学術論文

リモートブレイン方式における ソフトウェアプラットフォームの構造化と実現

加賀美 聡[*1] 稲葉 雅幸[*2] 井上 博允[*2]

Construction and Implementation of Software Platform in Remote-Brained Robot Approach

Satoshi KAGAMI[*1], Masayuki INABA[*2] *and* Hirochika INOUE[*2]

This paper describes a software framework which can support different types of bodies and behaviors with different processing times. The system we propose consists of 3 parts, "*Mother, Brain, Sensor&Actuator*". *Brain* consists of robot behavior programs and their supporting libraries, and *Sensor&Actuator* consists of sensor processing and actuator control system. *Mother* consists of tools to produce and evolve *Brain* programs.

In building tools and libraries for such a platform you are faced with two major problems. One is, system architecture will always have a trade-off relationship with "extensibility" and "shareablity". The other problem you encounter is the difference in execution time between the low-level real-time reactive environment and the high-level behavioral software.

In this paper we describe the methods and the software environment that we built which overcomes the problems mentioned above, based on the remote-brained robotic approach. This system enables us to develop flexible multi-process networks made of processes with differing computing times which in turn enables the contruction of various brain architectures.

Key Words: Remote-Braind Robotics, Robot Architecture, Parallel Distributed Processing System.

1. はじめに

リモートブレイン方式 [1] とはロボットの脳とボディを物理的に分離し，自由度の高いボディを，高速な CPU ，大規模なメモリを持つ計算機上のソフトウェアから動かすことにより，知能ロボットの研究を行なうパラダイムである．このアプローチによりロボットの研究において，学習，モデル，シミュレータという複雑で大規模なソフトウェアと，それを実現する Lisp や論理型言語等の言語上での並列プログラミング環境，の 2 点を実現し，知能ロボットの構成法を実証的に研究することが可能になってきた．

実験を通して実証的に研究を行なっていく時には，ソフトウェアの改良を繰り返すプロセスが長期間に渡って必要となる．実世界でよく動く知能ロボットの研究において，ソフトウェアを改良するプロセスをどのような環境でどのように行なうべきかが重要な問題となる．

原稿受付 1996 年 2 月 2 日
[*1] 東京大学大学院工学系研究科情報工学専攻博士課程
[*2] 東京大学工学部機械情報工学科
[*1] Graduate School, The University of Tokyo
[*2] Dept. of Mechano-Informatics, The Univ. of Tokyo

そこで本論文ではソフトウェア環境を複数の開発者が共有でき，環境そのものもロボットのトップレベルに合わせて進化できるソフトウェア環境としての「ソフトウェアプラットフォームの構成法」をテーマとする．ロボットのトップレベルのソフトウエアは，このプラットフォームのアプリケーションとなる．

実世界で行動するロボットにおいて知能処理や視覚処理といった大規模なソフトウェアからロボットを動かす環境としては，ハンドアイシステムにおいて COSMOS [2], Handy [3], 移動ロボットにおいて Navlab [4], Chatila [5], Yamabiko [6], Polly [7] 等の研究が行なわれているが，これらは特定の簡単なボディに対して，そのボディで行なわれるタスクを実現するための開発環境であった．

これまで実際にボディと大規模なソフトウェアを統合可能なリモートブレインロボットの環境において，様々な形態のロボットを作成し，そのトップレベルソフトウェアを繰り返し発達させる [8] と共に，このトップレベルソフトウエアをアプリケーションとするプラットフォームの発達の研究 [9,10] を行なってきた．本論文では，この問題を整理し，ソフトウエアプラットフォームの階層構造と，システム構成における拡張性と共有の問題，実時間並列処理システムについて述べ，作成したソフトウェアプラットフォームと，このシステムを用いたマルチロボッ

トのアプリケーションについて述べる．

2. 知能ロボット研究のためのソフトウェアプラットフォームにおける設計問題

2.1 知能ロボットシステムの構成法

処理の観点から大規模な知能ソフトウエアから実際にロボットを動作させるシステムの構成を整理すると，センサやアクチュエータの周期により律則された実時間処理を行う *Sensor&Actuator* と，そのセンサデータをもとに非実時間な行動決定のプロセスの周期に律則された知能処理を行い行動を出力する *Brain* の 2 種類がある（図 1）．また多様なロボットを行動させるための共有性と拡張性をそなえたプラットフォームは，その構成要素を生成，あるいは開発する環境を備えなければならない．ここではそのロボットを作る主体が用いるための環境を *Mother* と呼ぶ．

