〔解 説〕

ロボットソフトウェアのための データの表現と利用*

小 笠 原 司

1. はじめに

ロボットシステムは、多種類のソフトウェアを統合したトータルシステムである。各サブシステムは目的に応じて異なる機能、処理形態を有し、記述言語や取扱うデータの構造が異なるものになってしまうのが実情である。したがって、そのインプリメントには制御から人工知能まで幅広い分野の知識が必要となり、実際に全ての機能が統合され、しかもそれらが同時に動き、協調しながら作業を行うロボットシステムの報告は、対象を簡単な積木の世界に限ったものでもほとんど見あたらない。

Stanford 大学では、ロボット言語 AL¹⁾を中心としたプログラミングシステムが研究されている。ティーチング²⁾、デバックシステムまで含んだ優れたシステムであるが、視覚機能、問題解決機能が不十分である。視覚システムには、別に研究されている ACRONYM³⁾を用いるものと思われるが、AL との結合はまだできていない。

IBM では、ロボット言語 AML⁴⁾を中心にシステムを統合しようとしている。直交座標型ロボット RS-1 を使用するものである。AML はインタプリタ形式であり、ユーザが関数を定義してゆくことによって、独自のロボットシステムを構築できるのが特徴である。視覚機能についても、関数を定義して使っていこうという試みが報告されている⁵⁾。しかし処理は二次元に限られている。また問題解決機能については今後の課題である。

東大で開発された COSMOS システム^{6,7)}は、Lisp をベースにしたシステムであり、ミニコンピュータを中心とした分散型システムに、感覚、行動、思考機能、および人間との対話機能を統合したものとして特筆すべきである。このシステムの特徴は、FRL⁸⁾を用いた環境モデル管理システムと、Lisp をベースにしたことによるシステムの拡張性にある。問題解決機能については現在開発中である⁹⁾。

本稿では、これらのシステムを例にして、ロボットソフトウェアでデータがいかに表現され、またいかに利用されているかについて述べる。

2. ロボット用ソフトウェアの体系

本章では、ロボットシステム¹⁰⁾の構成について、COSMOS を例にして解説する。

COSMOS は東京大学工学部機械工学科で開発されている、研究用の知能ロボット実験システムであり、作業としては、機械装置の分解、修理、組立てを想定している。その基本方針は次の2点である。

(1) 感覚機能、行動機能、思考機能、人間との対話手段などを一つのシステムとして統合し、操作しやすい知能ロボットのプロトタイプを構築すること。

(2) このシステムをツールとして種々の研究を行い、それらの成果をシステム内に使用可能なソフトウェアとして蓄積することにより、知能ロボットの機能をより高度なものへと育てていくこと。

これらに従い、システムは拡張性、モジュール化を重視した設計がなされている。その特徴は以下の通りである。

(1) 複数の計算機を階層構造に結合した分散型システムである。

(2) 各プロセッサは GPIB で結合され、ハードウェアの拡張性、モジュール化を図っている。

(3) 上位のソフトウェアは、人工知能の手法の導入が容易であり、また対話形式の処理に適した Lisp¹¹⁾を用いて記述されている。

(4) サブシステムごとにプロセスを生成し、マルチプロセス方式のインプリメントを行い、ソフトウェアのモジュール化を図っている。通信はメッセージ交換で行う。

(5) 入力装置はターミナル、タブレット、音声入力装置など、互いに置き換えが自由に行えるよう、仮想端末の考え方で取扱っている。

システムのハードウェアの構成を図1に示す。

COSMOS ソフトウェアは、次の7つのサブシステム

原稿受付 1984 年 1 月 18 日

* Data Structure for Robot Software

図 1 COSMOS のハードウェア構成

から構成されている。

- (1) トップレベル
- (2) ロボット言語 AL/L¹²⁾
- (3) アームコントロールシステム
- (4) アームシミュレータ
- (5) 視覚システム
- (6) アームコントローラ
- (7) センサプロセッサ

(1) から (5) まではミニコン上で走り、(6) と (7) はそれぞれ別のマイコン上で走る。上位のソフトウェアは Lisp で書かれており、サブシステムごとに別の Lisp インタプリタが走り、サブシステム間はコマンドの受け渡しで通信している。下位の数値計算を多用する部分は Pascal を用いている。COSMOS のソフトウェア体系は図 2 のようにまとめられる。

