OpenHRP を使った人間型ロボットのシミュレータ開発*

長井　達一郎**
小神野　東賢***
河村　雅人****
荒牧　重登**

Development of Humanoid Robot Simulator using OpenHRP

Tatsuichiro NAGAI, Touken OKANO,
Masato KAWAMURA and Shigeto ARAMAKI

It has been recently requested to make a humanoid robot with high performance. It is difficult to prepare these robots because those are expensive. When the experiment of robot task fails, the robot might damage oneself. Then, OpenHRP was developed as a joint environment for robotic software. This environment also provides the simulation function of the robot. We developed the environment of simulation for our humanoid robot (MES-HR) by using OpenHRP.

Key Words: OpenHRP, HRP, Humanoid Robot, CORBA, Simulator, Jython

1.　はじめに

今日のロボット研究開発においては，例えば，アクチュエータや機構制御だけでなく，画像認識，音声認識，人工知能などの専門知識も必要となっている．そして，それぞれの異なる専門知識を有する開発者による大規模な開発が不可欠となってきた．その大規模開発にロボットのためのソフトウェア開発があり，ソフトウェアの共通基盤が必要となっている．また，開発途中の制御系や作業の実験を行った場合，ロボットが予期せぬ動作を行い，ロボット自身あるいは，その動作環境が破損する危険性がある．そのため，そのような実験を行う場合は，まずシミュレータで動作試験を行い，安全性と有効性を確認後に，実機にて実験するのが望ましい．しかし，従来のシミュレータは実機とは違うOSや言語である場合が多く，シミュレータ用のコードを実機用に移植する作業が必要となる．そのため，移植に時間がかかったり，移植の過程でバグが混入するなどの問題が起きていた．そこで，シミュレータおよびコントローラの機能を持つ，ロボットの研究開発のための共通基盤となるソフトウェアプラットフォーム OpenHRP (Open architecture Humanoid Robotics Platform) [1] [2] が開発された．これは，経済産業省のプロジェクト「人間協調・共存型ロボットシステム研究開発」(HRP：Humanoid Robotics Project, 1998～2002) [3] の中で開発された成果である．OpenHRP のシミュレータ部は非商用に限り，産総研(産業技術総合研究所)から無償で配布されている．

当研究室のロボットも大規模開発を想定し，目，首，腕，腰などの各部位を独立開発，制御できるようになっている．しかし，シミュレータの開発が困難で，今まで

* 平成20年１月10日受付
** 電子情報工学科
*** ゼネラルロボティックス㈱
**** 奈良先端科学技術大学院大学

は最初から実機による実験を行ってきた．そのため，破損することもしばしばあった．また，１台しか実機がないため，複数の実験を同時に行うことができなかった．そこで，複数のコンピュータによるロボット制御が可能な開発環境，3D の動作シミュレータ，シミュレート用と実機用で動作プログラムの変更がない等の条件から，OpenHRP を当研究室のロボットに適用した開発環境を開発した．

2. FUKHR/MES-HR の構成

当研究室にある研究用ヒューマノイドロボットは，三井造船株式会社製のロボット“MES-HR”である（図１参照）．身長は約150cm，重量は約150kg である．

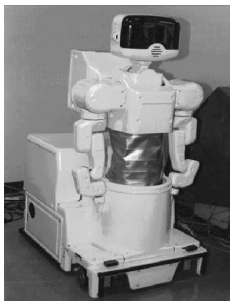


図1 MES-HR

MES-HR は，人間の目に当たるカメラと耳に当たるマイクと口に当たるスピーカーを搭載した頭部，前後左右の傾きに加えて回転する機構を備えた首，腰，それぞれ７自由度を有する２本の腕を搭載しており，人間の上半身を模した造りとなっている．下半身は台車となっており，移動は車輪を用いて行う．

2.1. ロボットの関節自由度

MES-HR の関節自由度を図２に示す．図に示すように目，首，腕，腰に関して人間とほぼ同等な動きを行うことが可能である．ただし，指は３本で関節はないが物を把持することができる．

2.2. 制御方法

目，首，腕，腰は，それぞれを独立して制御できるよう，各部位毎に制御用コンピュータ（CPU ボード）が搭載されている．制御を行う際には，各部位用の CPU ボードにプログラムを外部コンピュータからダウンロードして実行する．各部位と，制御するための CPU ボードと，外部コンピュータの接続関係を図３に示す．目，首，両腕，腰，車輪，そして人間の皮膚の役割をするタッチセンサ用など各部分の制御用コンピュータが独立して存在する．これらのコンピュータ上では WindRiver 社のリアルタイムオペレーティングシステム VxWorks[4] が

