動力学運動シミュレーション機能を埋め込んだロボットプログラミングシステム

○小倉 崇　岡田 慧　稲葉 雅幸 (東京大学)

Dynamics Simulator Embedded Robot Programing System

*Takashi OGURA, Kei OKADA, Masayuki INABA
(TheUniversity of Tokyo)

Abstract— In this paper, in order to make a robot's brain have a simulator, an efficient robot programming environment is proposed, and a simulation result is shown. It is difficult that a robot predicts the external world or it considers performing the action planning in consideration of the dynamic characteristic of the external world, since it was difficult to perform synchronization of a simulation, and dynamic changes of a structure object. since a robot's brain and a simulator are separated, therefore for a robot to determine action using a simulator is diffucult. The development environment embedding the dynamics simulator is proposed, and an example implementation by expanding the functions of EusLisp.

Key Words: simulator, programing system, development environments

1. はじめに

これらのシミュレータは実ロボットの代用として用いられることが多く，ロボットのコントローラを接続先を実機とシミュレータとを切り替えることで，同じプログラムで実験が可能なシステムになっていることがしばしばある．本稿では，ロボットが外界の予測を行ったり，外界の動的な特性を考慮した行動プランニングをシミュレータを用いて行うことを考え，そのためのシミュレータを上記の実ロボットの代用としてのシミュレータと区別し，ブレインシミュレータと呼ぶことにする．ロボットのブレインがシミュレータを持つために効率的なロボットプログラミング環境を提案し，実装例を示す．

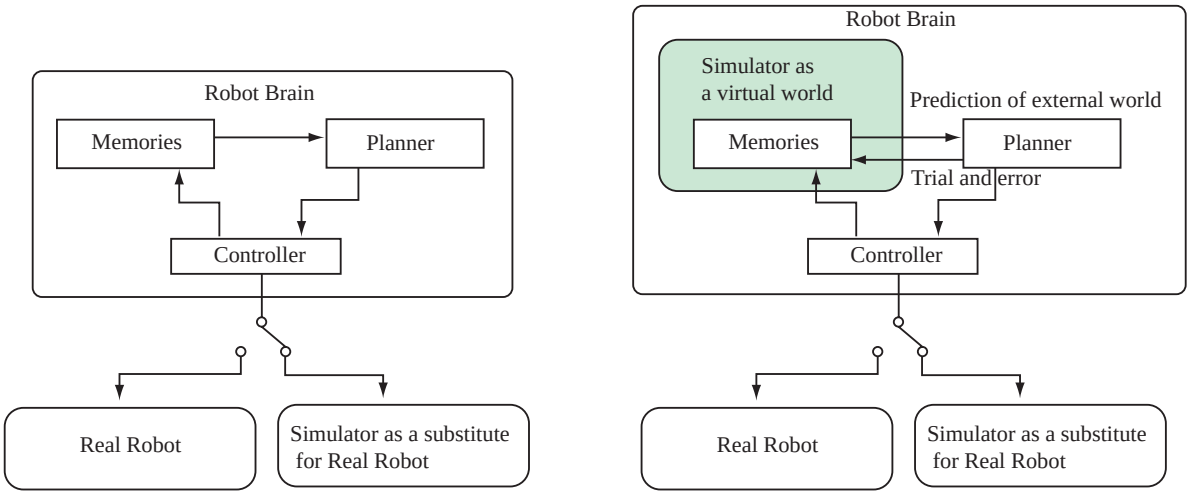
ブレインシミュレータを実現するためには環境内の物体を動的に生成できるような幾何モデリング環境に対する動力学シミュレーション機能が必要になる．従来の実ロボットの代用として利用されるシミュレータでは一般にシーンファイルと呼ばれるような，環境内の物体が静的に定義されたファイルを読み込みシミュレーションを行っている．また，同期シミュレータとブレインとの同期がとりにくいという問題がある．既存のロボット開発環境では Fig.1(a) に示すような，ロボットのブレインとなるプログラムとシミュレータは切り離された環境にある．このような状態ではシミュレーションの同期や構成物体の動的な変化をロボットが行うのが困難であるという問題があり，そのため，これまでロボットがシミュレータを用いて行動を決定することが困難であり，ブレインシミュレータが実現しない大きな原因であった．これを解決するために，Fig.1(b) のように動力学シミュレータを埋め込んだ開発環境を提案し，その構成法を示す．そして具体例として EusLisp を拡張する形での実装法を示す．また，いくつかの簡単なコード例を挙げ，ロボットのブレインが操作可能なレベルでシミュレーションが記述が行われていることを示す．