COSMOS はロボット言語をベースとしたプログラミングシステムである。ユーザの指令は、各種入力装置を通じてシステムに与えられ、ロボット言語のソースプログラムに変換される。交換されたプログラムは、言語処理系によりプリミティブレベルの動作コマンドに展開され、各要素器官に転送されてロボットが動作する。この間のデータの流れは図 3 のようになる。

3. ロボット言語におけるデータ構造

3.1 ロボット言語の種類

ロボット言語は一般のプログラミング言語と異なり、ロボットの置かれている環境を正確に記述しなければなら

ない。そのためには、三次元形状を効率良く記述可能なデータ構造が必要となる。

ロボット言語の開発には、

- (1) 専用のロボット言語を新規に設計する。
- (2) 汎用のプログラミング言語にロボット制御用の機能を付加する。

の二つの方法がある。環境や動作の記述には、

(2) よりも (1) の方が有利である。汎用言語では記述能力や処理の効率が悪くなるのは否めない。(1) の代表的なものに Stanford 大学の AL, Edinburg 大学の RAPT¹³⁾, IBM の AML 等がある。しかし、最近では、CLU をはじめとする抽象データ型言語¹⁴⁾の研究により、プログラマが独自のデータ構造とその処理手続きを定義することができるようになった。このような言語を用いれば、(2) の方法でも効率のよい記述が可能である。

Purdue 大学の Paul らは、Pascal の record 宣言を用いてデータ構造を定義し、ロボットのプログラミングに用いた¹⁵⁾。現在はこれを C 言語¹⁶⁾におきかえて使用している¹⁷⁾。また MIT の MINI¹⁸⁾ や東大の AL/L は、Lisp にロボット言語の機能を埋め込んだものである。さらに、電総研でも Lisp を用いてロボット言語が開発されている¹⁹⁾。この方法は開発時間が短く、またユーザが新しく言語を学ぶ必要がなく、さらに他で開発されたソフトウェアの移植も容易であるといった利点があり、実験システム用には適している。

3.2 AL のデータ構造

AL は SCALAR, VECTOR, ROTATION, TRANS, FRAME の 5 種類のデータ構造を持っている。AL は対象物やハンドを指定する場合に、それらを代表する座標系名で表す。これが FRAME 型データである。座標系間の変換を表すのが TRANS 型データである。これらは平行移動量を表す VECTOR と、回転を表す ROTATION 型のデータより構成される。ROTATION は 3×3 の行列で表現されるが、回転軸のベクトルと回転量から計算される。データ間の演算については各種の関数が用意されている。また頻繁に用いる XHAT, YHAT, ZHAT (各軸の単位ベクトル) などの定数はマクロにより定義されている。

3.3 PAL²⁰⁾ のデータ構造

PAL は同次座標表現に基づいたデータ構造を用いている (図 4)。PAL では物体やハンドの位置を表す変換行列を用いた式で動作を指示する。たとえば、BOX をつかむ位置にハンドがあるときには、

$$Z+T6=BOX+GRA$$

なる PAL 式が成り立つ。ここで、Z は絶対座標からロボットの基準座標系への変換行列、T6 はロボットの基準座標系からハンドの基準座標系への変換行列である。また、+ は変換行列の積をとることを表している。そしてマクロ機能を用いて、 $ARM := -GRA + Z + T6$ と置きかえ、更に、 $ARM =$ を MOV と置きかえると、

MOV BOX

と書くことができる。これが PAL の動作プログラムである。環境が全て変換行列で表されるため、データの管理が非常に容易になる。

3.4 AML のデータ構造

IBM で開発された AML は、広範なレベルのユーザを対象としたロボット言語である。インタプリタ形式であり、関数を定義してゆくことにより、ユーザが独自のロボットシステムを形成してゆくことが可能になっている。AML には aggregate と呼ばれるリスト構造を表現するためのデータ型がある。aggregate の要素はスカラーあるいは aggregate であり、〈と〉で囲って、
 〈'BOLT', 10, 〈145.0, 73.5, 35.5〉〉
 のように書く。

aggregate に関してはさまざまな演算が用意されてい

図 2 COSMOS のソフトウェア体系

図3 ロボットシステム内でのデータの流れ

る。aggregate のコンカチネートは、# を用いて $\langle 1, 2 \rangle \# \langle 3, 4 \rangle$ のように記述する。aggregate 型変数への代入は、要素の長さが同じでない場合には必要となるまでが代入され、またスカラー量を代入すると全ての要素に代入される。さらに、代入文の左辺に参照式 (reference expression) を要素とする aggregate を書くことにより、並列に代入を行うことができ、x と y の値の交換は、

