第 14 回日本バーチャルリアリティ学会大会論文集 (2009 年 9 月)

大規模バーチャル世界のための高品質力覚提示

High Quality Haptic Display for Large Scale Virtual World

須佐育弥 [1)]，長谷川晶一 [1)]
Ikumi SUSA and Shoichi HASEGAWA

1) 電気通信大学 知能機械工学科
（〒 182-8585 東京都調布市調布ヶ丘 1-5-1, susa, hase@hi.mce.uec.ac.jp）

Abstract : In this paper, we present a technique for high quality haptic display working with a low update rate rigid body simulator. The proposal method uses two dynamics simulators. One is a low update rate dynamics simulator to simulate dynamics of whole virtual world and the other is a high update rate dynamics simulator to simulate dynamics of objects nearby the haptic pointer which the user controls (neighbor objects). Additionally, we calculate a mobility matrix of neighbor objects to consider contact force adding to the neighbor objects. We did a simulation and a experiment to check the effectiveness of the proposal method.

Key Words: *haptic rendering, rigid body simulation, multirate simulation*

1. はじめに

1.1 研究の背景

力覚提示システムを使用して，ユーザに高品質な力覚（バーチャル物体の詳細な形状，硬さ）を提示するためには，1kHz 以上の更新周波数で力覚インタフェースを制御し，力覚レンダリング（提示力の計算）をする必要がある [1]．力覚レンダリングではユーザが操作する力覚ポインタの位置，速度とユーザが触る物体の位置，速度が必要となる．ユーザが触る物体の位置，速度は物理シミュレータが管理しているため，物理シミュレータの更新周波数も力覚インタフェースの制御周波数に合わせなければならない．しかし，物理シミュレータの更新周波数を 1kHz 以上と高速にしてしまうと，計算量が膨大に増加する．このため，大規模で動的なバーチャル世界を構築することができない．

上記の問題に対して，物理シミュレータを低速で更新し，その結果を高速に補間しながら力覚レンダリングを行う研究 [2][3] はあるが，シミュレータとレンダリングの更新周期の違いにより，正確な力提示ができていない．

1.2 研究の目的

前節で述べた問題に対し，本研究は高品質な力提示を保ちつつも大規模なバーチャル世界を表現できるシステムを構築することを目的とする．

2. 提案手法

2.1 概要

本稿では，大規模なバーチャル世界において正確な力覚提示を実現する手法として，高速ループの力覚プロセスで実行させる局所的な物理シミュレーション (以下，局所シミュレーションと呼ぶ) と低速ループの物理プロセスで実行させるバーチャル世界全体の物理シミュレーション (以下，大域シミュレーションと呼ぶ) とで同期を取る手法を提案する．局所シミュレーションの更新周波数を力覚インタフェースの制御周波数に合わせることで，バーチャル世界の一部のみを高速にシミュレーションすることができる．

しかし，局所シミュレーションのみでは力覚ポインタで触れる近傍物体に接触する物体からの接触力を考慮していない．このため，近傍物体の運動を正確にシミュレーションできない．本研究では物理プロセスにおいて近傍物体に摂動力を加え，さらに 1 ステップシミュレーションを進めた結果から得られる近傍物体のモビリティを計算する．そして，近傍物体のモビリティを局所シミュレーションで使用することで接触力を考慮する．

提案手法を実現するためのシステム構成と処理の流れを図 1 に示す．

また，本稿では各プロセスステップで使用する記号を以下のように定義する．

- G: 物理プロセスで扱う変数の添え字
- L: 力覚プロセスで扱う変数の添え字
- i: 第 i 回目の物理プロセスステップ
- i, j: 第 (i, j) 回目の力覚プロセスステップ (第 i 回目の物理プロセスステップを行っている間の第 j 回目の力覚プロセスステップ ($0 \leq j < n$ で物理プロセスと同期後に 0 となる))

