The Virtual Reality Society of Japan

基礎論文

TVRSJ Vol.3 No.3, 1998

触覚用共通ソフトウェア(HIP)の開発

廣瀬 通孝*1 岩田 洋夫*2 池井 寧*3 小木 哲朗*1
広田 光一*4 矢野 博明*1 筧 直之*1

Development of Haptic Interface Platform(HIP)

Michitaka Hirose*1, Hiroo Iwata*2, Yasushi Ikei*3, Tetsuro Ogi*1,
Kouichi Hirota*4, Hiroaki Yano*1 and Naoyuki Kakehi*1

Abstract – **Haptic feedback plays an important role to recognize and manipulate objects in virtual environments. Recently, high quality haptic feedback devices have been commercially available. However, the basic software for haptic devices is not prepared sufficiently. In this paper, we propose Haptic Interface Platform(HIP) that is a common software library independent of the types of haptic devices to construct virtual environments with haptic feedback. HIP supports three types of haptic device types:Point Type, Surface Type and Texture Type. To generalize Haptic Interface Platform, we classified it into three functions: (1)device driver, (2)haptic renderer, (3)haptic simulation engines. HIP enables users to make complex virtual environments easily. Since HIP is compatible with many haptic displays, we will also be able to accumulate haptic software assets and reproduct software in making applications. In the last of this paper, we developed simple haptic models using HIP and experimented how humans would feel them for each haptic display. We verified that HIP made the similar sensation for each display.**

Keywords : **haptic library, haptic display, virtual environment, software library**

1. はじめに

仮想世界に物理法則を導入し、人間に触覚を提示することは、より自然な仮想環境を作り出すことを可能にする。特に触覚提示は、仮想物体の操作や認識等に非常に有効であると考えられ、触覚提示技術は手術シミュレーション、テレオペレーション等多くの分野への応用が期待されている。

近年、特に触覚提示研究に多くの研究者が取り組むようになり、質の高い触覚ディスプレイが様々開発されている。しかしハードウェアの整備が進む一方で、触覚用ソフトウェアの整備はあまりなされていない。現存する触覚ディスプレイの自由度やインタラクションの方式は多種多様であるが、一般的に触覚ディスプレイの制御には、高い制御周波数が必要であるとされ[1]、アプリケーション開発の際にも、触覚ディスプレイの制御速度等が優先されてきた。そのため、開発者の多くは、使用する個々の触覚ディスプレイに特化したソフトウェアを記述している。従って、他のディスプレイへのソフトウェアの移植ということは念頭になく、開発者はその蓄積、再利用が困難であった。

様々な方式の触覚ディスプレイで、ソフトウェアを共通に記述できるライブラリが整備されれば、

(1) 触覚ディスプレイ間に互換性が成立する

(2) 異種デバイス間での協調作業が可能になる

(3) 作成した触覚コンテンツを遠方に送信したり、保存蓄積したり、再利用できる

(4) 研究者はアプリケーションを容易に効率良く開発でき、生産性を高めることができる

という利点が生まれる。すなわち触覚情報の記述を一般化することにより、映像や音に加えて、触覚情報を電子メディアに組み込むことができるようになる。

最近では制御用の計算機が十分に速くなり、また触覚ディスプレイの制御も幾つかの典型的な方式が整ってきた。そこで本研究では、ソフトウェアの汎用性に主眼を置き、触覚ディスプレイに依存しない触覚用共通ソフトウェアHIPの開発を行うことを目的とした。HIPとは、Haptic Interface Platformの略で、様々な方式の触覚ディスプレイで共通に触覚情報を扱うための汎用ライブラリである。

ソフトウェアに汎用性を持たせるために、HIPでは、まずグラフィックスライブラリの処理を参考に触覚提示

*1 東京大学 工学系研究科
*2 筑波大学 構造工学系
*3 東京都立科学技術大学 工学部
*4 豊橋科学技術大学 情報工学系
*1 University of Tokyo
*2 University of Tsukuba
*3 the Tokyo Metropolitan Institute of Technology
*4 Toyohashi University of Technology

NII-Electronic Library Service

The Virtual Reality Society of Japan

処理のモジュール化を行い、現在開発されている主なデバイスを提示方式に着目して、分類整理を行った。また、多数の法則エンジンを持たせることで汎用性の高いライブラリを構築した。

さらに異なる触覚ディスプレイにおいて、提示した触覚情報を人間がどの程度同等に認識できるかの実験を行い、HIPの動作確認、及び開発意義の検証を行った。

なお、本論文に記されている触覚という意味は、力覚と皮膚感覚を含めた広義の触覚のことを表す。

2. 従来の研究

2.1 触覚用ソフトウェア

仮想環境を構築するソフトウェアツールは様々開発されているが、多くは視覚提示を目的としており、触覚ディスプレイと接続して用いることはできなかった。最近になってようやく触覚情報の生成に主眼を置いたソフトウェアツールの研究が報告されるようになってきた。その例としては、SensAble Technologies社のGHOST [2]、スタンフォード大のHL [3]、ノースカロライナ大のArmlib [4]が有名である。日本でも筑波大学でVECS [5]が開発されている。

GHOSTは、プリミティブやポリゴンによる形状定義、摩擦などの触覚効果機能をそれぞれ1つのクラスとして扱い、それを組み合わせてHaptic Scene Graphを構成し、触覚仮想世界を構築できるツールである。HLは、CGで表現されるような複雑なポリゴンモデルを触ったり、また動かしたりできる。しかし、これらのライブラリは特定の触覚ディスプレイ(PHANToM)に対してのみに有効であり、他の触覚ディスプレイとの互換性に関しては今のところ報告されていない。

複数の触覚ディスプレイに対応して設計されているライブラリとしては、ArmlibとVECSがある。Armlibでは、力覚ディスプレイの制御と仮想環境を管理するプログラムループを分割して、力覚ディスプレイの制御を行う。VECSは、ソフトウェアをデバイスドライバ、力覚レンダリング、法則エンジンを有するVECS本体部、アプリケーションプログラムの力覚プロセスにモジュール化し、複数のユーザをサポートした設計になっている。しかし、これらのソフトウェアは原理の異なる触覚ディスプレイに対する対応はまだ十分ではない。