$$\langle \&x, \&y \rangle = \langle y, x \rangle$$

でよい。さらに aggregate 同志の演算は、対応する要素ごとに行われる。例えば、以下の様である。

式	値
$\langle 10, 20 \rangle + \langle 3, 6 \rangle$	$\langle 13, 26 \rangle$
$\langle 1, \langle 2, 3 \rangle \rangle * \langle \langle 4, 5 \rangle, 6 \rangle$	$\langle \langle 4, 5 \rangle, \langle 12, 18 \rangle \rangle$
'S' GT $\langle \text{'S'}, \text{'T'}, \text{'Q'}, \text{'P'} \rangle$	$\langle 0, 0, -1, -1 \rangle$

このように、aggregate は Lisp のリスト構造と非常によく似たものであることがわかる。

AML では、目標値は各関節ごとの値を与える形式になっている。例えば、関節 4 を 30° 、5 を 90° にしたければ、

MOVE $\langle \langle 4, 5 \rangle, \langle 30, 90 \rangle \rangle$;

と書く。直角座標型マニピュレータの場合は問題にはならないが、作業座標系で目標値を指示する場合には、アームソリューションのプログラムを書かなければならない。工業用にはこのレベルで十分なのだろうが、知能ロボットのためのロボット言語としてはものたりない。言語処理系内で組込みにするか、もっと下のレベルで処理すべきことと考える。

図4 同次座標表現

AML は Lisp の影響を非常に強く受けたロボット言語である。必要最低限のものを用意しておき、拡張はユーザーに任されている。幅広いレベルのユーザーを対象にした言語としては、この方法が優れている。しかし、動作の構文などは他の高水準ロボット言語と比較すると見劣りがするのは否めない。ただ、AML はあくまでも基本システムである。これを使った高度のロボットシステムは早晩出現するであろうし、AML はそれに十分な機能を備えた言語である。

4. 環境モデルにおけるデータ構造

4.1 環境モデルの機能

ユーザーがロボット言語を介して与えた指令は、ロボットシステム内に入力され、内部処理に適した形式に変換される。ロボット言語処理系によって、高レベルの動作記述がプリミティブレベルの動作コマンド列に展開される。この処理を行うためには、処理系は内部に環境モデルを持ち、動作をシミュレートして、時々刻々変化する物体の位置関係を正確に把握していなければならない。ロボット言語のレベルが高くなるほど、環境モデルに収納すべきデータの種類の増加し、その管理システムは複雑なものになる。環境モデル管理システム内では、そのレベルに応じ、次のような処理を行う。

(1) 座標系型データの管理

物体の基準座標系に名前を付け、その値を記録する。さらに動作によって物体が移動した場合には、それらの値を自動的に変更する。

(2) 物体間の結合関係の管理

物体の結合関係 (semantic affixment) を表すために、物体間をリンクで接続し、結合関係のネットワークを作り上げる。このリンクには、結合関係、相対位置等を記入しておく。動作ごとに物体の位置、他の物体との関係がわかるようにする。AL では affix, unfix 文を用いてこの管理を行っている。

(3) 物体に関する情報の管理

物体間の関係はネットワークで表現されるが、各ノードが物体に相当し、そこに物体に関する情報を記録する。例えば物体の形状、大きさ、重さ、色、把握点、近接点、等である。これらの情報は、動作の決定や問題解決の際の知識として利用される。

(4) 作業に関する情報の管理

ユーザ定義の手続きや物体の組立て方法など、作業に関する知識を環境モデル内に記述しておく。作業レベルのロボット言語の実現に必要であり、これによって動作の展開を環境モデル管理システム内で行うことができる。

4.2 COSMOS の環境モデルの構成

COSMOS では、FRL (Frame Representation Language) を用いて環境モデル処理システムをインプリメントしている。物体間の結合関係や、物体に関するデータを表現するにはフレーム構造²¹⁾が都合がよい。

環境モデル内には、図5に示すような概念フレームが存在する。各概念に関する操作は、FRL の副作用 (\$IF-ADDED, \$IF-REMOVED 等のファセット) を用いて記述されている。THING は、フレームの木構造のルートに当たるフレームであり、全てのフレームから継承メカニズム (inheritance) によってアクセスできる。THING フレームには、PROTOTYPE スロットに対する副作用の処理、および拘束条件が図6のように記述されている。PROTOTYPE スロットは、実際のフレームの分類に用いている。SCALAR, VECTOR といったデータタイプの場合には、APVAL スロットに値が代入され、型のチェックを行う関数が \$REQUIRE ファセットに記述されている。COORDINATES 型フレームには、値を計算するための手続きが記述されている。