知能ロボットのシステムにおいては，この *Mother*, *Brain*, *Sensor&Actuator* の各階層の構成法と構成要素，およびそれらの間のインターフェースが重要な課題となってくる．また階層化とそのインターフェースを定義することにより，各階層の独立した開発が可能となるが，それらの成果を直接統合したり，上部のロボットを行動させるトップレベルのソフトウエアに全く新しい要素機能が付け加わったときに，その変化に各階層の構成要素が対応可能な拡張性が必要である．

2.2 ソフトウェアの共有と拡張性のトレードオフの調整

大規模なソフトウエアを複数の開発者が構築していく上で，システムの機能の共有性と拡張性が重要であるが，これらはトレードオフの関係となる．一般的にオブジェクト指向により共有性を高めるが，共有性をあげるためにはオブジェクトは根が全ての親となるツリー構造で，枝の分岐するノードでは子孫のノードの共有部分はなるべく上位に含まれるようにクラス階層を設計する（これを縦型と呼ぶことにする）．従って，様々な機能を全てメソッドとして持ち，必要な内部変数も全てそのスロットとして持つような一つの大きなオブジェクトを作成することにより，ライブラリが製作される．

しかし，クラスの持つ機能の変更や追加はシステム全体へ大きく影響することからクラスの階層を縦型にすることには，保守，管理の手間を増大させる．またグループで作業する際には基本クラスへの変更が行なわれる時には，全てのライブラリに変更が必要になる可能性があり，同時開発を不可能としてしまう．

2.3 実時間環境モニタと非実時間上位ソフトウエアとのインターフェース問題

実世界を行動するロボットでは常時まわりの環境をスキャンしているモニタ式のセンサ処理が重要である．また即応性のあるシステムを構築するためには実時間で処理の結果が得られる必要がある．

一般にロボットのセンサ処理，動作制御システムは並列に動作する複数のプロセッサ上に置かれる．このため，環境モニタ型の処理を行なうシステムにおいては，処理を複数並列に動作させる並列ライブラリと，システム上で複数のセンサ処理を分散して行なう並列処理マネージャ，その処理結果を上位のソフトウエアに実時間で渡すインターフェースの機構の３つが必要となる．

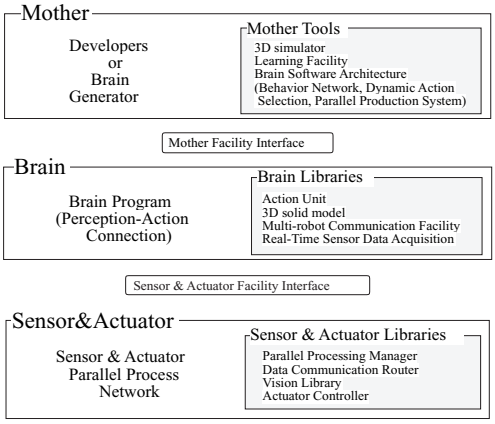

Fig. 1 Robot Software Components

3. リモートブレインプラットフォームの設計

ソフトウエアプラットフォームは複数の開発者が共有してそれぞれのロボットボディのブレインプログラムを開発し，その進歩にともなって同時にプラットフォームそのものも拡張していくための枠組みである．

リモートブレインアプローチによりシステムの構成はブレインとボディを物理的に分離し，ブレイン側のソフトウエアについて外界からの入力により律則された実時間処理を行う *Sensor&Actuator* 部をトランスピュータネットワークで，その処理結果から非実時間の知能処理により行動を出力する *Brain* 部を WorkStation のネットワークでそれぞれ行う．以下では第２節で述べた２つの問題に関してその方針を述べる．