HIPでは、複数ディスプレイへの対応をさらに拡張し、後述するような触覚生成の原理が異なるディスプレイに対しても対応できることを目的にしている。また触覚ディスプレイの方式と切り離して、種々の法則エンジンを体系的に用意していることが特徴である。

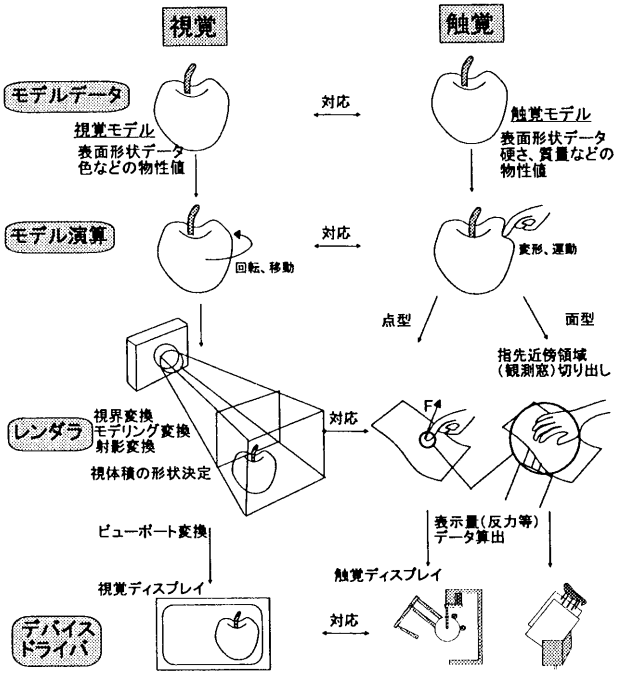


図1 視覚と触覚レンダリングの関係
Fig. 1 relation between visual and haptic rendering

3. HIPの基本設計

3.1 HIPの設計指針

仮想世界を構築しようとする場合、HIPではその基本ソフトウェア構造について、視覚とのアナロジで図1のように考えることにした。視覚の場合、(1)モデルが存在し、(2)モデルに対する演算が行われ、(3)カメラが撮影し、カメラの視点における透視画が求められ、最終的に(4)画面上に描画されるというプロセスをとる。(1)(2)(3)(4)のモジュールは、それぞれモデルデータ、モデル演算、レンダラ、デバイスドライバと呼ばれる。

同様にして、著者らは触覚用ソフトウェアを

(1) デバイスドライバ
(2) 触覚レンダラ
(3) モデル演算
(4) モデルデータ

の4つにモジュール化を行った[6]。それぞれの機能の詳細は次節以降で説明される。

3.2 デバイスドライバ

デバイスドライバとは、触覚ディスプレイを駆動するための機能であり、これは触覚ディスプレイ固有のものである。

現存する触覚ディスプレイは、原理や構造によって様々な提示方式がある。触覚用基本ソフトウェアを開発する上で、開発者がデバイスの差を意識することなく触覚情報を記述するために、各デバイス固有の問題はデバイスドライバの中だけにとどめている。

NII-Electronic Library Service

The Virtual Reality Society of Japan

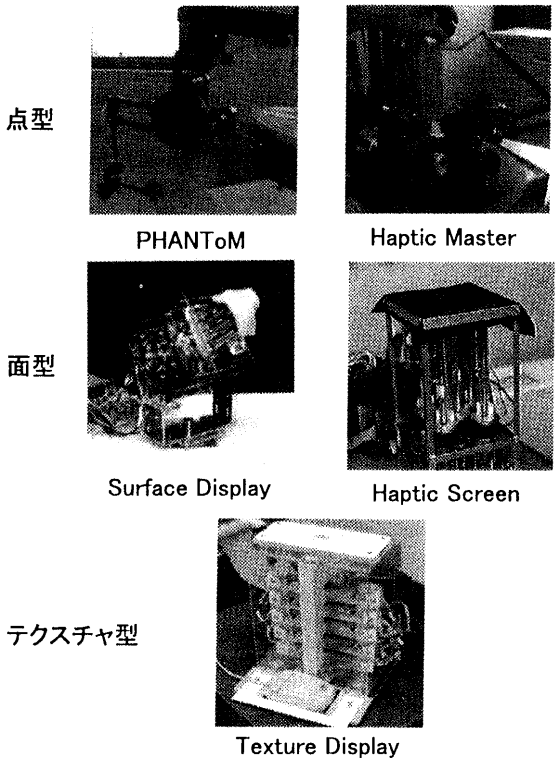


図2　点型、面型、テクスチャ型ディスプレイ
Fig. 2　haptic displays of point type, surface type and texture type

触覚ディスプレイ	型名	接触箇所	入出力方法
PHANToM	点型	空間上の1点	位置入力-力出力
Haptic Master	点型	空間上の1点	位置入力-力出力
Surface Display	面型	手の平全体	力入力-位置出力
Haptic Screen	面型	手の平全体	力入力-位置出力
Texture Display	テクスチャ型	指先腹部	位置入力-力出力

表1　触覚ディスプレイの分類
Table 1　classification of haptic displays

本研究では、現存する触覚ディスプレイの中から、SensAble Technologies社のPHANToM[7]、筑波大学で開発されたHaptic Master[8]、東京大学で開発されたSurface Display[9]、筑波大学で開発されたHaptic Screen[10]、東京都立科学技術大学で開発されたTexture Display[11]を取り上げ、HIPの動作確認を行った。図2に上記の各ディスプレイ装置を示す。また表1に今回用いた触覚ディスプレイの特徴を示す。図2、表1に記述されている点型、面型、テクスチャ型の分類に関しては、3.3で説明される。

3.3　触覚レンダラ

触覚レンダラとは、仮想空間の中のモデルという大域的情報からユーザの立場（視点になぞらえれば触点）における事象を局所的情報として切り出す機能である。具体的には、触覚ディスプレイの提示能力や可動範囲を考慮に入れて、仮想物体のどの部分を提示するかを決定する処理を行う。

触覚レンダラの機能は、視覚においてカメラモデルを設定して視体積を決める機能と同様で、指先モデルを設定し、指先近傍の領域（観測窓）を切り出し、触覚ディスプレイに提示する表示量（反力等）を計算することである。