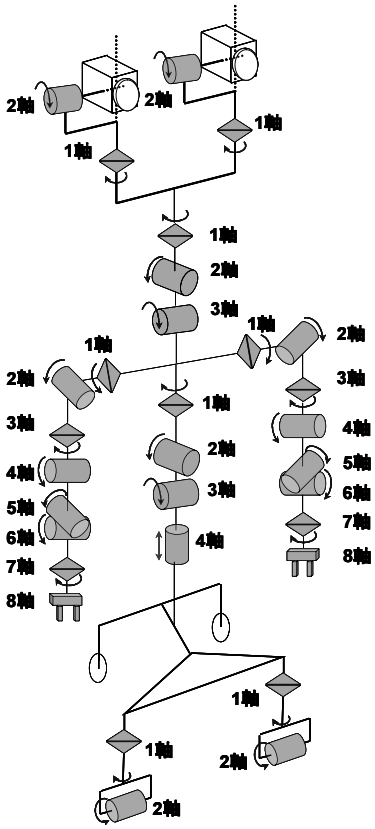


図2 ロボットの自由度

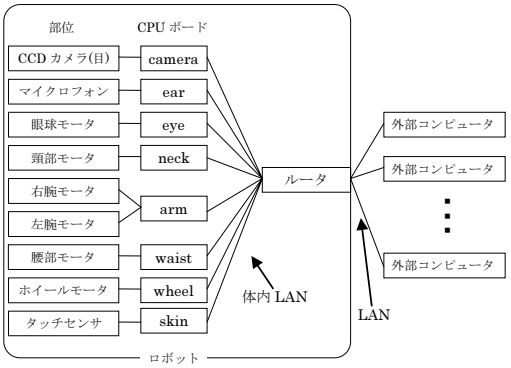


図3 ロボットの制御ネットワーク構成

実行されており，マルチタスクの機能をもっている．また，これらのコンピュータが協調できるように LAN に接続されている．さらに，ルータを介して外部との LAN 環境に接続可能であり，これにより複雑な処理やロボットの動作プログラムの開発支援を，外部のコンピュー

タ上で行うことを可能にしている．この外部コンピュータは LAN 接続できるものであれば良く OS に依存しない．

2.3. 既存の開発環境

今までは，外部コンピュータで各部位の動作プログラムを作成し，VxWorks の開発環境 Tornado で CPU ボード用にコンパイルしたものを，CPU ボードにダウンロードし実行するものであった．そのため，動作プログラムにミスがあった場合は，実機が壊れることもしばしばあった．

3. OpenHRP

人間型ロボットのソフトウェア開発プラットフォーム OpenHRP は，モジュール性，資産継承，シミュレーションと実機との互換性，オープンアーキテクチャという設計方針に基づいて構成されている．

3.1. CORBA

OpenHRP は CORBA（Common Object Request Broker Architecture）[5] に基づく分散オブジェクトシステムとして実装されている．この CORBA の構成要素として制定されている ORB（Object Request Broker）が，異なるマシン上に分散して存在するオブジェクト（モジュール）間でデータや処理要求などのメッセージをやりとりする際に用いられる．

3.2. モジュール

OpenHRP には次のようなモジュール（サーバまたはプラグインと呼ばれることもある）が準備されており，必要に応じて組み込み，取り外しが可能である．

モデルパーサ（Model Loader）：ロボットや環境の 3 次元モデルが記述された VRML ファイルの読み込み，形状データ及び機構・動力学パラメータ等の情報を他のサーバに提供する．

干渉チェッカ（Collision Detector）：モデルパーサから形状データを受け取り，干渉の位置，法線，干渉深さを返す．

動力学計算（Dynamics）：コントローラから各関節への入力トルクを受け取りシミュレーション世界の順動力学計算を行い，ロボットに搭載されるセンサから出力を返す．

コントローラ（Controller）：ロボットのコントローラであり，動力学計算サーバからセンサ出力を受け取って制御計算を行い，関節へのトルク指令を出力する．通常 OpenHRP のユーザによって開発される．

