関数型言語の特徴をもつ動作記述型ロボット言語†

荒牧重登†† 長澤勲†††

現在，生産現場におけるロボット言語のプログラミングのほとんどは，職業的プログラマでない作業者がロボットを操作しながらプログラムを作成するオンラインプログラミングである．したがって，ロボット言語が対話性に優れていることが必要である．そこで本研究では関数型言語がもつ会話型プログラム作成の高生産性を導入したロボット言語 IML を提案する．また，複雑な軌道を描く場合のようにプログラム記述による方式だけではその作業プログラムを作成することがむずかしい場合がある．このため，IML では教示データを言語の中に組み込んで利用できる機能がある．本論文では IML の概要を述べ，いくつかの作業例を IML で記述し，実際にロボットに作業を行わせることによってその有効性を示す．

1. まえがき

ロボット（本論文では人工の手を指す）に作業を行わせる方式を大別すると，プレイバック方式とプログラム記述による方式の二つがある．プレイバック方式は，操作者がロボットを操縦して作業経路や作業点を適当な記憶装置に記憶して（以下，教示と呼ぶ），作業時にそのデータを再生させる方式である．この方式では教示したときと同じ環境内での同じ位置でないとその作業を再生できないので柔軟性に欠ける．一方，プログラム記述による方式は，計算機を用いてプログラムによって作業を行わせる方式である．この方式は，前者に比べて柔軟性があり，最近では産業用ロボットに普及しつつある．作業プログラムを作成する場合，一般のプログラミング言語を用いるのではなく，ロボット専用の言語を開発することが必要となる．ロボット言語は従来から多くの研究がなされてきており[1)~4)]，VAL[5)]のように実用化されているものもある．ロボット言語は，一般のプログラミング言語と同様に構造の簡明さ，概念の統一性，機能の拡張性などが要求される[6)]．また，実際の現場では作業者がロボットを操作しながらプログラムを作成する場合が多いので，言語が対話性に優れていること，初心者にとってその言語が学びやすいことも重要である．現在実用化されている言語はこれらを十分満たしているとはいいがたい．たとえば，ループを組むことによって繰返しの制御を行っているが，これはプログラムの対話性を重視する立場から望ましくない．また，プログラム記述による方式だけでは複雑な軌道を描かせる場合のように，その作業プログラムを作成することが困難な場合がある．これらの問題点を解消するため本研究では関数型言語の性質を取り入れ，また教示したデータを利用できるロボット言語 IML (Interactive Manipulator Language) を提案する．ロボット言語を動作記述型，作業記述型，仕事記述型の三つに分類する[7)]と，IML は基本的には VAL 等と類似の動作記述型言語であるが，マクロ機能を有しているので作業記述型言語への拡張性を備えている．

本論文では IML の概要を述べ，いくつかの作業例を IML で記述し，実際にロボットに作業を行わせることによってその有効性を示す．

2. 設計方針

2.1 関数型プログラミングの導入

現在，生産現場におけるロボット言語のプログラミングのほとんどは，職業的プログラマでない作業者がロボットを操作しながらプログラムを作成するオンラインプログラミングである．したがって，ロボット言語を設計する場合，対話性に優れていること，システムの習得が容易であること，プログラムの構造が理解しやすいことなどが重要である．このため関数型言語がもつ会話型プログラム作成の高生産性を IML に導入する．たとえば，ある処理を繰り返したい場合（たとえばパレタイジング作業のように基本動作を順次繰り返して作業するような場合），従来のロボット言語ではループを用いる方法をとる．しかしこれはプログラムの生産性を低くする．IML ではループを用いないで LISP の MAP 関数や APL[8)]のオペレータのように関数型言語に見られる繰返し制御を導入する．

† The Motion Oriented Robot Language with the Characteristic of Functional Programming Language by SIGETO ARAMAKI (Educational Center for Information Processing, Kyushu University) and ISAO NAGASAWA (Computation Center, Kyushu University).

†† 九州大学情報処理教育センター
††† 九州大学中央計数施設

2.2 教示方式の採用

ロボットの作業点，作業経路あるいは対象物の位置・姿勢などの教示法は従来から多くの研究がある．いずれも生のデータを取り扱わず，各種の装置を使って会話的にデータを生成していく考え方である．IML では作業者がロボットを操作しながらこれらの値を教示できる機能を備えている．

2.3 教示データの利用

複雑な軌道を描く場合のようにプログラム記述による方式だけではその作業プログラムを作成することがむずかしい場合がある．このため，IML では教示データを言語の中に組み込んで利用できる機能がある．その場合，教示データは座標変換によって教示したときと異なる位置で再生される．

3. 言語の概要

IML における命令はコマンドで与えられ，インタプリタがそれを解釈実行する．コマンドはシステムが提供する基本コマンドとユーザが定義するコマンドとに分けられる．取り扱うデータ型はスカラ（整数または実数）と6個のスカラからなるベクトルである．このベクトルで三次元空間におけるロボットや作業対象物の位置・姿勢を表す．IML では三次元空間を直交座標系 (O-XYZ) で表し，位置を X, Y, Z，姿勢を ϕ (roll), θ(pitch), φ(yaw)（それぞれ，Z, Y, X 軸まわりの回転角）で表す．直交座標系はロボットに固定した絶対座標系と作業に応じて指定できる作業座標系とからなる．