また触覚レンダラは、後述のモデル演算機能とデバイスドライバの間で、各触覚ディスプレイの可動範囲や提示力の相違を吸収する機能を持つ。仮想空間で用いる世界座標系とデバイスで用いるデバイス座標系の変換を行い、デバイスの多様性を吸収する。このため触覚レンダラでは、デバイスを図2、表1で示す3種類のタイプに分類し、体系化している。

現存する触覚ディスプレイの多くは人間と仮想物体との接触方法に着目し、点型、面型、テクスチャ型の3つのタイプに分類することができる。レンダラを3つのタイプに分類することで、入出力形式さえ同じであれば新しい触覚ディスプレイが開発されたときにも、デバイスドライバを入れ替えるだけでレンダラ以降の部分には影響を与えないですむ。

点型触覚ディスプレイは、指先と仮想物体とのインタラクションが1点で行われるタイプである。点型では、指先位置を表す1点の位置と力とモーメントの関係が計算され、ユーザへ提示される。

面型触覚ディスプレイは、ユーザと仮想物体との接触境界が面であるタイプである。面型では、接触面における形状と力分布の関係が計算され、ユーザへ提示される。

テクスチャ型触覚ディスプレイは、仮想物体の凹凸など微細な表面形状を提示するタイプである。テクスチャ型では、ある領域の触覚刺激分布がユーザに提示される。

3.4　モデル演算機能

モデル演算機能とは、人間が仮想世界に対し何らかの触動作を働きかけた場合に仮想世界を更新し、人間に提示する表示量の計算を行う機能を持つ。

図3に人間が行う代表的な触動作とHIPで実装したモデル演算機能との対応を示す。触覚モデルには、表面形状を規定するサーフェスモデルと3次元的に分布するデータを規定するボリュームモデルの2つがあり、それぞれのモデルに対し、モデル演算を行う機能を持つ。

これらのモデル演算は、位置を入力とし、力を出力する「位置入力-力出力」の場合とその逆の「力入力-位置出力」がある。例えば、同じ変形運動を表現するために、指先の位置を計測し、そこまでの強制変位によって得られる反力を提示してもよいし、指先から加えられる力を計測し、これによって引き起こされる変形（位置）を提示してもよい。

一方で、触覚ディスプレイにも、表1で示したように「位置入力-力出力」と「力入力-位置出力」のディスプレイがある[12]。そのため各モデル演算は、デバイスドライバから渡される入力条件に従った演算方法で起動され

NII-Electronic Library Service

The Virtual Reality Society of Japan

ることになる。このようにモデル演算を起動方法によって整理することで、個々の触覚ディスプレイ、あるいは触覚提示方式の種類と切り離して体系化することができる。

3.4.1 サーフェスモデル演算

・表面接触モデル演算機能

この機能は他の演算機能の上位に位置し、仮想物体と指先が接触しているか否かについての接触判定を行う機能である。接触していた場合には、どの物体のどの部分にどれだけ触れているか、また物体に加えられた力の情報を含む「接触判定データ」を求め、下位の機能へ出力する。

この演算機能は定義された表面接触モデルとの間で行われる。表面接触モデルは、仮想モデルの表面形状を定義するモデルであり、ポリゴンのように点で面を定義する離散型と球などのように数式で面を定義する連続型を用いて定義される。

・変形運動モデル演算機能

この機能は、物体の変形運動をリアルタイムにシミュレーションする機能である。一般的に、触覚提示には非常に高い周波数による制御が必要とされ[1]、変形運動の厳密さと触覚提示の制御速度はトレードオフの関係にある。そこでHIPでは物体の変形運動をリアルタイムで計算する方法に、バネモデルによって近似的に変形を計算する方法と、有限要素法により厳密に計算を行う方法を用意している。

(1) バネモデル演算

バネモデルは、仮想物体の表面を幾つかの格子に分割し、格子点同士をバネで結んだモデルで定義される。ある格子点が変位するとバネを通じて周りの格子点に変位が発生し、仮想物体全体が変形する。

(2) 有限要素モデル演算

有限要素モデルは、対象とする物体をメッシュ状に分割したモデルで定義される。有限要素法によるマトリクス方程式を解くことにより、物体の変形運動の計算を行う。有限要素法の処理をリアルタイムで行うには現状ではかなり負荷を要するので、物体の対称性を利用して表面接触計算は3次元で行いながら、変形計算は2次元で行う等の工夫が必要である。

・剛体運動モデル演算機能

(1) 剛体非把持運動モデル演算

これは非把持状態における剛体の運動を計算する機能である。仮想物体に力が作用されると、剛体座標系で記述されたニュートンの運動方程式およびオイラーの運動方程式から、仮想物体の加速度、速度、位置の変化を計算し、運動のシミュレーションを行う。