3.1 *Mother* 部のプログラムレベルツール

Mother はロボットの振舞いを生成するソフトウエアであり，開発者が直接ロボットの振舞いを開発したり，動作中のインスペクションやデバッグを行なうためのツール，振舞いを生成するソフトウエアを開発するための環境として３次元シミュレータや学習システムのライブラリ，*Brain* のソフトウエアを階層化するライブラリ（Behavior Network, Dynamic Action Selection, Parallel Production System, State Transition Network）等をツールとして用いることができる．これらのライブラリ（ツール）は Lisp 上で開発することから，通常の縦型のオブジェクト階層とする．

3.2 *Brain* 部の縦型構成ライブラリ

Brain はロボットのトップレベルのソフトウエアがおかれ，ロボットの行動ライブラリや，ボディや環境の３次元ソリッドモデル，マルチロボットの通信機能をライブラリとして用いることができる．ここでもライブラリは Lisp 上で開発することから，通常の縦型のオブジェクト階層とする．

3.3 *Sensor & Actuator* 部の横型ファンクションの並列処理構成

Sensor&Actuator はロボットのセンサ信号処理，動作制御を行ない，通常は複数のプロセッサのネットワークにより構成される．ここには並列処理マネージャ，プロセッサ間のデータ通

信用ルータ，センサ処理ライブラリ，アクチュエータコントローラ等が存在する．ここでは並列処理が基本となることから，拡張性を優先するために選ばれた基本部分以外の全てのスロット変数やメソッドを共有しないでユーザが用いる機能オブジェクトのクラスに含まれるようにクラス階層を設計する（これを横型と呼ぶことにする）．

3.3.1 ウインドウ主体のモジュラ型並列視覚処理ライブラリ

センサ処理部の画像処理においては，機能レベルのオブジェクト指向の単位はウインドウ（矩形領域）処理とする．ここで，画像処理を行なうウインドウはタスク固有ともいえるほど頻繁に新しい処理を作成する必要があることと，一度作成したウインドウもさまざまな機能の変更や追加を行なう必要がある．

しかし並列処理ライブラリにおいて，縦型のオブジェクト階層を作成することは，その変更が非常に困難になることから，ソフトウエアの共有性を高めるために，横型のオブジェクト階層を作成する．

3.3.2 制御構造の基本枠組を与える並列処理マネージャ

センサ機能が環境を常にモニタするために，与えられたセンサ処理を複数のプロセッサに分配し，常時センサ処理を行なう並列処理マネージャを開発する．図 2 の Servo & Vision Processes が並列に立ちあげられたセンサ処理機能である．またマネージャが複数のプロセッサに処理を割り当てたり，実時間で処理結果を取得するためにはプロセッサ間の通信を行なうルータ機能が必要である．

3.3.3 処理結果の実時間取得の為の非同期インターフェース

実時間で処理結果を返す方式としては，(a) 各プロセッサ上のセンサ処理プロセスまで上位のソフトウェアから問い合わせる，(b) 下位のプロセッサの最も上位に近い場所で，センサ処理プロセスの最新の結果を全てバッファリングして，上位のソフトウェアからの問い合わせには，そこから答える，(c) センサ処理の結果を上位のソフトウェアに事象駆動的にセットし，上位のソフトウエアはその値を用いる，の３つの方式が考えられる．

これらの３つの方法の選択はシステムの持つ，通信能力とシステム形態に強く依存するが，以下にその特徴を比較する．(a) の方法は最も単純で必要なときにのみ通信が発生することから，下位のシステムへの負荷が小さいが結果を得るのに時間がかかる点が欠点である．(b) の方法は上位と下位の間の通信量が小さい点が特徴である．従って，下位のシステムは十分な通信能力があるが，上位と下位を結ぶ部分が比較的細い場合に有効である．(c) の方法は上位のソフトウェアから見て，最も自然に処理結果を利用できるが，上位の Lisp の処理系で並列処理の機能が必要になること，上位と下位のソフトウェアを結ぶ部分での通信量の増大が問題となる．

ここでは実際のシステムの上位と下位の通信性能から (b) の方法を採用する (図 2)．

4. リモートブレインプラットフォームの実現

Mother と *Brain* の実現法はさまざまな研究が行なわれているので，本論文ではそれらの下位で並列に用いられている *Sensor & Actuator* 部に焦点をあててその実現法と *Brain* 部とのインターフェースの詳細を述べる．