LINK フレームで結合関係を実現している。LINK 型フレームの PARENT スロットに COORDINATES 型の親フレームを記述し、CHILD スロットに COORDINATES 型の子フレームを記述する。HOW-AFFIXED スロットで結合の種類を表し、TRANSFORM スロットで位置関係を表す。また、COORDINATES 型フレームから LINK 型フレームをアクセスするために、PARENT-LINK と CHILD-LINK の二つのスロットを追加する。このため図7のようなフレームを定義している。例えばマニピュレータが箱を持っている場合には、HAND 1, BOX, LINK-OF-BOX-TO-HAND 1 の三つのフレームが作られる。LINK-OF-BOX-TO-HAND 1 は、HAND 1 と BOX の結合関係を記述しているフレームであり、それらは図8に示すように実現されている。

COSMOS では、この他にも把握点、近接点に関する

図5 COSMOS の概念フレーム

図6 THING フレーム

情報が継承メカニズム、付加手続き (attached procedure) といった FRL の特徴を生かして記述されている。

5. 軌道の実現

言語処理系では、高水準の記述をプリミティブな動作コマンド列に展開する。たとえば、

(MOVE BOX 1 TOP-OF-BOX 2)

という記述が、

move-arm[A],[B],[C],[D],[E]

といった具体的な目標点の列に置きかえられる。ここで [A]-[E] は図9における A-E 点を表す変換行列である。このプリミティブ動作から、実際にロボットが動くまでには各種の処理が施され、データはその形式を変化させてゆく。主な処理は図3にも示したように、アームソリューションの計算、軌道の計画、そしてサーボループである。

ロボット制御システムでは、実際の制御はマニピュレータの各関節ごとに目標値が与えられ、それに対してソフトウェアあるいはハードウェアによってサーボループが形成される。一方プリミティブ動作は、直交座標系における点列で表されているため、それを実現するための各関節角の目標値を求めなければならない。この処理が

注1) Minsky の提唱したフレームであり、座標系の意味ではない。

```

COORDINATES
(PARENT_LINK
($IF-ADDED
  ((PUT_VAL :VALUE 'PROTOTYPE 'LINK))
  ((PUT_VAL :VALUE 'CHILD :FRAME)))
($IF-REMOVED
  ((REMOVE_VAL :VALUE 'CHILD :FRAME))))
(CHILD_LINK
($IF-ADDED
  ((PUT_VAL :VALUE 'PROTOTYPE 'LINK))
  ((PUT_VAL :VALUE 'PARENT :FRAME)))
($IF-REMOVED
  ((REMOVE_VAL :VALUE 'PARENT '$VALUE))))

LINK
(PARENT ($IF-ADDED
  ((PUT_VAL :VALUE 'CHILD_LINK :FRAME)))
($IF-REMOVED
  ((REMOVE_VAL :VALUE 'CHILD_LINK :FRAME)))
  ((REMOVE_VAL :VALUE 'CHILD '$VALUE))))
(CHILD ($IF-ADDED
  ((PUT_VAL :VALUE 'PARENT_LINK :FRAME)))
($IF-REMOVED
  ((REMOVE_VAL :VALUE 'PARENT_LINK :FRAME)))
  ((REMOVE_VAL :VALUE 'PARENT '$VALUE))))
(HOW_AFFIXED ($DEFAULT (NONRIGID)))

```

図 7 結合関係を実現するフレーム

アームソリューションの計算である。

またロボットを制御するためには、サーボサイクルごとの目標値を計算しなければならない。プリミティブ動作で与えられる中継点からこの目標値列を生成するのが軌道の計画である。

軌道計算法としては、Paul の多項式近似²²⁾が有名である。彼は軌道の中継点によって区間に分け、各区間ごとに多項式で表現することを考えた。そして、なめらかな動作をさせるために、区間の変わり目で位置、速度、加速度が連続であること、および動作開始・終了点では速度、加速度が0であるという境界条件を与えると、両端の区間では4次式、途中では3次式になる。各多項式の係数は、区間の数を m とすると、 $3m-2$ 元の連立方程式を解くことにより得られる。各関節ごとにこの計算を行うと軌道が生成される。この方法の欠点は、計算量が膨大であることである。そのため、小型計算機では実現しずらく敬遠されてきた。また動きはなめらかであるが、作業空間上で直線や円弧といった幾何学曲線に沿って動かすことが難しい。