(2) 剛体把持運動モデル演算

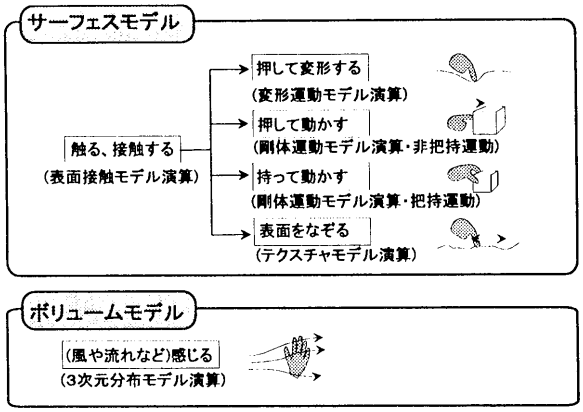


図3 人間の触動作とモデル演算機能
Fig. 3 human haptic motion and model simulation

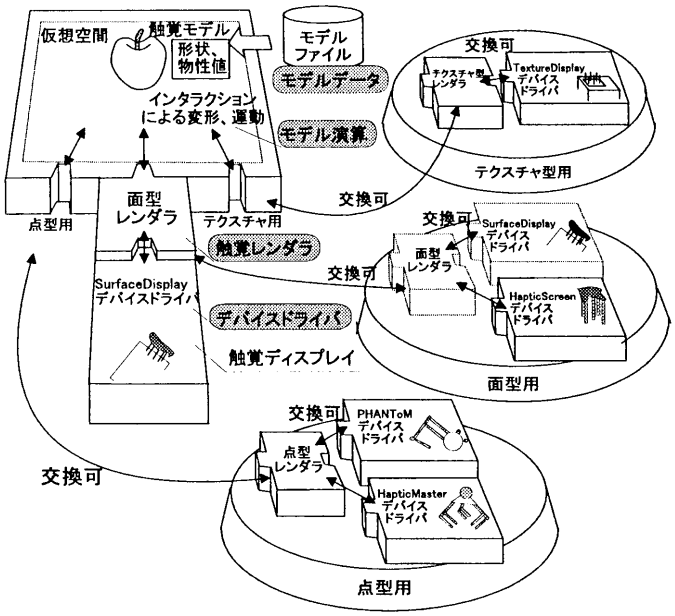


図4 HIPの概念図
Fig. 4 concept of Haptic Interface Platform

これは把持状態における剛体の挙動と操作者への反力を計算する機能である。指先の動きに追従した対象物の運動を計算し、またそれをもとに物体に作用している力及びモーメントの計算を逆動力学に基づいて行う。

・テクスチャモデル演算機能

この機能は物体表面の微小な凹凸、つまり物体表面の質感、肌理(テクスチャ)を表現する機能である。テクスチャモデルは、面状の触覚強度分布が矩形領域のマップとして表現される。テクスチャモデルには連続型と離散型があり、離散型では画像のテクスチャと同様にピクセルに相当する触覚要素の集まりで記述され、連続型では領域を2次元の座標の関数として記述される。

3.4.2 ボリュームモデル演算

・3次元分布モデル演算機能

この機能は、形状を持つ物体とは異なり、温度や圧力、

NII-Electronic Library Service

The Virtual Reality Society of Japan

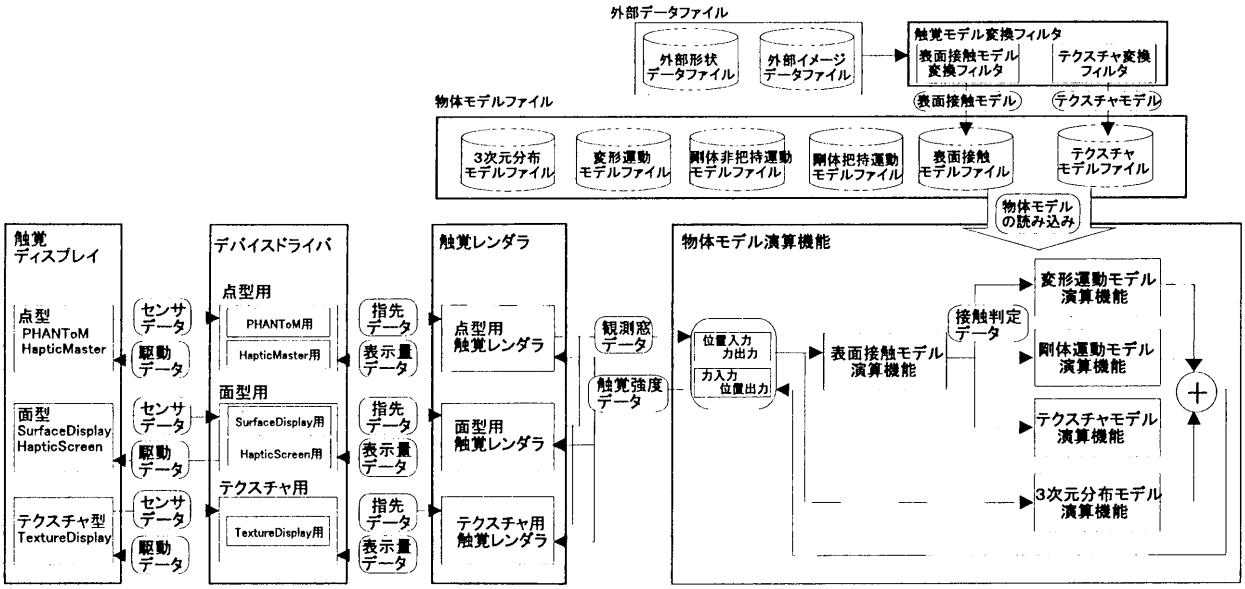


図5　HIPの全体構成
Fig. 5　whole structure of Haptic Interface Platform

流れ場のような空間に分布するデータを触覚表現する機能である。モデル構造には離散型と連続型があり、離散型は3次元メッシュ上におけるデータ値が書かれた3次元分布モデルファイルを読み込むことにより実現し、連続型は3次元の座標の関数を定義することで実現する。

3.5　触覚モデルデータ

3.4で説明したモデル演算は、定義されたモデルの表面形状や物性値に従い計算される。これらの触覚モデルデータは、すべてモデルファイルに記述される。つまり、モデルファイルに記述されているモデルの属性を変えることで、異なるモデルを生成することが可能になる。例えば、バネモデルファイルには、格子点の質量、格子間の弾性係数、減衰係数、復元力係数などの変形モデルのパラメータの値が記述されているが、これらの値を変えることで、様々な触覚表現が可能となる。

また、外部のファイルを読み込んで、触覚モデルを生成する機能も、高度な触覚世界を生成する上で重要である。この機能により、既に汎用的に用いられている視覚モデルから触覚モデルに容易に変換できるようになる。具体的には以下の2つの機能を持つ。

・表面接触モデル変換フィルタ

これは対象物体の表面を記述するDXFファイルやVRMLファイルを読み込み、表面接触モデルに変換する機能である。例えばインターネットからVRMLファイルを入手すれば、この機能により変換してすぐに触ることができるようになる。

・テクスチャ変換フィルタ

これは、対象物体の表面の画像イメージを記録したTARGA形式のファイルを読み込んで、テクスチャモデルファイル形式で出力する機能である。

図4にHIPの概念図を示すが、触覚ディスプレイを提示方式により体系的に分類し、仮想空間における演算部を触覚ディスプレイから切り離すことで、汎用性を高めることが可能になる。ソフトウェアを他の触覚ディスプレイに移植する場合には、デバイスドライバ、型が異なるときにはレンダラを入れ替えることで、移植が可能になる。