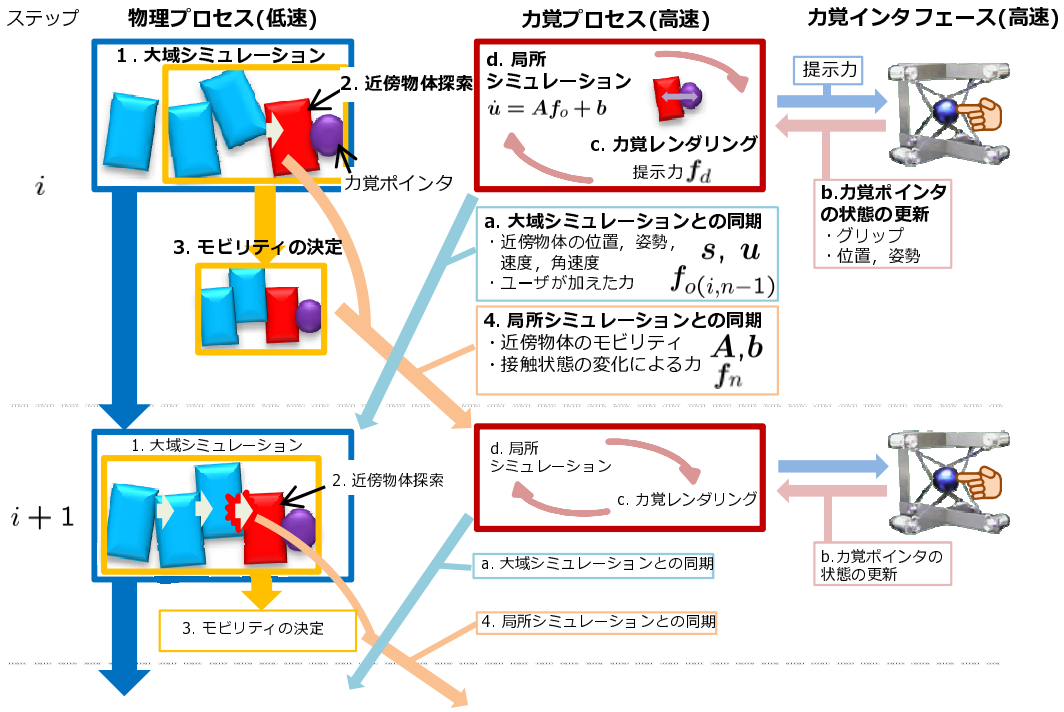

図 1: システム構成と処理の流れ

例えば，第 i 回目の物理プロセスステップで大域シミュレーションを行い，ある剛体の位置，姿勢 $\boldsymbol{s}^G_{(i-1)}$ を更新したとすると $\boldsymbol{s}^G_{(i)}$ となる．また，第 (i,j) 回目の力覚プロセスステップで局所シミュレーションを行い，近傍物体の位置，姿勢 $\boldsymbol{s}^L_{(i,j-1)}$ を更新したとすると $\boldsymbol{s}^L_{(i,j)}$ となる．

2.2 物理プロセスの処理

2.2.1 近傍物体のモビリティの導出

力覚ポインタが近傍物体に加える力 $\boldsymbol{f}_o$ は，ユーザの操作に依存するため定まらない。そこで，近傍物体の運動と $\boldsymbol{f}_o$ の関係を考える。$\boldsymbol{f}_o$ と近傍物体の運動に線形性を仮定すると，近傍物体の運動はメカニカルインピーダンスを用いて，

$$\boldsymbol{M}\dot{\boldsymbol{u}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{K}\int \boldsymbol{u}dt + \boldsymbol{f}_e = \begin{pmatrix} \boldsymbol{f}_o \\ \boldsymbol{r}\times\boldsymbol{f}_o \end{pmatrix} = \boldsymbol{J}_h\boldsymbol{f}_o \qquad (1)$$

と書ける．ただし，

$\boldsymbol{M},\boldsymbol{B},\boldsymbol{K}$: メカニカルインピーダンス行列 ($6\times6$ 行列)

$\boldsymbol{u}$: 近傍物体の速度，角速度 (6 次元ベクトル)

$\boldsymbol{f}_o$: ユーザが加える力 (3 次元ベクトル)

