ヒューマノイド行動ソフトウェア基盤における マルチスレッドLispへの実時間GC機能の導入

岡田 慧　花井 亮　神崎 秀　稲葉 雅幸　湯淺 太一

新しいソフトウェアの応用領域として実世界において認識行動処理を行うヒューマノイドを対象とし，実世界知識表現の上位の記述から，認識行動の並列性記述，リアルタイム性の必要な反射系記述を扱える大規模かつ複雑なヒューマノイドソフトウェアを効率よく開発するためのプログラミング環境を構築することを目的としている．本論文は並列性，実時間性，対話性，三次元モデリング機能を特徴とする Lisp 言語を基盤にしたヒューマノイドのシステムプログラミング環境 EusLisp に対し，ごみ集め処理 (GC) による実行停止時間により関節角度の更新が停止することないよう実時間 GC 機能を導入し，ロボットの運動制御ソフトウェアに対する評価を行った．

In this paper, we propose humanoid programming environment as a new application area for software system. In order to realize productive programming environment for developing complex humanoid software that is required to include high-level description of real-world representation, concurrent processing of recognitive behaviors and real-time/reactive modules, we adopt the EusLisp which was designed for robot software application and features such as interactive, processing, 3D geometrical modeling, multi-threading and automatic memory management. However, suspension time caused by a garbage collector prevents periodic update of joint angles and sometimes induces serious malfunctions. Therefore we have implemented a real-time GC to reduce suspension time and proceed with GC effectively without suffering memory starvation.

1 はじめに

近年のヒューマノイドロボット研究ではハードウェアプラットフォームの開発が進み，高機能なソフトウェア基盤の重要性が高まってきている [1–4]．そこでは実世界知識表現の上位の記述から，認識行動の並列性記述，リアルタイム性の必要な反射系記述を扱う大規模かつ複雑なプログラミング環境を持ったロボットシステムの構成法が課題となる．

これまでに筆者らは三次元モデリング機能とマルチスレッド機能を特徴とするオブジェクト指向型 Lisp 言語 [5] 上に，環境と身体の幾何モデルを中核とした上位の知能ロボットソフトウェアを構築してきている [6]．しかしこの環境では Lisp におけるごみ集め (GC) 処理の実行停止時間が問題となり実時間性を必要とされる下位の動作ソフトウェアの実現が困難であった．下位の動作ソフトウェアの例としては，ロボットが倒れないようにするためのバランス制御や，タイミングを考慮しながら全身の関節を動かす歩行動作が挙げられる．

そこで本研究では実時間 GC 機能 [7,8] を導入することでごみ集め (GC) 処理の実行停止時間の問題を解決し，上位から下位までの大規模かつ複雑な行動プログラムを統一して扱えるソフトウェア基盤を構築することを目的とし，ヒューマノイドソフトウェアの構成法の構造化・体系化を目指すものである．

Real-Time Garbage Collection on Multi-thread Lisp for Humanoid Behavior Software System

Kei OKADA, Shigeru KANZAKI, Masayuki INABA, 東京大学大学院情報理工学系研究科, Graduate School of Information Science and Technology, The University of Tokyo.

Ryo HANAI, Taiichi YUASA, 京都大学大学院情報学研究科, Graduate School of Informatics, Kyoto University.

2 ヒューマノイドの行動ソフトウェア基盤

これまで図 1 に示すようなヒューマノイドロボットの行動ソフトウェア基盤のプロトタイプを実現してきている [9–11]．このソフトウェア基盤は，(1) 実機を利用する感覚でインタラクティブに動作を検証できる行動シミュレーション環境により，実ロボットを用いた実験での誤動作による破損や故障等のリスクを回避できる．(2) インタプリタ言語環境を用いることで，行動プログラムの逐次的な実行・評価が可能になり開発の効率が上がる．(3) 透過的なソフトウェア構成により実機とシミュレータで同一の認識行動プログラムが可能．の特徴を持ち行動プログラムの反復的・漸進的開発が容易になっている．

これまでにこのソフトウェア基盤上でロボットと環境のモデリング，視覚による環境地図の作成，経路計画，移動動作の実行，環境の変化の認識と経路の再計画，対象物の把持を含む視覚誘導型動作などの上位の行動プログラムを実現してきている．

一方，実時間管理が重要になるモータサーボループ，動作軌道補間，安定化制御や身体のコンプライアンス制御等の下位の反射系動作プログラムの記述は，実時間 OS を用いたリアルタイムモジュールや体内マイコン上のプロセスとして実現してきた．