以上の機能をまとめると、HIPの全体構成は図5のようになる。各モデル演算機能は、同時に並列して用いることも可能である。

4.　HIPの実装方法

4.1　システム構成

まず、ユーザは仮想環境で提示する触覚情報、用途等を考えて、使用する触覚ディスプレイの選定を行う。HIPでは、触覚ディスプレイの制御用計算機PCとホスト計算機WSの2台を用いて触覚ディスプレイの制御を行う。PC上ではデバイスドライバが動作し、WS上ではデバイスドライバの通信部分および触覚レンダラ、モデル演算機能が動作する。PCとWS間では、デバイスにより最適な通信手段（RS232CあるいはTCP/IP）を用いて必要なデータ通信を行うが、通信はデバイスドライバが行うので、ユーザがプログラムを開発する上で意識する必要はない。

HIPは触覚表示を行うライブラリであり、視覚表示を行う場合はWS上で別プロセスに分けて行う。これは、視覚表示の計算負荷によって触覚制御の周期が引きずられないためである。また視覚モデルと触覚モデルは、3.5で説明したように共通なデータを用いることができる。2つのプロセス間では、モデルの形状などの必要なデー

NII-Electronic Library Service

The Virtual Reality Society of Japan


```
1: void main(void)   /* 触覚生成メイン関数 */
2: {
3:   HIPPDevice    pdev;     /* デバイス設定データ(点型用) */
4:   HIPPFinger    finger;   /* 指先位置データ(点型用) */
5:   HIPWindow     window;   /* 観測窓データ */
6:   HIPContact    contact;  /* 接触判定データ */
7:   HIPIntensity  intense;  /* 触覚強度データ */
8:   HIPPValue     value;    /* 表示量データ(点型用) */
9:
10:  HIPSurfModel   *surfmodel;          /* 表面接触モデル */
11:  HIPSpringModel *springmodel=NULL; /* バネモデル */
12:  HIPSimuContext simu;    /* シミュレーション定義 */
13:
14:  simu.dt=0.3; /* 仮想空間の時間の進むスピード */
15:  /* 表面接触モデルファイルの読み込み */
16:  surfmodel = hipSurfModelLoad("cat.hip",&simu);
17:  /* バネモデルファイルの読み込み */
18:  springmodel=hipSpringModelLoad(surfmodel,simu.allnodenum,"spring.dat");
19:
20:  if(hipPHANToMOpen(TCP_PORT)==-1) hapticflag=0;/*PHANToMOpen*/
21:  hipPLoadDevice("phantom",&pdev);/* デバイス能力データ読み込み */
22:  hipPSetObserveFunc(phwind);      /* 観測窓作成関数の登録 */
23:  hipPSetRenderFunc(phint);        /* 切出変換関数の登録 */
24:  hipPWindow(&pdev,&finger,&window); /* 観測窓データへデバイス数登録 */
25:  hipContactInit(&window,&contact);  /* 接触判定データの初期化 */
26:
27:  while(hapticflag)
28:  {
29:    /* 指先位置データの取得(PHANToMドライバ) */
30:    if(hipPHANToMGetFinger(&finger)==-1) break;
31:
32:    /* 指先位置データを観測窓データへ変換(点型レンダラ) */
33:    hipPWindow(&pdev,&finger,&window);
34:
35:    /* 表面接触モデル演算 */
36:    hipSurfModel_Calc(&simu,&window, surfmodel, &contact);
37:    /*バネモデルによる変形運動演算 */
38:    hipSpringModel_Calc(&simu,&contact,surfmodel,springmodel,&intense);
39:
40:    /* 触覚強度データを表示量データへ変換(点型レンダラ)*/
41:    hipPRender(&pdev, &intense, &value);
42:
43:    /* 表示量データの出力(PHANToMドライバ) */
44:    hipPHANToMPutValue(&value);
45:  }
46: }
```

(a)点型ディスプレイ(PHANToM)の場合

```
1: void main(void)   /* 触覚生成メイン関数 */
2: {
3:   HIPSDevice    sdev;     /* デバイス設定データ(面型用) */
4:   HIPSFinger    finger;   /* 指先位置データ(面型用) */
5:   HIPWindow     window;   /* 観測窓データ */
6:   HIPContact    contact;  /* 接触判定データ */
7:   HIPIntensity  intense;  /* 触覚強度データ */
8:   HIPSValue     value;    /* 表示量データ(面型用) */
9:   int           dispid;
10:  HIPSurfModel   *surfmodel;          /* 表面接触モデル */
11:  HIPSpringModel *springmodel=NULL; /* バネモデル */
12:  HIPSimuContext simu;    /* シミュレーション定義 */
13:
14:  simu.dt=0.3; /* 仮想空間の時間の進むスピード */
15:  /* 表面接触モデルファイルの読み込み */
16:  surfmodel = hipSurfModelLoad("cat.hip",&simu);
17:  /* バネモデルファイルの読み込み */
18:  springmodel=hipSpringModelLoad(surfmodel,simu.allnodenum,"spring.dat");
19:
20:  dispid=hipSurfaceDisplayOpen("violet", 20000) /* デバイス初期化 */
21:  hipSLoadDevice("sdisplay",&sdev);/* デバイス能力データ読み込み */
22:
23:  hipWindowSdevInit(&sdev,&window); /* 観測窓モデル作成 */
24:  hipSWindow(&sdev,&finger,&window); /* 観測窓データへデバイス数登録 */
25:  hipContactInit(&window,&contact);  /* 接触判定データの初期化 */
26:
27:  while(hapticflag)
28:  {
29:    /* 指先位置データの取得(SurfaceDisplayドライバ) */
30:    hipSurfaceDisplayGetFinger(dispid, &finger);
31:
32:    /* 指先位置データを観測窓データへ変換(面型レンダラ) */
33:    hipSWindow(&sdev,&finger,&window);
34:
35:    /* 表面接触モデル演算 */
36:    hipSurfModel_Calc(&simu,&window, surfmodel, &contact);
37:    /*バネモデルによる変形運動演算 */
38:    hipSpringModel_Calc(&simu,&contact,surfmodel,springmodel,&intense);
39:
40:    /* 触覚強度データを表示量データへ変換(面型レンダラ)*/
41:    hipSRender(&sdev, &intense, &value);
42:
43:    /* 表示量データの出力(SurfaceDisplayドライバ) */
44:    hipSurfaceDisplayPutValue(&value);
45:  }
46: }
```