ビューシミュレータ（Vision Sensor）：視点の位置／姿勢を受け取って，ロボットのカメラから見た視野画像を生成する．

パターンジェネレータ（Pattern Generator）：一歩ごとの着地位置データから動的に安定な歩行動作を生成し，その関節角軌道と ZMP（Zero Moment Point）の軌道を出力する．

モーションプランナ（Motion Planner）：環境とロボットとが干渉せず，かつ動力学的に安定な動作を計算する．

入力デバイス（Input Device）：ジョイスティックのような入力デバイスの状態を提供する．

このように，OpenHRP では 2 足歩行の開発が行えるように，歩行動作を生成するモジュールや，重力，重量，モーメントを考慮した動力学計算を行うシミュレータ環境が用意されている．

3.3. モデル

OpenHRP ではロボットおよび環境のモデルは VRML97（Virtual Reality Modeling Language，VRML 2.0）をベースに h-anim WG [6] で策定されたヒューマノイドの記述フォーマットを一部拡張したものを用いている．これにより 1 つのファイルに形状，機構，動力学パラメータといったシミュレーションに必要な情報を全て埋め込んでいる．そのため，ロボットなどを記述する場合，各関節の 3 次元形状だけでなく，質量や重心の位置なども必要となる．

3.4. リアルタイム OS

OpenHRP は，シミュレーションから実機での実験への移行をスムーズに行うために，コントローラの一元化を行っている．その実現のために ART-Linux [7] を採用している．ART-Linux は Linux のリアルタイム拡張の一種である．これは，リアルタイムタスクが通常のアプリケーションが実行されるユーザ空間で実行されるため，既存のソフトウェアとの親和性が極めて高い．リアルタイム OS を採用しているため，動作シミュレーションを行い，動作プログラムの検証後，実機による実験を直接行うことができる．また，OpenHRP のシミュレーションで得られる各モジュールによる動作命令を保存し（動作を作る），実機でその動作命令を再生することも可能である．

4. OpenHRP for FUKHR/MES-HR

OpenHRP は人間型ロボットのソフトウェア開発プラットフォームであり，VRML を用いてロボットを記述し，ロボット用のコントローラを作成すれば，そのロボット用のシミュレーション環境が整う．しかし，多自由度の関節を持つロボットのモデルを記述するにはかなりの時間を必要とする．また，コントローラの実装では，ロボットの I/O に直接アクセスするためにデバイスドライバなどが必要となる．そこで，今回は人間型ロボット HRP-2 [8] 専用の OpenHRP [9] を開発したゼネラルロボティ

クス(株)が開発したモデルとコントローラならびにユーザインターフェースを採用した．HRP-2には HRP-2W (ホイールタイプ)とよばれるロボットがあり，これは足の部分が車輪になっており，当研究室の MES-HR に近い形であるためである．

4.1. システム構成

今回開発したシステムの構成を図４に示す．

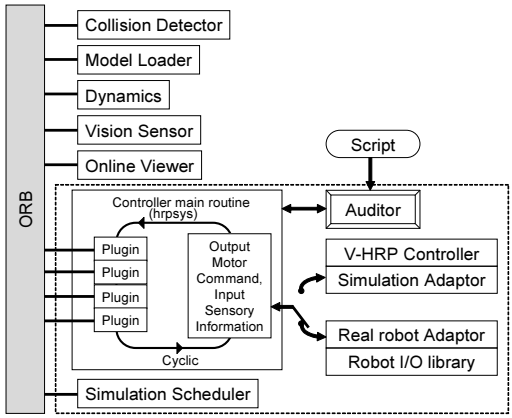


図4 システム構成

図４は産総研から無償で提供されている OpenHRP を構成する CORBA オブジェクト群を含めたシステム全体の構成を示している．破線の部分がゼネラルロボティクスによって開発されたものである．コントローラ内部はプラグインと呼ばれる複数の機能モジュールとそれらを制御するプラグインマネージャと呼ぶ部分に分割して実装されている．ここのプラグインは OS が提供するダイナミックローディング機能を用いてコントローラ本体部分に動的にリンクされるため，必要に応じて組み込み，取り外しが可能である．

4.2. OpenHRP と VxWorks の通信

図４の Real robot Adaptor と図３の MES-HR の各部位の CPU ボードの VxWorks は，LAN で接続されており，プロトコルは速度を維持するために UDP 通信である．VxWorks 側は受信した動作命令の実行とエンコーダ値の返信を行っている．OpenHRP を実行しているコンピュータは MES-HR の CPU ボードに比べ処理速度が十分に速いため，OpenHRP から高速に大量の命令を送信すると，VxWorks 側の処理能力を超えてしまい動作が不安定になることがある．そこで，今回は OpenHRP から VxWorks に動作命令(関節角データの指令値)を送信し、VxWorks から OpenHRP に関節のエンコータ情報を送信するループを10ms ごとに行っている．