2. プログラミング環境におけるヒューマノイドロボットのシミュレータ

ヒューマノイドのような多自由度系に対応した動力学シミュレータとして，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 (HRP) では，シミュレーションによってヒューマノイドの実時間制御ソフトウェアを開発するための仮想プラットホーム OpenHRP[1] を開発し，公開されている．ただし，シミュレーションは速度より正確性を重視しており，実時間に対して数倍から数十倍程度の計算時間がかかる．また，金広らはゲーム用動力学エンジンを使用し，実時間に近い速度でシミュレーションを行うシミュレーションソフトウェア [2] を開発している．これらはいずれも開発したソフトウェアを全く変更なしに実ロボットに適応できる点を特徴としており，シミュレータはソフトウェア開発における，実ロボットの代用，もしくは解析を目的として用いられている．

3. ブレインが用いる実世界シミュレーション

シミュレータを単なる実ロボットの代用として用いるのではなく，ロボットのブレインとして用いることが行われつつある．GA などの学習をシミュレータ上で行い，その結果を実機を用いて行うことが行われている [3]．また，環境の変化の予測をシミュレータを用いて行う試み [4] が行われている．このような場合のシミュレータの用いられ方は，これまでに述べたような実ロボットの代用というよりはむしろ，ロボットのブレインが仮想世界で行動計画を行うために，記憶を用いていると考えたほうがふさわしい．このような場合は Fig.1(a) のような構成のソフトウェア環境よりも Fig.1(b) のように，ブレインとシミュレータが一体となった環境が望ましい．このようなソフトウェア環境を構成するためにはソフトウェア開発言語自身に動力



(a) Normal robot software system

(b) Proposed system that a robot brain has a simulator

Fig.1 difference between robot software systems. Usually simulator is used for substitution of real robot and contoller can switch the connection to the target. Proposed system has a simulator in a robot brain. The simulator world is structed as its memories. The simulator world must be written by easy codes because it must be contructed by robot sensors and robot brain not by human.

学計算ライブラリを有していることが大きなアドバンテージとなる．以下ではそのようなソフトウェア環境の構築法を具体例を通して解説する．

4. EusLisp を用いたシミュレーション環境の実装

ここでは動力学運動シミュレーション機能を埋め込んだロボットプログラミング環境の実装のひとつの例として，EusLisp の拡張による実装の例を挙げて解説する．以下に述べる実装法は Java3D[5] や VPython[6] といった 3 次元の幾何モデルを扱うことができるが，物理モデルを持たない処理系の拡張手法としては共通した構成法である．

まず，EusLisp の基本機能について説明する．EusLisp は幾何モデリング機能をもったマルチスレッドなオブジェクト指向の Lisp 言語であり [7]，３次元表示機能などももっている (Fig.3(a))．C/C++言語に比べ，キーワード引数，プログラムやデータの S 式による表記，lambda 式，ハッシュ表など優れた機能を持つ．幾何モデリング機能に関しては，EusLisp が持つ物体形状プリミティブとして直方体，円柱，回転体などがあげられる．また，これの和・差・積などの集合演算によって複雑な形状を合成することができる．これまでに主にロボットのモデリングや作業計画，ロボットの行動計画に応用されてきているが，動力学シミュレータを持つことで，すでに述べたように記憶構造の中に動力学を組み込んだモデルを作成，もしくは，動力学モデリングのプロトタイピングとして用いることができるようになる．EusLisp 単体でも接触状態などを取り扱うことができるが，C++で書かれた動力学計算ライブラリ (ODE[8]) を他言語インタフェースを利用して新たに組み込むことで，この物体形状に動力学的な特性を与える (Fig.3(b))．ODE はインタラクティブで実時間を志向したシミュレーション用に作られた動力学ライブラリであり，物理計算を Lisp で記述する場合に比べ計算速度が早い利点がある．また，シミュレーションループはマルチスレッド機能により別スレッドで行うことができる．