3 マルチスレッド Lisp への実時間 GC 機能の導入

ロボット制御の分野で一般的に十分とされている制御周期は，ロボット機構の固有振動数である 1-50[Hz](周期 20-1000[msec]) の 1/10 程度の 2-100[msec] とされている [12]．そこで GC によって許されるアプリケーションの再開の遅れの目安として 100[μsec] をターゲットとした．このような実時間 GC の実現には (1) アプリケーションの実行の妨げになる処理を排すること (2) 適切な GC の進め方によりアプリケーションによるメモリ要求時に答えられるだけのヒープの空き領域の確保，が必要になる．そこでリターンバリア方式によるインクリメンタル処理と GC スレッドを併用する方法 [13] を利用する．

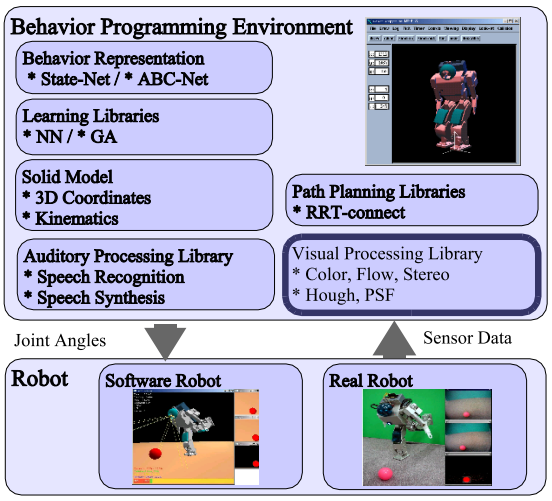

図 1 ヒューマノイドの認識導行動ソフトウェア環境

3.1 リターンバリア付きインクリメンタル GC

アプリケーションが GC 処理に妨げられることなく実行を続けるためには非プリエンプティブな GC 処理の単位を小さくする必要がある．

そこで本研究ではスナップショット方式にリターンバリア方式を組み合わせた手法 [8] を採用した．これは，マーク&スイープ方式をインクリメンタル化することで小さな停止時間を実現するものである．

スナップショット方式は GC 開始時にルートから到達可能であったオブジェクトが全てマークされることを保障する手法であるが，ルートスタックが大きいとスキャンに時間がかかるという問題がある．リターンバリア方式はルートの中のスタックは実行中の関数のフレームしかアクセスされないことに注目し，スタックのカレントフレーム以外のスナップショットをとるのを後回しにする手法である．

これにより，(1) 複数のスレッドが存在する場合，イベント待ちになっているスレッドのスタックスキャンを後回しにできる．(2) 再帰関数等により関数が連続リターンしているような場合，自分より優先度の低いスレッドでのフレームの連続スキャンがおこらない．という特徴を持つため，EusLisp の特徴であるマルチスレッド処理時において特に効果を発揮すると考えた．

表 1 ルートスキャンにおける停止時間 [μsec]

スレッド数	平均	最大	GC 回数
1	31.0	32.0	40
2	40.2	78.0	40
3	47.0	97.0	43

3.2 GC スレッドとインクリメンタル処理の併用による GC 制御

実時間 GC を実現するためには，適度にヒープに空きがある状態で GC を開始し，アプリケーションのメモリ要求に応えられない事態に陥る前にメモリ回収を行わなければならない．インクリメンタル GC はメモリ割当量に応じて GC 処理が進むため，飢餓状態の回避に向いているが CPU の空き時間を有効に使えない，バースト的なメモリ要求に対して GC 処理の負荷が大きくなりすぎる，という問題がある．一方，専用スレッドを用いて GC 処理を行う場合は飢餓状態の回避はスケジューラに依存する．そこで，GC スレッドとインクリメンタル GC を併用する．GC スレッドは低い優先度で実行し CPU の空き時間に GC を進め，インクリメンタル GC はメモリ割り当て時に GC を行う．

3.3 ルートスキャン処理停止時間の評価

実装を行った GC の評価として，ルートスキャン処理における停止時間 (ヒープの空きが少なくなったことを検出したスレッドが GC 開始要求を出してから実時間スレッドが処理を再開するまでの時間) の測定を行った．