4.3. インターフェース Auditor

OpenHRP のインターフェース Auditor[9]を図５に示す．

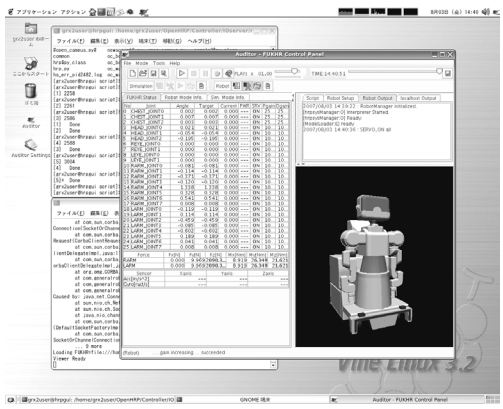


図5 Auditor

実行するスクリプト，各軸の角度，ゲイン値，シミュレーションと実機の軸のずれ，ロボットの３次元画像を表示することができ，シミュレーションにてその挙動の確認が容易な作りになっている．

5. 動作実験

OpenHRP は，ロボットの動作プログラムとして，スクリプト言語とプログラム言語の両方が用意されている．今回は，操作が簡単なスクリプトの例を示す．

5.1. スクリプト

OpenHRP を用いて簡単にロボットを動かす方法にスクリプトの実行がある．OpenHRP は Jython[10][11]のスクリプトを処理できる．ここでは，Jython を用いたロボットの動作命令を示す．

ユーザは Auditor を通して，OpenHRP にロボット，ここでは MES-HR のプラグインをインポートし，そのコントローラに対してロボットの各関節角度を指定する．OpenHRP 上に定義されているロボットは，各関節を指定された角度に移動させる．

以下に，関節角度を指定するスクリプトの書式を示す．関節は Joint0～Joint25の26個ある．それらを全て指定することで，同時に関節を動かすことが可能である．

send seq：joint-angles　Joint0の角度　Joint1の角度
　　…　Joint25の角度　動作スピード

角度は度(°)で指定する．全ての角度を０にすると各関節は原点に移動する．以下に例を示す．

```
send seq：joint-angles 0 0 0 0 0 0 90 0 0 0 0 0 0 0 0 0
                       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7
```

これは，スピード 7 で Joint5（首）を90° の位置に回転させる命令である．原点位置とプラスの回転方向は予め決まっている．

スクリプトは Auditor 上で直接入力編集することが可能である．また，予めスクリプトのリストをファイルに保存しておき，読み込むことも可能である．今回実験に使用したスクリプトを以下に示す．これらは，予めファイルに保存しておき，実行時に読み込むものである．２種類あり，１つはロボットを OpenHRP にインポートし動作のスクリプトを呼び出すファイル，もう１つは動作の内容を記述したスクリプトのファイルである．

```
import hrp
ms = hrp.findPluginManager("motionsys")
ms.sendMsg(":interpret load script/common")
ms.sendMsg(":interpret load script/test")
ms.load("seqplay")
seq = hrp.findPlugin("seq")
seq.sendMsg(":wait-interpolation")
```

(a) OpenHRP にロボットをインポートするスクリプト

```
send seq :joint-angles 0 0 0 0 0 0 0 0 0 0 0 15 -10 0
                       -90 0 0 0 0 15 10 0 -90 0 0 0 0 7
send seq :wait-interpolation
send seq :joint-angles 15 15 15 10 10 10 0 0 0 0 -60 -40
     -30 -60 -20 -20 -30 10 -60 40 30 -60 20 -20 30 10 6
send seq :wait-interpolation
send seq :joint-angles 0 0 0 0 0 0 0 0 0 0 0 15 -10 0
                       -30 0 0 0 0 15 10 0 -30 0 0 0 0 4
```

(b) ロボットの動作スクリプト

図6 Jython による動作プログラム

図６(b)の：wait-interpolation は，１行前の動作が完了するまでスクリプトの実行を待たせる命令である．

5.2. シミュレーション

図４の Simulation Adaptor を選択することで，実機を動かさずに動作の検証を行うことができる．図６のスクリプトをシミュレーションした画面を図７に示す．