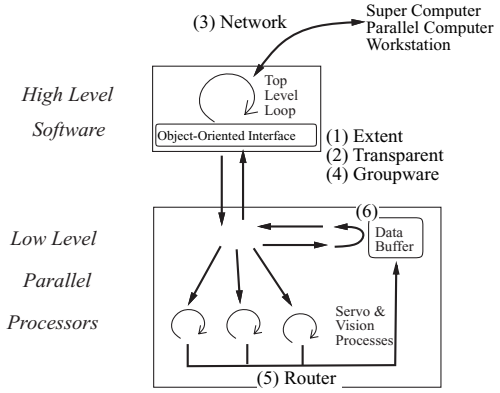

Fig. 2 Real-Time Result Sensor Data Transmition

4.1 システムの特徴

開発したリモートブレインロボットのためのソフトウエアプラットフォームの *Sensor & Actuator* 部の構成図を 図 2 に示す．システムの特徴について以下に述べる．

(1) システムの拡張性を持たせるため各視覚機能は３次元モデラを持つオブジェクト指向 LISP である Euslisp/MT [11] の上でオブジェクトとして実現，

(2) トップレベルのソフトウェアから，実際には複数のプロセッサ上に存在する視覚機能オブジェクトの実体をスロットやメソッドとして利用する形で透過性をもたせ，その実体は並列処理機能を利用するために横型の構成とした，

(3) ネットワーク上で複数の *Brain* のプログラムと下位の *Sensor&Actuator* のシステムの間の通信機能を実現 [12] した，

(4) *Sensor & Actuator* 部では 2.2 節で述べた共有と拡張のトレードオフの問題に関して，複数の開発者がグループウエアとして同時に開発できるように，機能レベルで横型のオブジェクト構造を持たせ，視覚機能を追加する際に必要な変更部分をまとめ，拡張を容易にするインターフェースの仕様を定めた，

(5) *Sensor & Actuator* 部ではトランスピュータを CPU とする Vision Board を複数枚，イメージバスに並列に接続したハードウエアの上に実現されるが，それぞれの CPU 間の通信を行なうルーターを実現 [13] してデータの受渡しを可能とし，(4) で標準化したインターフェースから複数プロセッサ上の実体を扱う機構を実現する．

(6) *Sensor & Actuator Facility* インターフェースは，上位と下位の結合部分が比較的細いことから，実時間性のあるシステムを構築するために，ここでは 3.3.3 節で述べた (b) 「下位のプロセッサの最も上位に近い場所で，センサ処理プロセスの最新の結果を全てバッファリングして，上位のソフトウェアからの問い合わせには，そこから答える」方法をとる．

4.2 *Sensor & Actuator* 部の実現

4.2.1 データ通信用プロセス間ルータの実現

システムはトランスピュータのネットワークの上に構築されている．システムの開発において各プロセッサ間のデータのやりとりを簡略化するために，任意のデータをネットワークの任意のプロセッサへ送ることのできるバッファを持つルータを作成した（図 3）．システムはバッファが溢れない限りはデッド

Table 1 Base Functions for Communication

void *SendPacket*(Channel *to, int my_num, int to_num, int type, void *ptr);
int *ReceivePacket*(Channel *from, Channel *to, int from_num, int type, void *ptr);

ロックしない仕組みになっている．

送信，受信の関数とも伝わる情報は冗長であるので，データタイプを問わない送受信，または送信元を問わない送受信等の様々なバリエーションの関数がある．またブロードキャストモードや，エラーメッセージを割り込みで送信する緊急モード等の通信データのレベルを実現した．

システムはトランスピュータのネットワークの上に構築されている．システムの開発において各プロセッサ間のデータのやりとりを簡略化するために，任意のデータをネットワークの任意のプロセッサへ送ることのできるバッファーを持つルーターを作成した．表 1 にユーザープログラム中から用いる通信用の基礎関数を示す．