$\boldsymbol{f}_e$: $\boldsymbol{f}_o$ 以外の外力 (6 次元ベクトル)

$\boldsymbol{r}$: 近傍物体重心からの $\boldsymbol{f}_o$ の作用点

$\boldsymbol{J}_h$: $\boldsymbol{f}_o$ を剛体に加える力とトルクに変換するヤコビアン (6×3 行列)

である．

$\boldsymbol{f}_o$ が変化しても $\boldsymbol{u}$, $\int\boldsymbol{u}dt$ は変化しない．そこで式を変形させ，これらの定数をまとめると，

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

となる．$\boldsymbol{A}$ は近傍物体のモビリティ（メカニカルアドミタンス）の慣性成分を表す 6×3 行列で $\boldsymbol{b}$ は重力等の外力 (ユーザが加える力によって変化しない力) から由来する加速度項 (6 次元ベクトル) である．

近傍物体が他の物体と接触していたり，リンクやバネ，ダンパで接続されている場合でも，接触状態や接続が変化しない限り線形性の仮定は成り立つ．静止摩擦と動摩擦の切り替わりや接触の増減がある場合は線形性は成り立たないが，そのような変化は大域シミュレーションのステップでのみ起こると考え，局所シミュレーションでは扱わない．

2.2.2 テストシミュレーションによる近傍物体のモビリティの決定

本研究では大域シミュレーション後にもう 1 ステップシミュレーションを進めるテストシミュレーションを 4 回行うことで近傍物体のモビリティ $\boldsymbol{A}$ と重力等の外力に由来する加速度項 $\boldsymbol{b}$ を決定する．ここでは第 i 回目の物理プロセスステップでのモビリティの決定について説明する．

まず，テストシミュレーションのための近傍物体に加える摂動力を $\boldsymbol{f}^G_{o(i+1)} = (0,0,0)^t$ としてテストシミュレーションを実行すると $\boldsymbol{u}^G_{0(i+1)}$ が得られる．ここで，大域シミュレーションの刻み時間を Δt^G として，式 (2) を差分形にすると，

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

となる．$\boldsymbol{u}^G_{0(i+1)}, \boldsymbol{u}^G_{(i)}, \boldsymbol{f}^G_{o(i+1)}, \Delta t^G$ は既知なので式 (3) に代入すると，$\boldsymbol{b}_{(i+1)}$ を求めることができる．次に第 $(i-1, n-1)$ 回目の力覚プロセスステップから受信したユーザが実際に入力した力 $\boldsymbol{f}^L_{(i-1,n-1)}$ のノルムを使用して摂動力 $\boldsymbol{f}^G_{o(i+1)}$ を

$$\boldsymbol{f}^G_{o(i+1)} = \begin{cases} (||\boldsymbol{f}^L_{o(i-1,n-1)}||, 0, 0)^t (= \boldsymbol{f}^G_{o1}) \\ (0, ||\boldsymbol{f}^L_{o(i-1,n-1)}||, 0)^t (= \boldsymbol{f}^G_{o2}) \\ (0, 0, ||\boldsymbol{f}^L_{o(i-1,n-1)}||)^t (= \boldsymbol{f}^G_{o3}) \end{cases} \quad (4)$$

とし，それぞれについてテストシミュレーションを行い，1 ステップ先の近傍物体の速度，角速度を求める．ユーザが実際に入力した力を使用する理由は，より正確な近傍物体のモビリティを求めるためである．摂動力 $\boldsymbol{f}^G_{o(i+1)}$ のノルムが小さい場合には静止摩擦力により，近傍物体は動かず，大きい場合には動摩擦により近傍物体は動く．ここでテストシミュレーションにより得られた近傍物体の速度，角速度を $\boldsymbol{u}^G_{1(i+1)}, \boldsymbol{u}^G_{2(i+1)}, \boldsymbol{u}^G_{3(i+1)}$ とすると，これらはそれぞれ式 (3) より，

