forces compared to Method 3. Although our implementation of Method 2 could deal about 80 cubes, because of the computational resources of the local dynamics simulation, the force feedback was unstable when interact with 8 cubes at once. In addition,

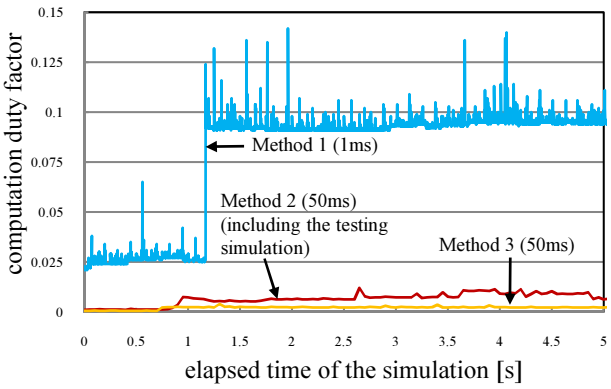


Figure 6: Result of measuring computation time with 1 virtual cube

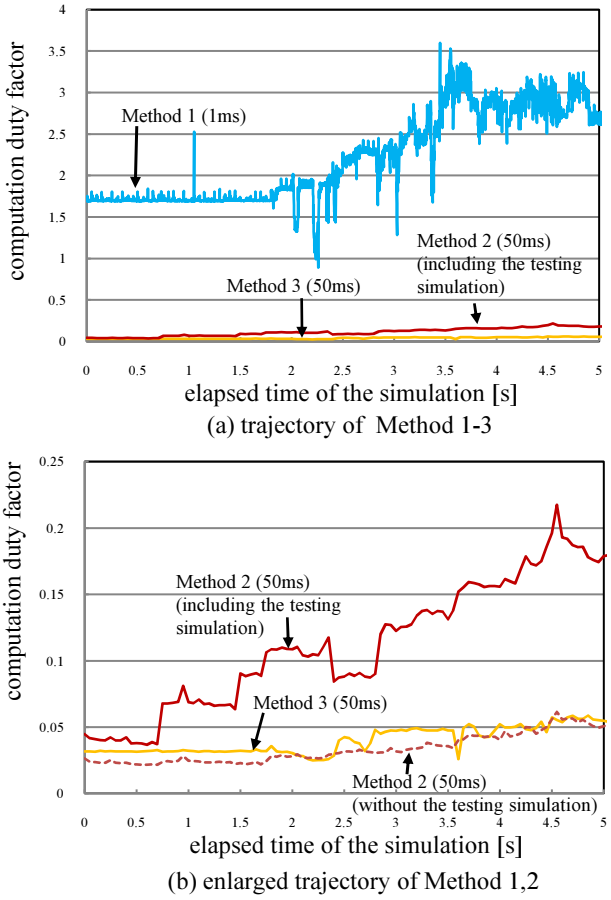


Figure 7: Result of measuring computation time with 10 virtual cubes

we confirmed our method realized stable dynamic haptic interaction with a virtual world filled with rigid bodies of various shapes and joints (Figure 8).

## 8 Conclusions

In this paper, we propose a multi-rate haptic interaction system using local dynamics simulation. To establish consistency between global dynamics simulation and local dynamics simulation, we proposed calculating the accelerances of objects which are near the haptic pointer and use it as a local dynamics simulation. Then we evaluated the effectiveness of proposed method with a simulation and an experiment.

Our method has several directions for future work. We would like to extend our method to 6-DoF haptic display for grasp manipulation and displaying friction force. Finally, though it might be difficult, we would like to examine our method applying to deformable objects.

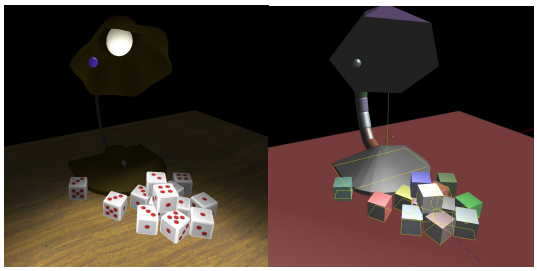

Figure 8: Haptic interaction with a light stand and dices

## References


[1] Springhead2. *http://springhead.info/wiki/*.

[2] Y. Adachi, T. Kumano, and K. Ogino. Intermediate representation for stiff virtual objects. *Virtual Reality Annual International Symposium*, pages 203–210, March 1995.

[3] K. Akahane, S. Hasegawa, Y. Koike, and M. Sato. A proposal of a high definition haptic rendering for stability and fidelity. *ICAT2006*, pages 162–167, November 2006.

[4] J. E. Colgate, M. C. Stanley, and J. Michael. Issues in the haptic display of tool use. *IEEE/RSJ International Conference on Intelligent*, pages 140–145, 1995.

[5] E.Gilbert, D. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in threedimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203, 1988.

[6] L. Glondu, 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. *Proceedings of Sixth Eurographics Workshop in Virtual Reality, Interaction and Physical Simulations*, 6191:51–56, 2010.

[7] S. Hasegawa, M. Ishii, Y. Koike, and M. Sato. Inter-process communication for force display of dynamic virtual world. *Proc. of the ASME-Dynamic Systems and Control Division-1999*, 67:211–218, 1999.

[8] S. Kim, J. Berkley, and M. Sato. A novel seven degree of freedom haptic device for engineering design. *VIRTUAL REALITY*, 6(4):217–228, 2003.

[9] L. Love and W. Book. Contact stability analysis of virtual walls. *Proc. Of Dynamic Systems and Control Division ASM*, pages 689–694, 1995.

[10] W. R. Mark, S. C. Randolph, M. Finch, J. M. V. Verth, and R. M. TaylorII. Adding force feedback to graphics systems: Issues and solutions. *Proc. of the 23rd annual conference on Computer graphics and interactive techniques*, pages 447–452, 1996.

[11] C. A. Mendoza and C. Laugier. Realistic haptic rendering for highly deformable virtual objects. *IEEE Virtual Reality Conference 2001*, page 264, 2001.

[12] M. Ortega, S. Redon, and S. Coquillart. A six degree-of-freedom god-object method for haptic display of rigid-bodies. *Proc. of IEEE Virtual Reality Conf.*, 2006.

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

[14] D. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. *SIGGRAPH 97 Proceedings*, 1997.

[15] C. B. Zilles and J. K. Salisbury. A constraint-based god object method for haptics display. *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems*, 1995.