(b)面型ディスプレイ(Surface Display)の場合

初期化
デバイスドライバ
レンダラ
モデル演算（共通）
レンダラ
デバイスドライバ

図6 HIPのサンプルメイン関数
Fig. 6 sample source code of HIP

タを共有メモリを介して共有する。

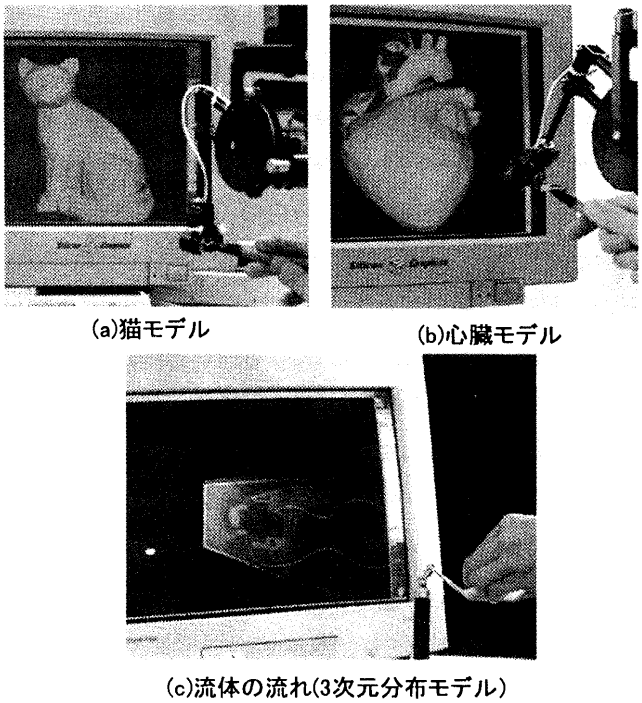
4.2 プログラムの流れ

HIPを用いたプログラムのメイン関数作成の例を図6に示す。図6(a)のプログラムにより、ユーザはPHANToMを用いて図7(a)に示すような猫のモデル(ポリゴン数671)を触り、押して変形させることができる。このプログラムでは、PHANToM用デバイスドライバ、点型レンダラ、表面接触モデル演算機能、バネモデル演算機能を用いている。以下にこのプログラムを順を追って、説明する。

*l*3～*l*25では、各種初期設定を行なっている。具体的には、表面接触モデル、バネモデルの読み込み(*l*15～*l*18)、デバイスドライバの初期化(*l*20)、デバイスの可動範囲、提示最大力などが記述されたデバイス能力記述ファイルの読み込み(*l*21)、点型レンダラの処理を行う関数の登録(*l*22～*l*23)などを行う。

*l*27以降が、サーボループ時の処理で、実行時にはこれらの処理が繰り返される。

1. デバイスドライバがデバイスからのセンサデータを読み、指先位置データを算出する(*l*30)。
2. 点型レンダラにより、指先位置データを仮想空間座標系で記述される観測窓データに変換する(*l*33)。
3. 表面接触モデル演算機能により、モデルとの接触判定を行い、接触判定データを求める(*l*36)。



(a)猫モデル　(b)心臓モデル

(c)流体の流れ(3次元分布モデル)

図7 HIPで触覚表現できるモデルの例
Fig. 7 sample models represented haptics by HIP

4. バネモデル演算機能により、形状の変形計算をするとともに触覚強度データを算出する(*l*38)。

NII-Electronic Library Service

The Virtual Reality Society of Japan

5. 点型レンダラにより、触覚強度データをデバイス座標系で記述される表示量データに変換する(*l*41)。

6. デバイスドライバにより、表示量データをデバイスに出力する(*l*44)。

図7(b)に示すような心臓のモデルを触覚表現するならば、*l*16で心臓を表す表面接触モデルを読み込むだけでよい。また、図7(c)に示すような流体の流れ場を触覚表現するならば、*l*18を

hipVolModel_Load("test.dat",&vol);

に変更して(volは3次元分布モデル構造体の変数)、流れ場を表す3次元分布モデルファイルを読み込み、*l*36～*l*38を

hipVolModel_Calc(&simu,&window,&vol,&intense);

に変更して、3次元分布モデル演算機能をコールするだけでよい。剛体運動、テクスチャモデル演算機能に関しても同様に対応する関数を組み込んで、触覚表現できる。

PHANToMではなく、他のデバイスを用いる場合には、デバイスドライバに関する部分、異なる型のデバイスを用いる場合には、加えてレンダラの部分を書き換えるだけで、同一のモデルに触ることが可能になり、非常に移植性が高いものとなった。図6(b)は、(a)のプログラムをSurface Displayに移植したプログラムである。初期化、デバイスドライバ、触覚レンダラの部分を入れ替えるだけで、移植が容易に実現できる。テクスチャ型のディスプレイに対しても、同様に移植を行なうことができる。

5. 評価実験

5.1 実験内容

HIPの有効性を検証する意味で、生成される触覚世界の同等性を評価する実験を行った。

触覚ディスプレイは、それぞれ摩擦や慣性等の機械的特性や提示可能な力の範囲等が異なるが、これらの異種デバイスを用いて同等の感覚を生成できるかどうか調べることが、ここでの目的である。本実験では、触覚表現として硬さ/柔らかさの変化を取り上げ、知覚実験を行った。

実験には、図8に示すような格子間をバネとダンパで結ぶバネモデルによって弾性変形する立方体のモデル(頂点数7×7×2)を用いる。この変形モデルをPHANToM、HapticMaster、HapticScreenの3つの触覚ディスプレイで被験者に体験してもらう。被験者は、PHANToMではペンを、HapticMasterではグリップ(直径35mm)を、把持してモデルを触り、またHapticScreenでは、実際に提示される面を指先腹部で触ってもらった。