この実装のひとつの特徴は，スクリプト言語であるために特別なインタフェースを容易することなしにオブジェクトをインタラクティブに生成，操作，廃棄が可能である点である．他に ODE を利用した容易に環境を記述することが可能な実装 [9] が存在するが，インタラクティブ性に欠けているか，もしくはロボットプログラミング環境としては拡張性に乏しい．

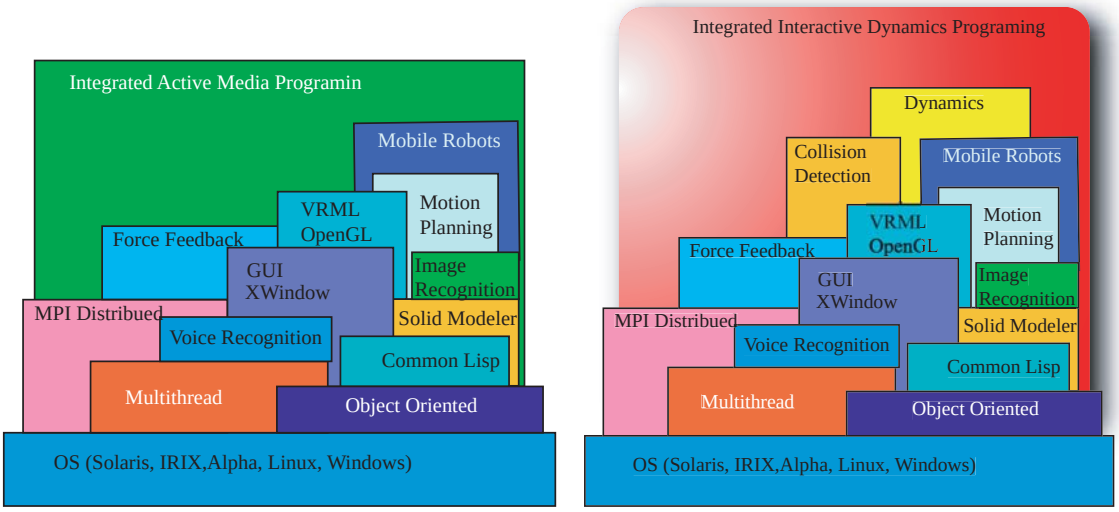
EusLisp の幾何モデルに ODE の幾何モデルと物理モデルを結びつけるための流れを以下に示す．

1. プリミティブ形状作成関数 (make-cube など) を一時避難する．
2. (make-cube) をオーバーライドする
3. オーバーライドした関数の中で避難したプリミティブ形状作成関数を呼び，形状オブジェクトを作成
4. 動力学ライブラリの幾何モデルと物理モデルを作成
5. これらをまとめて取り扱う
6. 各形状オブジェクトに対する操作関数をオーバーライドし，物理モデルに反映するようにする

–は Lisp のコードでは Fig.4. のように表現される．コードは一部簡略化してある．

5. サンプルコードと実行結果

ロボットがボールと箱を視覚を用いて観測したと仮定し，近い将来の予測を行うことを考える．ボール，箱



(a) EusLisp System　　(b) EusLisp with ODE System

Fig.3 Expansion of EusLisp

```
1  (make-cube (x y z)
2    (setq eus-obj (make-cube-org x y z))
3    (setq ode-obj (dCreateBox *space* x y z))
4    (setq ode-body (dBodyCreate *world*))
5    (dGeomSetBody ode-obj ode-body)
6    (putprop eus-obj ode-obj :ode-obj)
7    eus-obj)
```

Fig.2 make-box override source code

の質量を既知とし，ボールの位置，初期速度をビジョンによって観測することができれば，リンクを持たない仮想世界であれば構築可能である．

Fig.6. は Fig.6. に示すソースコードを示す．(d-init) 関数によりシミュレーション世界を初期化している．シミュレーションループはスレッドで自動で行うことも可能であるが (d-tick) 関数によりシミュレーションループを一回ずつまわすことが可能になっている．(make-cube) などにより動的に世界が構築されている．これにより生成された仮想世界とシミュレーション結果である．200[msec] 後までのシミュレーションを行った．箱の倒壊の様子が予測されている．