産業用ロボットの場合は、少々動きは悪くても計算量の少ない方が好まれる。実際には2点を指定し、その間を補間して軌道を生成するPTP制御を用いることが多い。Paulは、この場合に有効な軌道計画法として次のようなものを挙げている¹⁵⁾。

図 9 箱の移動

```

HAND1
(PROTOTYPE ($VALUE (HAND)))
(AKO ($VALUE (HAND)))
(CHILD-LINK ($VALUE (LINK-OF-BOX-TO-HAND1)))

LINK-OF-BOX-TO-HAND1
(PROTOTYPE ($VALUE (LINK)))
(AKO ($VALUE (LINK)))
(PARENT ($VALUE (HAND1)))
(CHILD ($VALUE (BOX)))
(TRANSFORM ($VALUE ((TRANS (0 0 1 10)
                             (0 1 0 20)
                             (-1 0 0 15)
                             (0 0 0 1)))))

(HOW-AFFIXED ($VALUE (RIGID)))

BOX
(PROTOTYPE ($VALUE (PART)))
(AKO ($VALUE (COORDINATES)))
(PARENT-LINK ($VALUE (LINK-OF-BOX-TO-HAND1)))

```

図 8 結合関係を表すフレームの例

図 10 で A, B, C, D は与えられた目標値、 τ はマニピュレータを停止状態から最大速度に加速するのに要する時間とする。 $\tau < t < T_1 - \tau$ では直線軌道に沿って動かし、 $-\tau < t < \tau$ では多項式に沿って動かすことにすると、境界条件(加速度まで連続)より、

$-\tau < t < \tau$ のとき、

$$q = \left[\left(\Delta C \frac{\tau}{T_1} + \Delta B \right) (2-h) h^2 - 2 \Delta B \right] h + B + \Delta B \quad (1)$$

$$\dot{q} = \left[\left(\Delta C \frac{\tau}{T_1} + \Delta B \right) (1.5-h) 2h^2 - \Delta B \right] \frac{1}{\tau} \quad (2)$$

$$\ddot{q} = \left[\left(\Delta C \frac{\tau}{T_1} + \Delta B \right) (1-h) \frac{3h}{\tau^2} \right] \quad (3)$$

ただし、 $\Delta C = C - B$, $\Delta B = A - B$, $h = \frac{t + \tau}{2\tau}$

$\tau < t < T_1 - \tau$ のとき、

$$q = \Delta C \frac{t}{T_1} + B \quad (4)$$

$$\dot{q} = \frac{\Delta C}{T_1} \quad (5)$$

$$\ddot{q} = 0 \quad (6)$$

となる。この計算を各自由度について行えばよい。先程の多項式近似に比べると、動きは少々ぎこちなくなるが、計算量ははるかに少なくなる。この方法の欠点は、指定された中継点を通らないことである。その点を通るためには、同じ目標値を2度出力する必要がある。この方法は計算量が少ないため、各関節ごとではなく、作業座標系でサーボを行うことを提唱している。

作業座標系でサーボを行うと、アームソリューションの計算が不用になる²²⁾、物体にまっすぐ接近するといった作業空間に基づいた動きが可能になる、といった長所がある。しかし計算が複雑なため、これまでは制御系への組み込みは困難であった。しかし計算機技術の向上に

注 2) ただし、サーボループ内で関節角から作業座標系への変換が必要となる。

より、最近ではサーボサイクルで座標変換処理とダイナミクス計算を行うことが可能になった。電総研では、高速のマイコン MIPROC を用いて、7 自由度のサーボを 2.5msec の間隔で実現している²³⁾。豊田中研では、インテル 8086, 8087 を用いてサーボサイクル 30msec で実現した²⁴⁾。この方式のサーボ系は今後ますます取入れられるものと考えられる。

6. おわりに

以上、ロボット言語、環境モデル、および軌道計画について、データの表現とその処理について述べてきた。

ロボットに関するソフトウェアの体系は、まだまだ確立されておらず課題は多い。第 1 に、構成要素ごとのデータの処理と表現方法の検討が必要である。特にセンサからのデータの取扱いについては、「何番目の入力ラインの値が幾つになれば……」といった記述が多く、もっと概念的なプログラミングを行う方向に進むべきである。