4.2.2 ルータを用いた通信接続

図 3 にルータによるデータの通信の概要を示す．始めにユーザーは各プロセッサに独立な番号をつけておく．プロセッサ M のプロセスからプロセッサ N のプロサスへデータを送信するには，送る側のプロセスから以下のように送信する．送信されるデータは任意のデータ構造でいいが，typeof 関数で独立なデータ番号を返すようにしておく必要がある．ここで $E1, E2, E3$ はルータとユーザープロセス間のエッジである．

```
struct { .... } DATA;
/* initialize DATA */
SendPacket(E1, M, N, typeof(DATA), &DATA);
```

データは送信元のプロセッサ番号，送信先のプロセッサ番号，データのタイプと共にパケットとして目的のプロセッサのバッファーまで送られる．ルーターは目的のプロセッサまでパケットを転送し，目的のプロセッサではそれをバッファーに格納することにより，データを受けとるプロセスが任意の時刻に受信できるようにする．ここでルータープロセス，バッファープロセスも並列プロセスで実行されている．受信するプロセスは任意の時刻にバッファーに対して送信元のプロセッサから来た目的のデータタイプのデータの要求を出す．

```
struct { .... } DATA;
ReceivePacket(E2, E3, M, typeof(DATA), &DATA);
```

ルータの顕著な効果として，トランスピュータネットワークの欠点である，Host につながっていないネットワークの奥のトランスピュータからでも fprintf, fscanf 等の入出力が簡単にできることが上げられる．これにより，開発効率は大きく上がった．

4.2.3 ウインドウを基本とする並列視覚処理ライブラリの構成

並列視覚処理は矩形領域の局所処理を行なうウインドウを基本とする．この視覚処理ライブラリを構成するために，縦型のオブジェクト階層では拡張性が得られないことから，横型のオブジェクト階層を作成する．

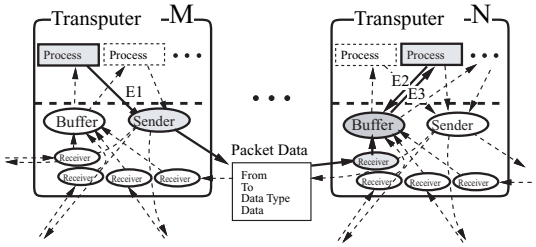

Fig. 3 Components of Router

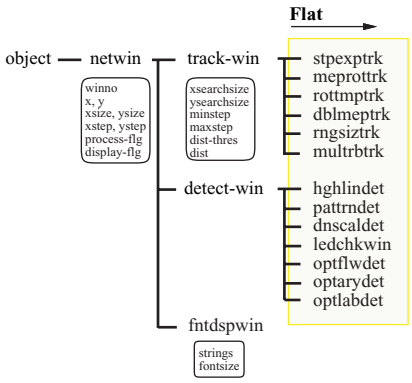

Fig. 4 Object Hierarchy for Netwin Library

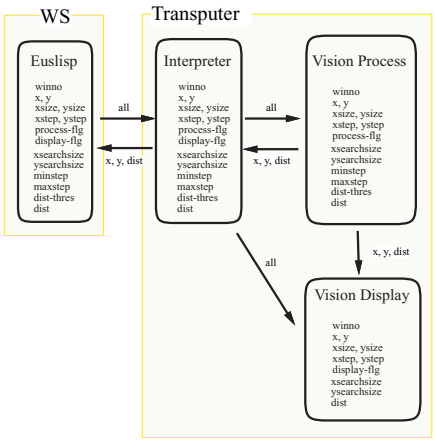

Fig. 5 Window Object Component of STPEXPTRK

基本的なウインドウのクラスと，その子クラスとして良く行なう処理に応じて追跡や検出，表示のフラグのためのスロットだけを持つクラスを用意し，各ウインドウオブジェクト [14, 15] はそれらの下に，並列に定義する (図 4)．これにより各ウインドウオブジェクトは，お互いには関係をもたず，それぞれのウインドウの開発，更新，保守をやり易くする．

4.2.4 並列処理マネージャでの制御構造枠組の実現

視覚機能のためのウインドウ（の一部）は，機能的な理由から複数のプロセッサ上に存在する可能性が大きいと考えられる．これは一つのウインドウが複数のプロセッサの機能を利用するためである．本システムでは４つのプロセッサ上に存在する．