測定環境には実時間 OS(ART-Linux 2.4.19) を搭載したマシン (PentiumM 1.7GHz, 主記憶 1GB) を用い複数のスレッドで別々に FFT を実行した．スレッドのうち 1 つは 10[msec] 周期で実行される実時間スレッドで，残りは通常のスレッドである．表 1 ではスレッド数の増加に伴い停止時間が増加しているが，これは全ての (イベント待ちでない) 実行中のスレッドがポーリングを行うまで GC を開始できないためにおこる停止時間，および OS レベルでのコンテキストスイッチにかかるオーバヘッドが加算されていると考えられる．

JSK-HOAP2 仕様	
身長	480[mm]
体重	6.5[kg]
自由度	24 (足 6, 腕 4, 首 2, 手 1)
センサ	姿勢センサ 足裏圧センサ ステレオ視覚
体内 LAN	USB 1.1
OS	RT-Linux

図 2 HOAP-2 ロボット

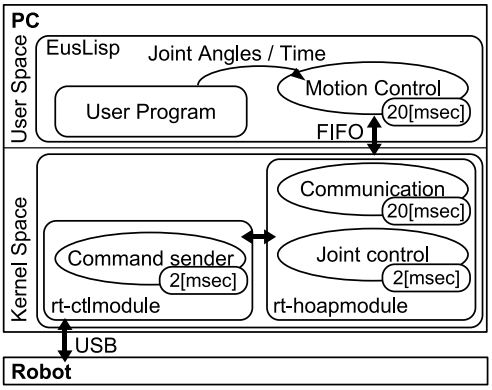

図 3 HOAP2 ロボットの制御プログラムの構造

しかしながら，スレッド数の増加に伴う同期コストの増加は OS とハードウェアのアーキテクチャに依存するが無視できるとし，実際上はスレッド数の増加に伴い CPU 台数を増加することで，CPU あたりのイベント待ちでない実行中のスレッドがそれほど多くならないようする．従って，ほぼ 100[μsec] の停止時間を実現できているとみて問題ない．

また，1[msec] の周期タスクを用いて 500[Hz] の音出力を行うプログラムを実行した．このときバックグラウンドでジョブを走らせ，実時間 GC を持った EusLisp 環境では周期のズレが生じていないことを確認した．