$$\begin{cases} \boldsymbol{A}_{(i+1)}\boldsymbol{f}^G_{o1} &= \frac{\boldsymbol{u}^G_{1(i+1)} - \boldsymbol{u}^G_{(i)}}{\Delta t^G} + \boldsymbol{b}_{(i+1)} \quad (= \boldsymbol{y}_1) \\ \boldsymbol{A}_{(i+1)}\boldsymbol{f}^G_{o2} &= \frac{\boldsymbol{u}^G_{2(i+1)} - \boldsymbol{u}^G_{(i)}}{\Delta t^G} + \boldsymbol{b}_{(i+1)} \quad (= \boldsymbol{y}_2) \\ \boldsymbol{A}_{(i+1)}\boldsymbol{f}^G_{o3} &= \frac{\boldsymbol{u}^G_{3(i+1)} - \boldsymbol{u}^G_{(i)}}{\Delta t^G} + \boldsymbol{b}_{(i+1)} \quad (= \boldsymbol{y}_3) \end{cases} \quad (5)$$

と書くことができる．さらに，式 (5) をまとめると

$$\boldsymbol{A}_{(i+1)} \begin{bmatrix} \boldsymbol{f}_{o1} & \boldsymbol{f}_{o2} & \boldsymbol{f}_{o3} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \boldsymbol{y}_3 \end{bmatrix} \quad (6)$$

となり，両辺右側から $[\boldsymbol{f}_{o1}\boldsymbol{f}_{o2}\boldsymbol{f}_{o3}]^{-1}$ をかけると，

$$\boldsymbol{A}_{(i+1)} = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \boldsymbol{y}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_{o1} & \boldsymbol{f}_{o2} & \boldsymbol{f}_{o3} \end{bmatrix}^{-1} \quad (7)$$

となることから，モビリティ $\boldsymbol{A}_{(i+1)}$ が得られる．得られたモビリティ $\boldsymbol{A}_{(i+1)}$ と重力等の外力に由来する加速度項 $\boldsymbol{b}_{(i+1)}$ を第 $(i+1, 0)$ 回目の力覚プロセスステップに送信し，2.3.2 節で説明する局所シミュレーションにて使用する．

2.3 力覚プロセスの処理

2.3.1 力覚レンダリング

力覚ポインタの位置と近傍物体の情報からユーザへの提示力の計算を行う．提示力は力覚ポインタが近傍物体に侵入した場合に，力覚ポインタと近傍物体を繋ぐ仮想のばねダンパモデルを用い，力覚ポインタが物体に侵入した量とその微分に比例した値を提示力及び近傍物体に加える力とする．近傍物体に加える力 $\boldsymbol{f}^L_o$ は

$$\boldsymbol{f}^L_o = k(\boldsymbol{x}_p - \boldsymbol{x}^L_o) + d(\dot{\boldsymbol{x}}_p - \dot{\boldsymbol{x}}^L_o) \quad (8)$$

となる．ただし，k はばね係数，d はダンパ係数，$\boldsymbol{x}_p$ は力覚ポインタの位置，$\dot{\boldsymbol{x}}_p$ は力覚ポインタの速度，$\boldsymbol{x}^L_o$ 近傍物体と力覚ポインタとの接触点，$\dot{\boldsymbol{x}}^L_o$ 近傍物体の速度である．

2.3.2 局所的な物理シミュレーション

2.3.1 節で計算した力 $\boldsymbol{f}^L_o$ を用い，近傍物体について局所シミュレーションを行う．局所シミュレーションで使用する近傍物体の運動を表す式は 2.2.1 節で導出した近傍物体のモビリティ $\boldsymbol{A}$，重力等の外力から由来する加速度項 $\boldsymbol{b}$ を使用して

$$\dot{\boldsymbol{u}}^L = \boldsymbol{A}\boldsymbol{f}^L_o + \boldsymbol{b} \quad (9)$$

となる．実際，第 $(i+1, j)$ 回目の力覚プロセスステップでの近傍物体の更新後の速度，角速度は式 (9) の差分形

$$\boldsymbol{u}^L_{(i+1,j)} = \boldsymbol{u}^L_{(i+1,j-1)} + (\boldsymbol{A}_{(i+1)}\boldsymbol{f}^L_{o(i+1,j)} + \boldsymbol{b}_{(i+1)})\Delta t^L \quad (10)$$