本システムでは，(1) 画像処理をするプロセッサ，(2) 表示を

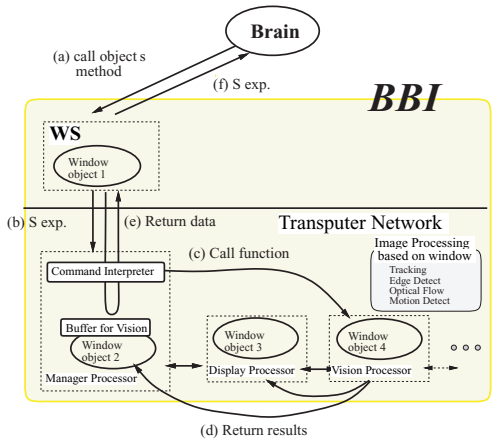

Fig. 6 Window Object Utilization

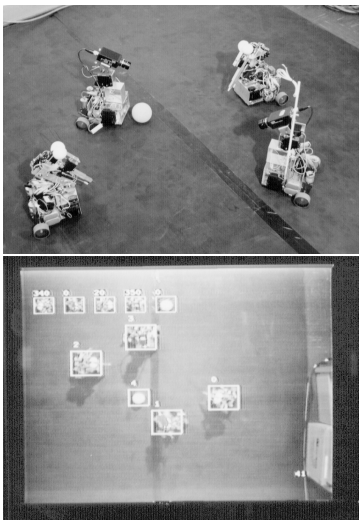

Fig. 7 Soccer Experiment

するプロセッサ，(3) インタプリタの存在するプロセッサ，(4) 脳部が用いるＷＳ上のオブジェクト，のそれぞれの上にウインドウの実体が分散して存在する．これは一つのウインドウが複数のプロセッサの機能を利用するため，また結果を実時間で返すために (3) でバッファリングしているためである．

これらの間で適切な制御，データの更新を行なうためのライブラリは上部（ＷＳ上に存在するオブジェクト）からパラメータやフラグを用いて処理を制御し，画像処理プロセッサ上のオブジェクト（の一部）は処理の結果をトップへ返す機能が必要である．

しかし，(1) プロセッサ間の通信を減らすために，(2) スロットによりデータを更新される方向があるために，オブジェクト全体のデータを通信してコピーし合うのは効率が悪い．

そこで，各プロセッサ上のウインドウオブジェクトは通信用の構造体と構造体の送受信のメソッドを持ち，必要な通信を最小限の通信量で行なうように設計する．図 5 にテンプレートマッチングによるトラッキングウインドウ (STPEXPTRK) を例に各プロセッサ上のオブジェクトのスロット変数を示す．

4.3 *Brain* 部の視覚処理機能インターフェース

上位のソフトウェアからは視覚機能は矩形領域を基本とする局所処理を単位とし，これをウインドウオブジェクトと呼ぶ．開発した視覚機能オブジェクトのライブラリを Netwin (Network Window Library) と名付けた．システムの処理の流れを 図 6 に示す．

上位のソフトウェア（図中の Brain）は Netwin のオブジェクトのインスタンスを生成し，必要なメソッドにアクセスする (a) ことによりその機能を呼び出す．呼び出されたメソッドは下位の視覚処理プロセッサ群の制御を行なっているマネージャプロセッサと通信し，S 式の形で機能を呼び出す (b)．マネージャプロセッサでは Command Interpreter が S 式を解釈し，視覚処理プロセッサの機能を呼び出す (c)．視覚処理プロセッサ上で行なわれた処理の結果は常時表示用のプロセッサとマネージャプロセッサに渡されるバッファリングされる (d)．処理の結果の要求が来た際には，このバッファから結果が返される (e)．返された結果は S 式の形で上位のソフトウェアに返される (f)．

各ウインドウは処理の開始 (:init)，終了 (:destroy)，結果の要求 (:ask)，画像処理パラメータの変更 (:set)，画像処理の停止 (:stop)・再開 (:start)，およびその他（表示の変更 (:show, :hid) 等）のメソッドを持つ．