そして第 2 に、システム化の技術を確立しなければならない。要素ごとの技術は確立しても、それを結び付ける部分、それらを統括する部分の機能が優れていなければ、ロボットは十分な能力を発揮できない。ロボット用のネットワーク技術の研究、OS の開発といったものが次世代のロボット技術のキーポイントになるであろう。

参考文献

- 1) S. Mujtaba and R. Goldman, "AL Users' Manual", Stanford University AI Lab., 1979
- 2) R. Brooks, "Symbolic Reasoning Among 3-D Models and 2-D Images", Artificial Intelligence, Vol.17, pp.285-348, 1981
- 3) D. Grossman and R. Taylor, "Interactive Generation of Object Models with a Manipulator", IEEE Trans. on SMC, Vol. SMC-8, No.9, pp.667-679, 1978
- 4) R. Taylor, P. Summers, and J. Meyer, "AML: A Manufacturing Language", Robotics Research, Vol. 1, No.3, pp.19-41, 1982
- 5) M. Lavin and L. Lieberman, "AML/V: An Industrial Machine Vision Programming System", Robotics Research, Vol.1, No.3, pp.42-56, 1982
- 6) 井上, 小笠原, "知能ロボットのシステム構成", 昭和57年電気四学会連合大会, 1982
- 7) H. Inoue and M. Inaba, "Hand Eye Coordination in Rope Handling", Proc. of ISRR, 1983
- 8) R. Roberts and I. Goldstein, "The FRL Manual", MIT AI Memo 409, 1977
- 9) 岡野, 井上, "対象物レベルのロボット言語の研究", 第1回日本ロボット学会学術講演会予稿集, pp.39-40, 1983
- 10) T. Lozano-Perez, "Robot Programming", Proceedings of the IEEE, Vol.71, No.7, pp.821-841, 1983

図 10 直線補間に基づく軌道の実現¹⁵⁾

- 11) P. Winston and B. Horn, "Lisp", Addison-Wesley, 1981
- 12) H. Inoue, et al., "Design and Implementation of High Level Robot Language", Proc. of 11th ISIR, pp.675-682, 1981
- 13) R. Popplestone, "RAPT: A Language for Describing assemblies", Industrial Robot, pp.131-137, 1978
- 14) 佐渡, 米澤, "抽象データ型言語", 情報処理, Vol.22, No.6, pp.525-530, 1981
- 15) R. Paul, "Robot Manipulator", The MIT Press, 1981
- 16) B. Kernighan and D. Ritchie, "The C Programming Language", Prentice-Hall, 1978
- 17) V. Hayward and R. Paul, "Robot Manipulator Control Under UNIX", Proc. of 13th ISIR, Vol.2, 1983
- 18) H. Inoue, "Force Feedback in Presice Assembly Tasks", MIT AI Memo 308, 1974
- 19) 末広, "作業座標系内動作記述に基づくロボット言語", 第 22 回 SICE 学術講演会予稿集, pp.103-104, 1983
- 20) K. Takase, R. Paul and E. Berg, "A Structured Approach to Robot Programming and Teaching", IEEE Trans. on SMC, Vol. SMC-11, No. 4, pp. 274-289, 1981
- 21) M. Minsky, "A Framework for Representing Knowledge", in P. Winston ed., "The Psychology of Computer Vision", McGraw-Hill, 1975
- 22) R. Paul, "Modelling, Trajectory Calculation, and Servoing of a Computer Controlled Arm", Stanford Univ. AI Lab., Rep. AIM 177, 1972
- 23) 川島, 末広, 高瀬, "作業座標軸サーボ系のインプリメンテーション", 日本ロボット学会誌, Vol.1, No.2, pp.72-74, 1983
- 24) 森部, 他, "実時間軌跡制御方式", 第1回日本ロボット学会学術講演会予稿集, pp.165-166, 1983

小笠原 司

(Tsukasa OGASAWARA)

昭和 30 年 10 月 16 日生れ。昭和 53 年 3 月東京大学工学部計数工学科卒業。昭和 58 年 3 月同大学院工学系研究科情報工学専門課程博士課程修了。同年 4 月電子技術総合研究所入所。知能ロボットの研究に従事。

計測自動制御学会, 情報処理学会の会員。工学博士。

(日本ロボット学会正会員)