として表すことができる．Δt^L は局所シミュレーションの刻み時間である．近傍物体の位置，姿勢は上式に基づいて更新する．力覚プロセスステップを n 回行った後，近傍物体の位置，姿勢，速度，角速度とテストシミュレーションのための摂動力 $f^L_{o(i+1,n-1)}$ を第 $i+2$ 回目の物理プロセスステップへ送信する．

3. 評価

提案手法の有効性を確認するために，計算機シミュレーションによる力覚レンダリングの評価とシミュレーションに必要な計算時間を計測した．

3.1 実験の構成

実験を行うためのシステムは計算機 (Intel(R) Core(TM)2 Duo CPU 2.33GHz)，力覚インタフェース SPIDAR-G6，物理シミュレータ Springhead2 で構成した．

力提示の手法としてシミュレータを高速更新する手法，提案手法 (局所シミュレーションを行う手法)，従来手法 (高速補間力覚レンダリング [3]) の 3 種類をシステムに実装した．シミュレーションの刻み時間は各手法の物理プロセスの更新周期と同値でシミュレータを高速更新する手法 1ms，提案手法 50ms，従来手法 50ms とした．また，それぞれの手法の力覚プロセス更新周期は 1ms とした．

3.2 計算機シミュレーションによる力覚レンダリングの評価

提案手法が従来手法と比較して正確に力覚レンダリングができているのかを評価するための計算機シミュレーションを行った．

3.2.1 シミュレーション内容

重力のない 3 次元バーチャル世界上で立方体を並進移動させる．この立方体を球の形をした力覚ポインタに衝突させ，はね返る間に力覚ポインタが立方体に加える力を計測する．力覚レンダリングによる力を全て立方体に加えるために，力覚ポインタの位置更新，速度更新はせず，その場に留まるように設定した．

上記のシミュレーションを実装した 3 手法 (シミュレータを高速更新する手法，提案手法，従来手法) について立方体の質量 60kg，速度 0.017m/s で行った．

3.2.2 シミュレーション結果

図 2 にシミュレーション結果を示す．図 2 より従来手法はシミュレータを高速に更新する手法，提案手法に比べて，立方体と力覚ポインタとの接触時間が長く，立方体へ加える力が大きい．一方，提案手法はシミュレータを高速に更新する手法と同様の軌跡が得られている．また，各手法ごとの立方体と力覚ポインタが接触している間の力積はシミュレータを高速更新する手法，従来手法が 2.0Ns で従来手法

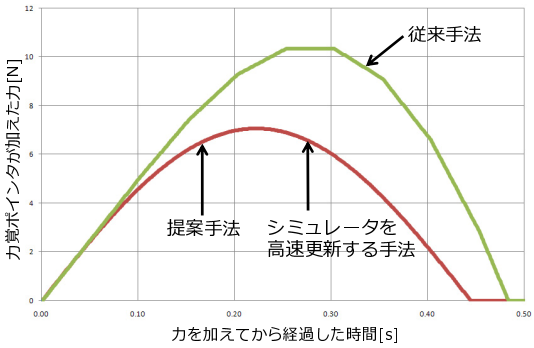

図 2: シミュレーション結果

は 3.2Ns であった．以上から，提案手法は従来手法と比較して正確な力を計算している．

3.3 計算時間の評価

提案手法では通常の物理シミュレーションに加え，テストシミュレーションを行っている．そこで，提案手法のテストシミュレーションがどの程度計算時間を増加させているか調査するために，シミュレーションの 1 ステップ当たりの計算時間を計測した．

3.3.1 計測内容

床の上にある立方体を力覚ポインタで押している時の物理シミュレーション 1 ステップ当たりの計算時間を計測した．計測はシミュレータを高速更新する手法，提案手法，従来手法で行い，提案手法についてはテストシミュレーションの計算にかかる時間も合わせて計測した．また，立方体の接触数により計算時間が変化するので，各手法について立方体の個数を 1，10 個にして計測した．個数が 10 個の場合，始めは立方体同士は接触しておらず，力覚ポインタで押していくうちに接触するようにした．