4 実時間マルチスレッド Lisp でのヒューマ

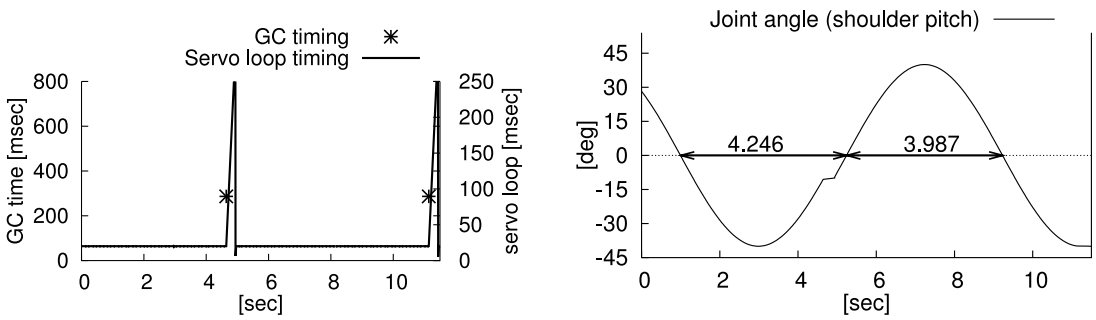

図 4　一括 GC 機能を有する EusLisp を用いてロボットの動作を生成した場合.
左）GC のタイミングとサーボループの周期，右）関節角度

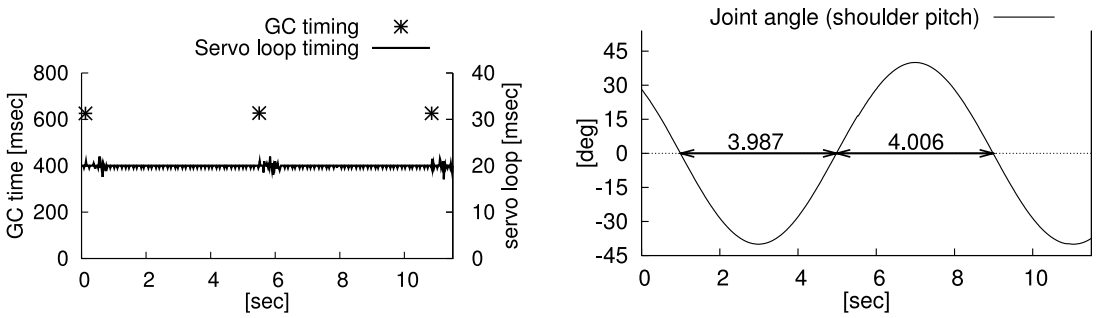

図 5　実時間 GC 機能を有する EusLisp を用いてロボットの動作を生成した場合.
左）GC のタイミングとサーボループの周期，右）関節角度

ノイド動作生成実験

4.1　ヒューマノイドソフトウェア環境

実際のロボットを用いた動作生成実験において，従来の一括 GC を用いた場合と，実時間 GC を用いた場合のデータを挙げる．実験に用いたロボットは図 2 に示す研究用ヒューマノイドロボット HOAP-2 [14] であり，ソフトウェア環境を図 3 に示す．ロボット身体と RT-Linux を搭載した制御用 PC は USB により接続されている．RT-Linux は標準 Linux の拡張により実現された実時間 OS であり，Linux カーネルのスケジューラには手を加えずに実時間スケジューラを追加したものである．ここでは標準 Linux カーネルを実時間スケジューラの 1 プロセスとして扱い，実時間タスクはカーネルモジュールとして実現している．

EusLisp はユーザ空間で動いているため，GC スレッドとの優先度差を設け，スケジューリングポリシーを POSIX スレッドの機能を利用し SCHED_RR(ラウンドロビン) に設定した．このシステムでは図 3 のように EusLisp 側の動作制御用スレッドが実時間タスク (rt_hoapnerve) に目標関節角度を書き込む．書き込みは 20[msec] 周期毎に FIFO 経由で行われる．

rt_hoapnerve に書き込まれた目標関節角度は，共有メモリを介して 2[msec] 周期で補間され rt_ctlmodule へ送られる．rt_ctlmodule はロボットに対して低レベルのコマンドにより制御を行う実時間タスクであり，与えられた全関節角度のデータを 2[msec] 周期でロボット側に送る．

4.2　ヒューマノイドソフトウェア環境上での実時間 GC 性能評価

ユーザ空間に存在する EusLisp 上で適当に関節角度を制御するようなプログラムを書きロボットを動作させた．動作は 8 秒で 1 周期となるような簡単な屈伸運動である．プログラムは EusLisp 側の動作制御用スレッドの周期である 20[msec] 毎に全身の関節角

度データを更新するようにしてある．

これによりロボットを動作させた際のデータを図 4 と図 5 に示す．図 4 は実時間でない一括 GC を用いた従来の EusLisp であり，図 5 は実時間 GC 機能を持った EusLisp を用いたものである．それぞれ左のグラフは GC と EusLisp 側の動作制御用スレッドのタイミングと処理時間を示し，右のグラフは実際のロボットの関節角度を示している．

一括 GC を用いた場合 (図 4) では 6 秒に一回程度の GC がおこり 300[msec] 程度の停止時間がある．

また，動作制御用スレッドの周期は，GC が走っていない期間は 20[msec] であるが，GC が走っている期間は GC による停止のため 250[msec] 程度に延びる．この結果，目標姿勢の定期的な更新がなされず関節角度の曲線が乱れていることがわかる．

一方実時間 GC を用いた場合 (図 5) では 5 秒に一回程度 GC が開始し約 600[msec] 程度をかけてインクリメンタルに処理している．また，GC 処理が走っても EusLisp 側の動作制御用スレッドの周期実行時間は一定であり，関節角度に影響が出ていないことがわかる．

4.2.1 歩行動作時の実時間 GC の性能評価実験

ヒューマノイドロボットの歩行動作時の実時間 GC の性能評価を行った．ヒューマノイドの歩行動作は一般的に ZMP(ゼロモーメントポイント) が床に接地している足底面内にあるように生成した全身の運動パターンを実行することで実現する．ここで，ZMP はロボットの各リンクの重力と慣性力の合力である総慣性力のモーメントが 0 になる足裏の点と定義される．たとえばロボットを質量台車型モデルに近似した場合，ロボットの重心の位置を x，高さを z_h，重力加速度を g としたときの ZMP の値 p は以下で表すことができる．

$$p = x - \frac{z_h}{g}\ddot{x}$$

このとき，動作パターン実行時に GC による停止時間が発生すると，$\ddot{x}$ の値が変化することで ZMP の値が動作パターン生成時と変わり，安定した歩行動作が行えなくなることが予想できる．