仮想モデルには、同一の形状で弾性係数が異なるものを10個用意し、マグニチュード推定法により、被験者の受ける硬さの感覚量の変化を測定した。最も弾性係数が大きいモデルを触った感覚を基準値100とする。各触覚ディスプレイにおける基準モデルの弾性係数と減衰係数は、表2に示す通りであり、それぞれ使用可能な提示力の範囲は大きく異なる。残りのモデルは、0から基準値までを10等分した弾性係数をそれぞれ持つモデルである。基準値と同じモデルを含む10個のモデルに対して、その基準値と比べていくつに感じられるかを、口頭により数値で被験者に答えてもらった。被験者は20代30代の健常な男性4人で、被験者への各刺激値を1回ずつ、ランダムに提示した。

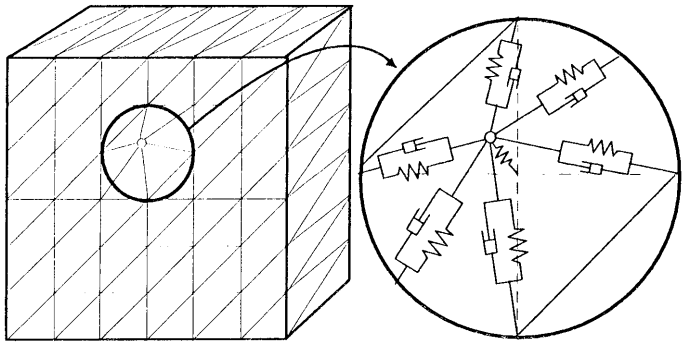


図8 実験で用いた立方体バネモデル
Fig. 8 spring model for experiments

5.2 実験結果

4人の実験結果をまとめたものを図9に示す。横軸は被験者に提示したモデルの基準値を100としたときの硬さ指標、縦軸は被験者が基準モデルと比べて知覚し評価した感覚量の平均値と標準偏差である。

被験者が答えた感覚量にばらつきが見られるものの、3種類の触覚ディスプレイではどれも似た傾向を示し、触覚ディスプレイによって知覚する感覚量の違いは、それほど見受けられない。

また一般に、提示力と感覚量の関係は、図10のようにべき乗則の関係にあると言われている[13]。使用する触覚ディスプレイにより提示力が異なるが、用いた触覚ディスプレイの最大提示力を基準値の100とし、そのと

触覚ディスプレイ	弾性係数(gf/cm)	減衰係数(gf·s/cm)
PHANToM	1.1×10^2	2.2×10^2
HapticMaster	3.5×10^2	7.0×10^2
HapticScreen	2.7×10^3	5.4×10^3

表2 各触覚ディスプレイに対する基準モデルの弾性係数、減衰係数
Table 2 physical constants of referencial model for each haptic display

変動要因	P-値
硬さ指標	3.06E-06
触覚デバイス	0.829044

表3 分散分析表
Table 3 table of dispersion analysis

NII-Electronic Library Service

The Virtual Reality Society of Japan

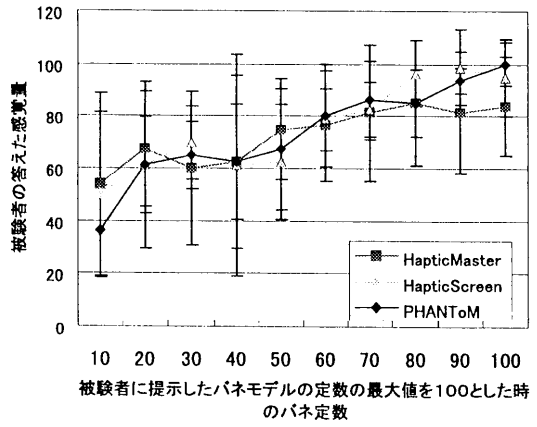


図9 実験結果
Fig. 9 result of the experiment

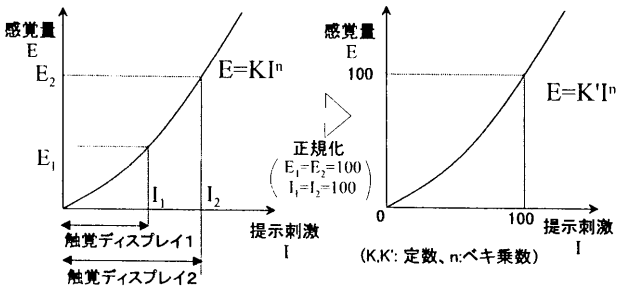


図10 提示刺激と感覚量の関係
Fig. 10 relation between sensation and stimulus

きの感覚量を100とする正規化を行なっているため、図10のようにどのディスプレイを用いても、硬さに対する被験者の感覚量が、ほぼ同じ値をとるのは妥当な結果と考えられ、また各触覚ディスプレイは、提示方法、機械的特性、提示物理量等が異なるが、硬さの変化という触覚表現に対しては、同等の感覚が提示できた。

表3に標本数40による分散分析表を示す。硬さごとによるP-値は小さく被験者は硬さを別々に感じていると言え、触覚ディスプレイごとによるP-値は大きくデバイスによる感覚量の差はあまりないと言える。このことから、同じ触覚情報をHIPにより異なる触覚ディスプレイに提示しても、同等の感覚変化が人間に知覚されることが示され、HIPの開発意義が検証された。

また、図9で、折れ線が原点を通らないのは、基準値として上限100の硬さのモデルのみをユーザに提示し、下限を指標として与えなかったことが原因の1つに考えられる。

6. 考　　察

HIPは非常に多様な法則エンジンを持っているため、図7に示した以外にも、剛体運動やテクスチャを表現することが、非常に簡単なプログラムで可能となる。プログラムを組むことよりも、触覚仮想世界を構築する上でむしろ大変なのは、どのような触覚モデルを作成するかである。しかし、HIPでは表面接触モデル作成に汎用的に使われるDXFファイル、VRMLファイルを読み込むことができ、またテクスチャモデルにおいてもTARGA形式ファイルを読み込むことができる。近年インターネットにより、車や人体などの複雑な形状データを表現するVRMLファイルが、誰でも手軽に手に入れることが可能になり、それらをダウンロードして、すぐに触覚表現できるのは非常に有効である。またモデルに変形や運動を付け加えるときにも、モデルに弾性係数や質量、重心の位置等の情報をモデルファイルに記述することで、簡単に表現することができるようになる。