3.3.2 計測結果

立方体数が 10 の場合の計測結果を図 3 に示す．図 3 のグラフの縦軸は物理シミュレーション 1 ステップの計算時間を刻み時間で除算した割合で，刻み時間当たりの計算占有率である．

立方体の数が 1 個の場合は 3 手法全て計算占有率が 1 を下回っており，リアルタイムに計算を終えていた．立方体の数が 10 個の場合，図 3 上部をみるとシミュレータを高速更新する手法は計算占有率が 1 を超えるようになり，設定した刻み時間 1ms 内で計算を終えていない．一方，提案手法，従来手法については立方体 10 個の場合についても計算占有率が 0.25 を下回り，リアルタイムに計算を終えている．

次に，提案手法と従来手法の結果を拡大したもの (図 3 下部) についてみる．参考のため，提案手法の結果の内，テストシミュレーションを含まない大域シミュレーションのみを計測したものを破線で示す．提案手法 (テストシミュレーションを含む) と従来手法を比較すると提案手法 (テストシミュレーションを含む) は従来手法に比べ，約 3 倍から 4 倍となり，計算時間が増加している．以上から，提案手法は

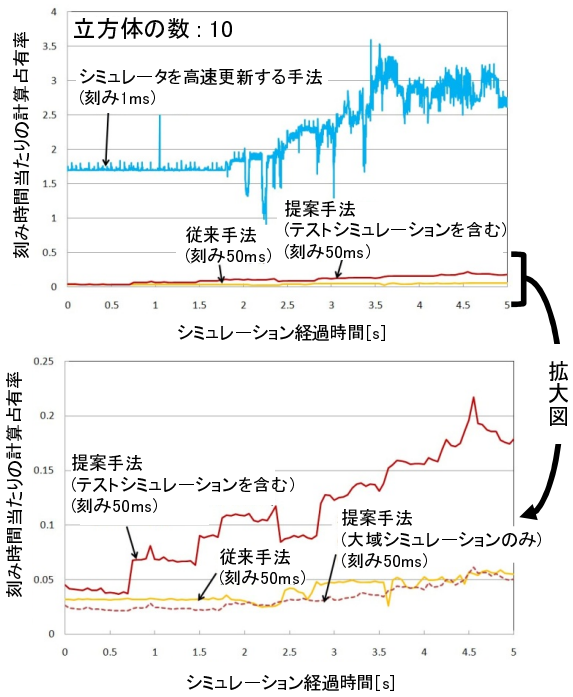

図 3: 立方体 10 個の場合の計算時間

従来手法に比べ計算量が増加するが，シミュレータを高速更新する手法に比べて多くの物体をリアルタイムにシミュレーションできることがわかった．

4. まとめと今後の展望

本稿では大規模なバーチャル世界上で高品質な力覚提示を実現するために，バーチャル世界内の力覚ポインタ近傍の物体のみを力覚インタフェースの制御周期と同様の周期で局所シミュレーションする手法を提案した．提案手法により，バーチャル世界のシミュレーションに必要な計算量を削減し，かつ高品質な力覚提示が可能となった．

今回は力覚ポインタと近傍物体の接触が 1 点であるため，力覚ポインタの形状により，接触点が断続的に変化し，6 自由度の力覚提示が行えていない．そこで，今後の展望として，提案手法を用いた 6 自由度の力覚提示を実現したいと考えている．

参考文献

[1] Lonnie Love, Wayne Book. “ Contact Stability Analysis of Virtual Walls ”, Proc. ofDynamic Systems and Control Division ASME, pp.689-694, 1995.

[2] Y. Adachi, T. Kumano, K. Ogino: Intermediate representation for stiff virtual objects; Proc. IEEE Virtual Reality Annual International Symposium, pp.203-210, 1995.

[3] 長谷川晶一, 石井雅博, 小池康晴, 佐藤誠: 力覚ディスプレイをもつ動的な仮想世界のための力積に基づくプロセス間通信; 電子情報通信学会論文誌 D-II, Vol.J82-D-II, No.10, pp.1758-1765, 1999.