図 4 に上位のソフトウェアが用いるウインドウオブジェクトの階層を示す．枠中の記号は，各オブジェクトの持つ代表的なスロット変数を示している．ここでオブジェクトの第３階層が各ウインドウオブジェクトのクラスであるが，ここで示すように各ウインドウオブジェクトはそれぞれスロットやメソッドを独立に持っている．

5. マルチロボット環境としてのプラットフォームの利用形態

ソフトウエアプラットフォームを複数の研究者が利用できれば，開発者がそれぞれに開発してきたロボットとそれを動かすソフトウエアを持ちよってマルチロボットの実験を行なえることになる．これにより，マルチロボットによる協調や競合の実験を容易に行なえるという利点がある．このようなシステムを作成するためには，個々のロボットのソフトウエアを構築した後，そのままネットワーク上に存在する複数のブレインとの通信や，他のボディを用いる機能が利用できる環境が必要である．他のブレインとの通信はブレインライブラリにネットワーク通信の機能を導入することで行なえる．また他のボディを用いるためにはブレインソフトウエアのインターフェースの共有が重要である．

開発したソフトウエアプラットフォームを用いたマルチロボットアプリケーションの例として２台 vs ２台でサッカーを行なう分散型ロボットの例を示す．図 7 に実験の風景と上部から見ているカメラの処理画像を示す．

ソフトウエアの共有に関しては，ボディの３次元モデルの作成や，行動ライブラリの作成は (どちらも Brain Library) は開発者がそれぞれ個別に作成することができる．シミュレータ上でのブレインのソフトウエアの開発環境 (Mother Tool) の作成も同様に別の開発者が同時に作成することができる．また，視

覚機能に関しても回転トラッカーによりロボットの位置と姿勢を検出するウインドウの作成も開発者が個別に行なえる (Sensor&Actuator Library). このように個別に作成可能なライブラリを用いてロボットのソフトウエアを開発することが可能である.

センサの実時間処理と結果取得は Brain Library から Sensor&Actuator インターフェースを用いることにより, 以下のように行なえる.

```
5.jskeusx$ (setq win (instance meprottrk :init
:x 30 :y 200 :xstep 3 :ystep 3))
#[meprottrk #X24aad4]
6.jskeusx$ (send win :start)
7.jskeusx$ (send win :show)
8.jskeusx$ (setq result (send win :ask))
(ask-meprot 30 203 30 1 5245)
9.jskeusx$ (send win :set :x 100 :y 200)
```

マルチロボットの環境は通信機能 (Brain Library) を用いることにより, またシミュレータ環境 (Mother Tools) を実現することによりソフトウエア上で各ロボットのブレインのソフトウエアを作成する.

6. ま と め

実世界で行動する知能ロボットのソフトウエア開発のためには, ソフトウエアの環境を複数の開発者が共有でき, 環境そのものもロボットのソフトウエアに合わせて進化できるソフトウエアプラットフォームが重要である.

本論文では知能ロボットのソフトウエアの階層構造として *Mother, Brain, Sensor&Actuator* からなる構成を考え, 大規模ソフトウエア開発の観点から共有と拡張性のトレードオフの問題と, 実時間センサ処理の観点から並列センサ処理システムとその結果の実時間取得の問題が本質的に重要であることを指摘し, それぞれについての設計方針とリモートブレインロボットでの設計及び実現の方法を述べ, マルチロボットへのアプリケーションの例を述べた.

リモートブレインロボットは本システムを用いて, 実世界で行動する知能ロボットの研究を行なうと共に本システムそのものを進化させていく予定である. 今後の課題としては, (1) センサ処理結果からイベントドリブンに上位のソフトウエアを駆動できる機能, (2) 反射動作のような早いループのために, センサ (視覚) の入力から行動の出力のループを脳のプログラムからセンサ処理部に与えられるように, 制御構造を含んだインターフェースのプロトコル, (3) 様々な構成の脳から利用できるように, スループットやセンサ処理部の処理能力をダイナミックに変化させる (migration) 機能, といった点が挙げられる.

参 考 文 献

[1] Masayuki Inaba. Remote-Brained Robotics: Interfacing AI with Real World Behaviors. In *Robotics Research: The Sixth International Symposium*, pp. 335–344. International Foundation for Robotics Research, 1993.