HIPは様々なアプリケーションへ応用可能であると考える。例えば図11に示すような車あるいは工業製品の分解組立シミュレーションへの応用が期待できる。また今まで難しかった異種デバイス間での協調作業の実現が可能となる。例えば造形作業において、それぞれの触覚ディスプレイの特徴を生かし、面型で大まかな形状を作り、点型で細かな修正を加えていくなどということができるようになる[14]。

5.では、デバイス間による感覚の同等性を確認した。また同じプログラムを異なるデバイスで動作させることは、逆にHIPを触覚ディスプレイのベンチマークテストの役割として使うことも可能である。

7. ま と め

本論文では、仮想世界において触覚を表現するための触覚用基本ソフトウェアHIPの基本構成とその実装方法について述べた。ユーザが触覚ディスプレイの形態に依らずにHIPを使用できるように、触覚ソフトウェアを体系的に構造化した。

HIPは、(1)個々の触覚ディスプレイの制御を行なう「デバイスドライバ」、(2)デバイスと仮想空間との間の

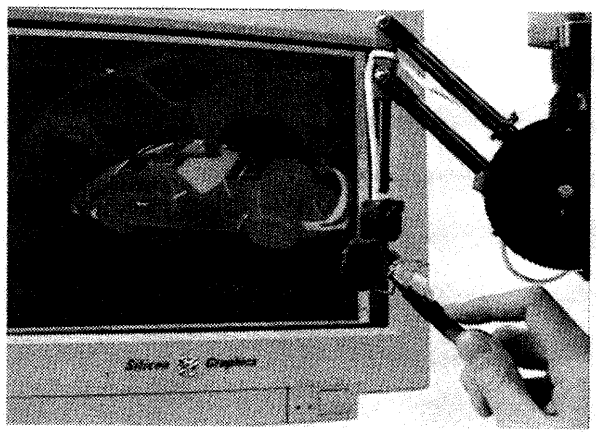


図11 HIPを用いたアプリケーション例
Fig. 11 sample application by HIP

NII-Electronic Library Service

The Virtual Reality Society of Japan

橋渡しの役割を行なう「触覚レンダラ」、(3)仮想物体の運動、変形のシミュレーション計算を行う「モデル演算」、(4)触覚モデルを扱う「触覚モデルデータ」の4つにモジュール化されている。またHIPの実装方法を述べ、非常に簡易なプログラムにより、様々な触覚仮想世界が構築できることを示した。

更に、同一モデルを異なる触覚ディスプレイに提示し、その硬さを被験者に認識してもらう実験では、被験者が答えた感覚量に触覚ディスプレイによる有意な変化は見られず、どの触覚ディスプレイを用いても、類似の感覚が得られることが分かった。この実験により、HIPで様々な触覚ディスプレイを使用しても、ある程度同等の仮想世界を構築できることが示された。

今後の課題としては、モデル演算機能の充実、触覚モデルを容易に作成することのできる触覚エディタの開発、HIPを用いたアプリケーションの開発等が挙げられる。

8. 謝　　辞

本研究は、平成8-9年度通産省情報処理振興事業協会「創造的ソフトウェア育成事業」の支援を受けて行なわれた。また研究開発を進めるにあたり、日商エレクトロニクス株式会社に協力して頂いた。ここに関係諸氏に感謝の意を表したいと思う。

参考文献

[1] B.Shimoga：A Survey of Perceptual Feedback Issues in Dexterous Telemanipulation: Part1. Finger Force Feedback、IEEE Virtual Reality Annual International Symposium (1993)

[2] GHOST General Haptics Open Software Toolkit、Brochure of SensAble Technologies Inc.(1996)

[3] C.Ruspini、K.Kolarov、O.Khatib:The Haptic Display of Complex Graphical Environments、Computer Graphics Proceedings p345～p352 (1997)

[4] Mark, Randolph:Adding Force feedback to Graphics Systems: Issues and Solutions、Computer Graphics Proceedings p447～p452 (1996)

[5] 矢野博明、岩田洋夫：力覚帰還型仮想環境構築ソフトウェア、日本バーチャルリアリティ学会論文集 Vol.2 No.2 p1～p9 (1997)

[6] 廣瀬通孝、岩田洋夫、池井寧、小木哲朗、広田光一、矢野博明、筧直之：デバイスに依存しない触覚用共通ソフトウェア(HIP)の開発、日本バーチャルリアリティ学会第2回大会論文集 p202～p205 (1997)

[7] Thomas H.Massie：Virtual Touch Through Point Interaction、International Conference on Artificial Reality and Tele-existence p19～p38 (1996)

[8] 浅野武夫、矢野博明、岩田洋夫：フォースディスプレイを用いた仮想環境における手術シミュレーションの要素技術開発、日本バーチャルリアリティ学会第1回大会論文集 p95～p98 (1996)

[9] 広田光一、廣瀬通孝：面提示型触覚ディスプレイのための曲面表現デバイス、第10回 ヒューマンインターフェースシンポジウム p193～p196 (1994)

[10] 岩田洋夫、市ヶ谷敦郎：ハプティックスクリーン、日本バーチャルリアリティ学会第1回大会論文集 p7～p10 (1996)

[11] 若松和史、池井寧、福田収一：触覚テクスチャの表示におけるデータ依存特性、日本バーチャルリアリティ学会第1回大会論文集 p185～p186 (1996)

[12] 横小路泰義、ラルフホリス、金出武雄：仮想環境への視覚／力覚インタフェース：WYSIWYFディスプレイ、日本バーチャルリアリティ学会論文集Vol2、No4 p17～p24 (1997)

[13] Stevens,S.S.： Pychophysics. New York: Wiley(1975)

[14] 廣瀬通孝、小木哲朗、矢野博明、筧直之：異種デバイス間の触覚協調作業の実現、第48回 ヒューマンインタフェース研究会資料 p121～p126 (1998)

(1998年4月3日受付)

NII-Electronic Library Service