このシミュレーションを CPU: Pentium Xeon 3.20GHz x 2, OS: Linux という構成の PC 上で行い，動力学計算を行うのに 80[msec] を要した．実時間の 25%の時間で計算が可能であり，実世界に対応するために十分早い時間で計算を行うことができている．計算は 10[msec] 刻みで行った．

次に 18 自由度のヒューマノイドが階段の上に立っている状況を想定する．自らの体に関する情報を既知とし，仮想世界を構築したとする．サーボゲインを十分に姿勢を保てない程度に弱くしているため，階段からころげ落ちる様子が Fig.6. に見て取れる．同上の PC 上で計算を行い，1500[msec] のシミュレーションを行うのに 880[msec] かっているが，危険を察知することができている．計算ループはブレインから動的に操作することができるため，判断にもとづき，必要に応じてさらに細かい計算を行うことが可能である．同様のことを既存のシステムで行うためには，コントローラ，シミュレータを別のプロセスで立ち上げ，それらの通信を行う必要がある．しかもそのシミュレータは柔軟に構成物体を生成，移動，消去ができる構成になっていなくてはならない．それに比較し，このシステムではブレインを単一プロセスで，しかも 10 数行程度の簡単な記述によって表現できており，通常のプログラミング環境に比べ有利であるといえる．

6. おわりに

本稿ではロボットのブレインが実世界シミュレーションを用いる際に有効なシステム構成をしめし，その実現例として EusLisp を動力学ライブラリ ODE を組み込むことで拡張したシステムについて述べた．また，このシステムを用いてシンプルなコードでシミュレーションを記述でき，シミュレーションを実世界より十分早く行うことが可能であることを確認した．ただし，実世界の情報を利用する場合には，観測による物体の認識の困難性に加え，物体の質量など観測が困難な情報を補う必要がある．このシステムを用いて近い将来の予測をしながら行動するロボットを作成することが今後の課題である．

参考文献

[1] Fumio KANEHIRO, Kiyoshi FUJIWARA, Shuuji KAJITA, Kazuhito YOKOI, Kenji KANEKO, Hirohisa HIRUKAWA, Yoshihiko NAKAMURA, and Katsu YAMANE. Open architecture humanoid robotics platform. In *Proceedings of the 2002 IEEE Intl. Conference on Robots and Automation (ICRA'02)*, pp. 24–30, 2002.

```
(defun test-box ()
  (d-init)
  (setq *boxes* nil)
  (dotimes (i 3)
    (dotimes (j 5)
      (setq b (make-cube 10 20 20
                :weight 1 :color :yellow))
      (send b :locate
       (float-vector 0 (* j 20) (+ 10 (* i 20))))
      (push b *boxes*)))
  (setq *ball*
        (make-sphere 10 :weight 20 :color :red))
  (send *ball* :locate #f(100 -100 10))
  (objects (list *boxes* *ball*))
  (send *ball* :set-linear-vel #f(-1.5 2.0 0))
  (send *ball* :set-angular-vel #f(-5 -4 0)))
```