このことを確認するために実験を行った．実験では予見制御を用いた歩行動作パターン生成法 [15] を用いて一歩 1.3 秒の歩行パターンを生成し，100[msec] 毎の歩行動作姿勢を図 3 の動作生成用スレッドへ入力する．動作生成用スレッドは歩行動作姿勢を補間し 20[msec] 周期で実時間タスクへ書き込む．GC による停止時間が生じるとこの動作生成スレッドの 20[msec] 周期を保つことができなくなり結果として目標姿勢の時間間隔が一定でなくなってしまう．

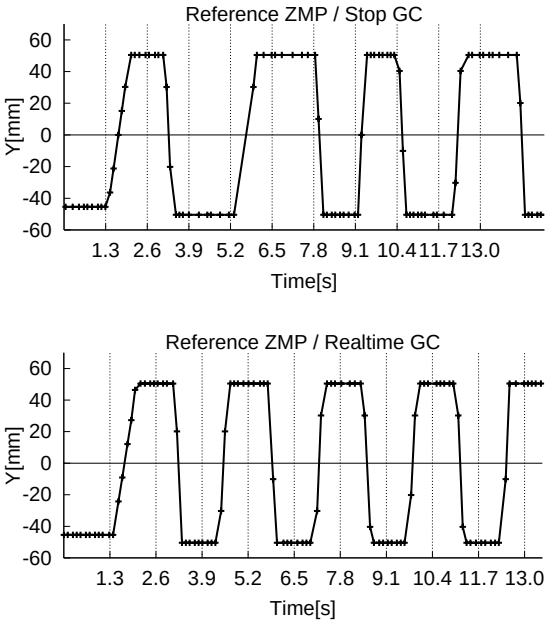

図 6 歩行動作時の目標 ZMP 位置．
上）一括ＧＣの場合，下）実時間ＧＣの場合

図 6 にこの結果を示す．横軸が時間，縦軸がロボットへ送信した目標姿勢から計算した ZMP の y 軸（ロボットの正面からみて左側を正とする）の値である．実時間 GC 利用時は ZMP の軌道に乱れがなく，一歩 1.3 秒毎に ZMP が右足から左足へ（あるいはその逆へ）遷移していることがわかる．一方，一括 GC 利用時は GC の発生によって ZMP の軌道が乱れていることがわかる．

このときのロボットの動作の様子を図 7 に示す．上段が一括 GC 利用時，下段が実時間 GC 利用時である．一括 GC を用いた場合は GC 処理によってロボットの全身のバランスを崩していることがわかる．

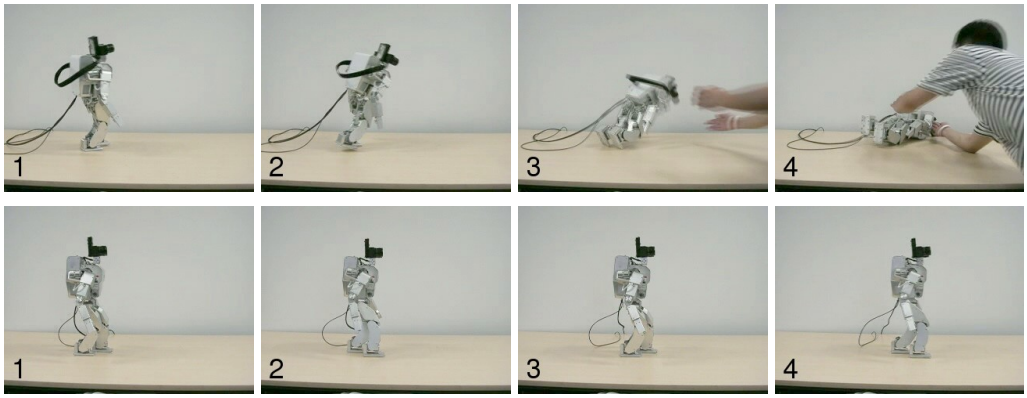

図 7 歩行動作によるＧＣ機能の性能評価．上）一括ＧＣがおこった場合，下）実時間ＧＣ環境での歩行動作

5 おわりに

本論文では，ソフトウェアの新しい応用領域として実世界において認識行動処理を行うヒューマノイドを対象としたソフトウェア基盤の構成法について述べた．具体的には筆者らが構築してきているヒューマノイドのソフトウェア基盤の記述言語である EusLisp に，スナップショット方式とリターンバリア方式の併用，ならびに GC スレッドとインクリメンタル処理の併用を特徴とする実時間 GC 機能を導入し，実ロボットでの検証実験を通じて評価を行った．