5.3. 実機による実験

図６のスクリプトを実機で実行した結果を図８に示す．(a)に示すように，実機が動作する場合は，Auditor 上の3D 画面のロボットは赤く塗られる．

実機の動作は，エンコーダ値の取得により Auditor 上の3D 画面でも確認できる．エンコーダ値はリアルタイムで取得しており，また今回は OpenHRP とロボットはハブなどを経由せず直接 LAN ケーブルで接続しているため，ほとんど遅延はなかった．

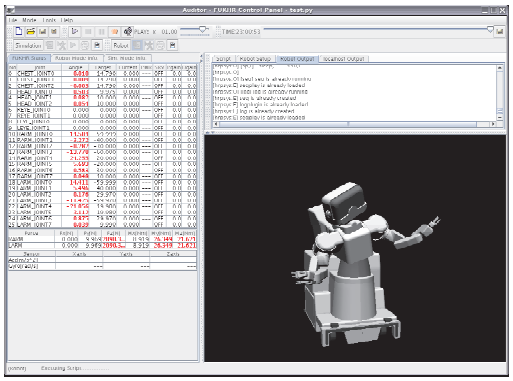
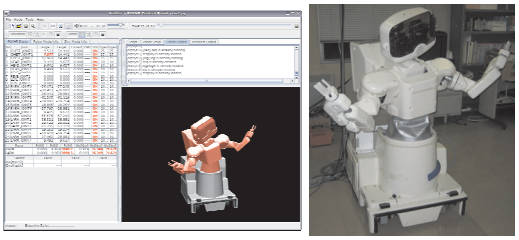


図7 シミュレーション



(a) 実機を動作中の画面　　(b) 実機

図8 実機での実行

6. まとめ

研究室の人間型ロボット用のシミュレータを OpenHRP を用いて開発した．これにより，簡単に動作命令をプログラムし，シミュレーションすることが可能になった．さらに，動作の安全性を確認後，コードの変更無しに実機を動かすことが可能になった．また，この人間型ロボット用の OpenHRP の環境を複数の計算機に準備することで，シミュレータとして複数の学生が同時に使用できるようになった．今回の開発により，コード開発時間の短縮，動作シミュレーションによる安全性の向上，複数のプロジェクトの同時開発が望める．

しかし，ロボット側の通信速度と CPU ボードの処理速度が遅いため，速い動作を行うとシステムが落ちることがあることが確認された．これは，OpenHRP の Real robot Adaptor と VxWorks 側のプログラムを調整することで，よりシステムを安定化させることが可能と考えている．今後は，OpenHRP に実装されている様々なモジュールの利用，カメラの映像とのリンク等による開発環境のますますの充実が課題である．

参 考 文 献

[1] 金広文男，ヒューマノイドソフトウェアプラットフォーム OpenHRP とその応用事例，日本ロボット学会誌，vol.21，no.6，pp.609-614，2003.
[2] 金広，藤原，梶田，横井，金子，比留川，中村，山根，ヒューマノイドロボットソフトウェアプラットフォーム OpenHRP，日本ロボット学会誌，vol.21，no.7，pp.785-793，2003.
[3] 井上，比留川，人間協調・共存型ロボットシステム研究開発プロジェクト，日本ロボット学会誌，vol.19，no.1，pp.2-7，2001.
[4] ウインドリバー株式会社，http://www.windriver.com/japan/
[5] Object Management Group，http://www.omg.org
[6] HUMANOID ANIMATION WORKING GROUP，http://www.h-anim.org/
[7] 石綿陽一，松井俊浩，高度な実時間処理機能を持つ Linux の開発，第16回日本ロボット学会学術講演会予稿集，pp.355-356，1998.
[8] 金子健二，金広文男，梶田秀司，横山和彦，赤地一彦，川崎俊和，太田成彦，五十棲隆勝，HRP-2プロトタイプの開発，日本ロボット学会創立20周年記念学術講演会予稿集，1D32，2002.
[9] 小神野東賢，OpenHRP 制御ソフトウェアの事業化，日本ロボット学会誌，vol.22，no.1，pp.18-19，2004.
[10] Samuele Pedroni, Noel Rappin, Jython Essentials, Oreilly & Associates Inc, 2002.
[11] マーク　ルッツ，デイビッド　アスカー，Mark Lutz, David Ascher，初めての Python，オライリージャパン，2004.