Fig.4 test-box source code

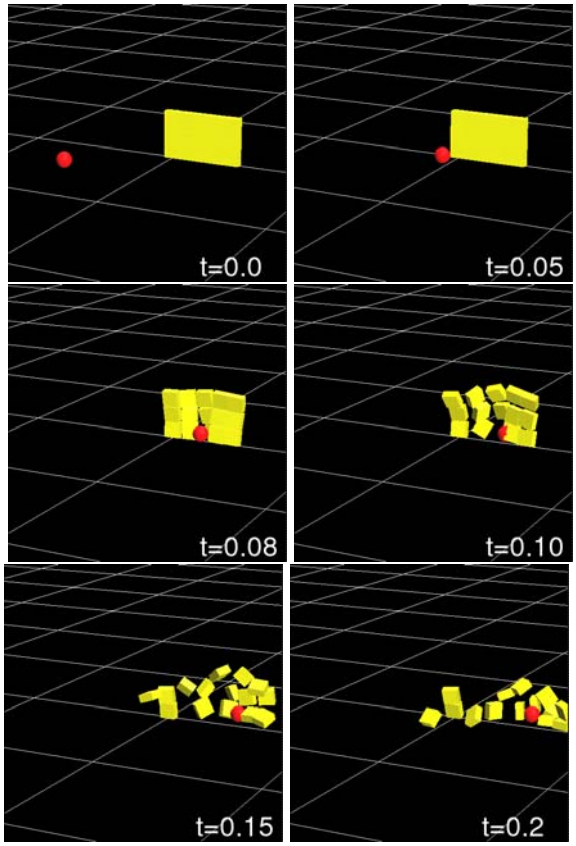


Fig.5 test-box simulation result

```
(defun test-stairs ()
  (d-init)
  (let (stairs b bb)
    (dotimes (i 10)
      (setq b
            (make-cube (* 100 (- 10 i)) 1000 100
                       :dynamics nil))
      (send b :locate
            (float-vector (* -50 i) 0 (* 100 i)))
      (if stairs
          (setq stairs (body+ stairs b))
        (setq stairs b)))
    (putprop stairs :green :face-color)
    (setq bb (make-ode-robot #'kaz))
    (send-all bb :locate #f(-390 0 950))
    (objects (list stairs bb))))
```

Fig.6 test-stairs source code

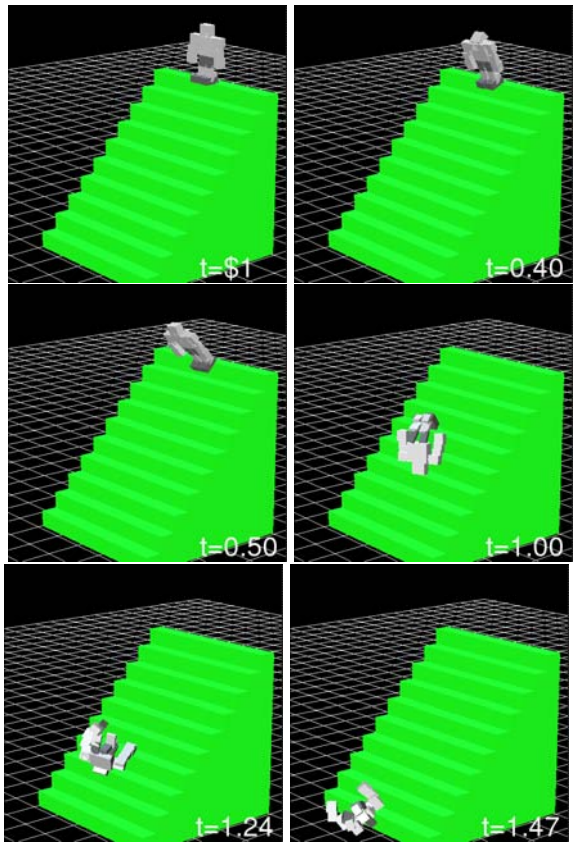


Fig.7 test-stairs simulation result

[2] 金広文男, 稲葉雅幸, 井上博允. ゲーム用高速動力学演算パッケージを用いたロボットボディの仮想化. 日本機械学会ロボティクスメカトロニクス講演会 2001, pp. 2P2–H3, 2001.

[3] Fumio Kanehiro, Masayuki Inaba, and Hirochika Inoue. Action acquisition framework for humanoid robots based on kinematics and dynamics adaptation. In *In Proc. of the 1999 IEEE International Conference on Robotics and Automation (ICRA'99)*, pp. 1038–1043, 1999.

[4] 幸坂大輔, 岡田慧, 稲葉雅幸, 井上博允. ヒューマノイドにおける先読み行動システムの構成法に関する研究. 日本機械学会ロボティクスメカトロニクス講演会 2004, pp. 2P1–H73, 2004.

[5] Java3D API. http://java.sun.com/prodects/java-media/3D.

[6] VPython. http://www.vpython.org.

[7] 松井俊浩. 幾何モデリング機能を備えたマルチスレッド並列オブジェクト指向言語 EusLisp. 日本ロボット学会誌, Vol. 14, No. 5, pp. 650--654, 1996.

[8] Open Dynamics Engine ODE. http://ode.org.

[9] Philoblock. http://cemm.educ.monash.edu.au/research/philoblock.