実時間 GC を導入することで実時間スレッドそのものは 1[msec] 程度の周期タスクの実行が可能であるが，実験に用いたロボットシステムでは 20[msec] の周期タスクの実現が限界となっている．これは処理系の問題ではなく，ロボットの幾何モデルを使った姿勢計算にかかる時間がリミットとなっている．

本稿で扱った歩行実験ではフィードバック制御のないオープンループ制御である．今後，歩行動作を生成するスレッドと，センサ情報に基づいてバランス制御をするスレッドを組み合わせること，さらに視覚により歩行経路をオンラインで計画するスレッドを組み合わせることによって，より大規模で複雑な行動を実現させ，我々の行動ソフトウェア基盤の有効性の高さを実証することが必要である．

謝辞 本研究の一部は，科学研究費補助金特定領域研究「情報学」（領域番号：006）「マルチスレッド Lisp の実時間 GC 機能の導入とヒューマノイド行動の実現」（16016214）のもとで行われました．

参 考 文 献

[1] 稲葉雅幸, 加賀美聡, 西脇光一. 岩波講座 ロボット学 7 ロボットアナトミー. 岩波書店, 2005.

[2] 金広文男, 藤原清司, 梶田秀司, 横井一仁, 金子健二, 比留川博久, 中村仁彦, 山根克. 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] Y. Kuroki, B. Blank, T. Mikami, P. Mayeux, A. Miyamoto, R. Playter, K. Nagasaka, M. Raibert, M. Nagano, and J. Yamaguchi. Motion Creating System for A Small Biped Entertainment Robot. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*, pp. 1394–1399, 2003.

[4] K. Okada, T. Ogura, A. Haneda, J. Fujimoto, F. Gravot, and M. Inaba. Humanoid Motion Generation System on HRP2-JSK for Daily Life Environment. In *Proceedings of 2005 IEEE International Conference on Mechatronics and Automation (ICMA05)*, pp. 1772–1777, 2005.

[5] 松井俊浩, 関口智嗣. マルチスレッドを用いた並列 Euslisp の設計と実現. 情報処理学会, Vol. 36, No. 8, pp. 1885–1896, 1995.

[6] M.Inaba, S. Kagami, F. Kanehiro, Y. Hoshino, and H. Inoue. A Platform for Robotics Research Based on the Remote-Brained Robot Approach. *The International Journal of Robotics Research*, Vol. 19, No. 10, pp. 933–954, 2000.

[7] T. Yuasa. Real-Time Garbage Collection on General-Purpose Machines. *The Journal of Systems and Software*, Vol. 11, No. 3, pp. 181–198, 1990.

[8] 湯淺太一, 中川雄一郎, 小宮常康, 八杉昌宏. リターン・バリア. 情報処理学会論文誌, Vol. 41, No. SIG 9(PRO 8), pp. 87–99, 2000.

[9] M. Inaba. Remote-Brained Robotics: Interfac-

ing AI with Real World Behaviors. In *Robotics Research: The Sixth International Symposium*, pp. 335–344. International Foundation for Robotics Research, 1993.

[10] 金広文男, 稲葉雅幸, 井上博允. 人間型ロボットの発展的実現のためのソフトウェアシステム構成法. コンピュータソフトウェア, Vol. 17, No. 6, pp. 52–56, 2000.

[11] K. Okada, Y.Kino, F. Kanehiro, Y. Kuniyoshi, M. Inaba, and H. Inoue. Rapid Development System for Humanoid Vision-based Behaviors with Real-Virtual Common Interface. In *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS'02)*, pp. 2515–2520, 2002.

[12] 日本ロボット学会編. 新版ロボット工学ハンドブック. コロナ社, 2005.

[13] 花井亮, 岡田慧, 湯淺太一, 稲葉雅幸. ロボット行動ソフトウェア環境に適した実時間ごみ集め. コンピュータソフトウェア, Vol. 22, No. 3, pp. 173–178, 2005.

[14] 村瀬有一, 安川裕介, 境克司, 植木美和. 研究用小型ヒューマノイドの設計. 第 19 回ロボット学会学術講演会予稿集, p. 3A18, 2001.

[15] 梶田秀司, 金広文男, 比留川博久. 予見制御理論を応用した 2 足歩行パターン生成. 第 20 回ロボット学会学術講演会予稿集, p. 1D23, 2002.