[2] 小笠原司, 井上博允. 知能ロボット: プログラミングシステム cosmos. 日本ロボット学会誌, Vol. 2, No. 6, pp. 507–525, 1984.

[3] T. Lozano-Perez, et al. Handey: A Task-Level Robot System. In Hideo Hanafusa and Hirochika Inoue, editors, *Proceedings of 4th International Symposium on Robotics Research*, pp. 29–36. Published as Robotics Research 4, 1988.

[4] Charles E. Thorpe, editor. *Vision and Navigation : The Carnegie Mellon Navlab*. Kluwer Academic Publishers, 1990.

[5] Raja Chatila. Representation + reason + reaction → robot intelligence. In *Proceedings of 6th International Symposium on Robotics Research (ISRR6)*, pp. 387–397. MIT Press, 1993.

[6] S. Suzuki, J. Iijima, and S. Yuta. Design and Implementation of an Architecture of Autonomous Mobile Robots for Experimental Researches. In *Proceedings of '93 ICAR*, pp. 423–428, 1993.

[7] L. D. Horswill. Polly: A vision-based artificial agent. In *Proceedings of AAAI 1993*, pp. 824–829, 1993.

[8] Masayuki Inaba, Satoshi Kagami, Fumio Kanehiro, Koji Takeda, and Hirochika INOUE. Vision-Based Adaptive and Interactive Behaviors in Mechanical Animals using the Remote-Brained Approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 933–940, 1994.

[9] 加賀美聡, 稲葉雅幸, 井上博允. リモートブレインロボットの視覚システム : BBIV - I. 日本機械学会ロボティクスメカトロニクス講演会 '95 予稿集, pp. 1072–1075, 1995.

[10] 加賀美聡, 稲葉雅幸, 井上博允. リモートブレインロボットの事象駆動型並列視覚システム : BBIV-II. 第 13 回日本ロボット学会学術講演会予稿集, pp. 613–616, 1995.

[11] 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.

[12] 加賀美聡, 稲葉雅幸, 井上博允. Euslisp を用いたリモートブレインマルチロボット環境. 日本機械学会ロボティクスメカトロニクス講演会 '94 予稿集, pp. 225–228, 1994.

[13] 加賀美聡, 稲葉雅幸, 井上博允. 脳を持ち歩かないマルチロボットのための視覚と動作制御の並列処理システム. 日本機械学会ロボティクスメカトロニクス講演会 '93 予稿集, pp. 797–802, 1993.

[14] 森武俊, 稲葉雅幸, 井上博允. 複数注視トークンの生成・追跡システム. 日本機械学会ロボティクス・メカトロニクス講演会'94 講演論文集, pp. 209–212, 1994.

[15] 松本吉央, 桑原太地, 柴田智広, 稲葉雅幸, 井上博允. ハイパースクーターの研究：複数の注視領域の管理システムによる環境情報の獲得. 日本機械学会ロボティクス・メカトロニクス講演会'94 講演論文集, pp. 489–492, 1994.

加賀美 聡 (satoshi KAGAMI)

1970 年 3 月生まれ. 1992 年上智大学理工学部機械工学科卒業. 1994 年東京大学大学院工学系研究科情報工学専攻修士課程終了. 現在, 同大学院博士課程在学中. (日本ロボット学会学生会員)

稲葉 雅幸 (masayuki INABA)

1958 年 5 月生.1986 年東京大学大学院工学系研究科情報工学専門課程博士課程終了. 工学博士.1989 年東京大学工学部機械情報工学科助教授, 現在に至る. 日本機械学会, 情報処理学会, 計測自動制御学会, 人工知能学会. (日本ロボット学会正会員)

井上 博允 (hirochika INOUE)

1942 年 7 月生.1965 年東京大学工学部産業機械工学科卒業. 1970 年同大学院博士課程終了. 工学博士. 同年電子技術総合研究所入所.1977 年東京大学工学部機械工学科助教授.1984 年教授. 現在, 機械情報工学科教授. ロボット全般, 人工知能, 情報システム工学の研究と教育に従事. (日本ロボット学会正会員)