3.1 基本コマンド

IML が提供する基本コマンドはロボットの動作に関するコマンドとプログラムの制御に関するコマンドとに分けられる．ロボットの動作に関するコマンドの一覧表を**表1**に示す．これらのコマンドは教示モード，作業実行モード，およびこれら両モードにおけるコマンドに分類できる．これらのモードは陽に指定する必要はない．ロボットの腕の動きは “MOVE” コマンドによって行われ，指の開閉は “OPEN”, “CLOSE” コマンドによって行う．これらのコマンドは，本来作業実行時に使用されるものであるが，IML では教示モードのときも使用でき，それぞれ HERE コマンド，WIDTH コマンドの機能も兼ね備えている．したがって，ロボットや作業対象物の位置・姿勢を教示しながら作業プログラムを作成することができる．この教示方法は，ロボットの実際の動作と対応させて教示できるので直感的である．TRAJ コマンドは，ロボットの腕や手先の軌道を教示するためのコマンドであり，再生するときにはその軌道名をコマンドとして入力すればよい．SPEED コマンドは，ロボットの手先の速度を指定するコマンドであり3段階 (H, M, L) 指定できる．IML は作業座標系における動作として記述することができるが，WITH コマンドは次の WITH コマンドが現われるまで現在の作業座標系が有効となる．WITH コマンドを省略すると絶対座標系を指定したものとみなす．プログラムの制御に関するコマンドの一覧表を**表2**に示す．表2の代入文で，v=exp の場合，演算子は加算(+)，減算(−)，乗算(*)，除算(/)のいずれかである．APL のようにこの演算子は被演算子としてスカラだけでなく，ベクトルにも適用できる．代入規則は単一代入とする．したがって，左辺の変数は代名詞のようなものであり，この変数名

表1 ロボットの動作に関するコマンド
Table 1 Commands for the motion of robot.

モード	コマンド名	オペランド	機能・意味
教示モード	HERE	位置変数	ロボットや作業対象物の位置・姿勢を教示する
	TRAJ	軌道名	軌道（位置・姿勢の集合）を教示する
	WIDTH	変数	指の開き幅を教示する
	COORD	座標系名	作業座標系を教示する
作業実行モード	軌道名	[座標系名]	TRAJ コマンドで教示した軌道を指定した座標系で再生する
教示および作業実行モード	OPEN	変数またはスカラ	指を指定した幅に開く
	CLOSE	同上	指を指定した幅に閉じる
	MOVE	位置変数	指定した位置にロボットを動かす
	WITH	作業座標系名	以後の作業座標系を指定した作業座標系とする
	SPEED	H, M, Lのいずれか指定	手先の動きの速度を指定する
	HOME	なし	ロボットを home position（原点位置）へ動かす

[ ] は省略可を示す

表 2 プログラムの制御に関するコマンド
Table 2 Commands to control program.

分類	コマンド名	形式	機能・意味
代入		v=a	スカラの代入
		v=(a_1, a_2, a_3, a_4, a_5, a_6)	ベクトルの代入
		v=exp	スカラまたはベクトルの演算結果の代入
繰返し	FOR	①FOR EACH 仮パラメータ／実パラメータ… コマンド名 仮パラメータ…	各仮パラメータに対する実パラメータの各値でコマンドの実行を繰り返す
		②FOR 仮パラメータ／実パラメータ コマンド名 仮パラメータ	実パラメータの各値でコマンドの実行を繰り返す．ここで，実パラメータの形式は (L, M, N) またはリストである．L, M, N はそれぞれ初期値，終値，増分である
コマンドの定義	DEFINE	DEFINE コマンド名 パラメータ… コマンド群 END	コマンドの定義
条件分岐	IF	IF 条件 THEN コマンド名 [ELSE コマンド名]	条件が真のとき THEN 部分のコマンドを実行，偽のとき ELSE 部分のコマンドを実行する
出口	EXIT	EXIT	繰返し処理からぬけ出るときまたはコマンド定義のなかから出るときに使う
	END	END	プログラムの終りを示す

[ ] は省略可を示す

をプログラム中の他の場所で再定義することはできない．

3.2 コマンド定義機能

基本コマンドだけでは作業プログラムの記述性に欠けるため，IML では関数型言語における関数定義のように，既存のコマンドを使って新しいコマンドを作成できる機能がある．このコマンド定義機能は一種のマクロ機能と考えることもでき，この機能を使うことによって作業プログラムを簡潔な記述に圧縮することができる．

いま例としてロボットがある位置に置かれているワークをつかんで持ち上げる作業（コマンド名を PICKUP とする）と，つかんでいるワークをある位置に置く作業（コマンド名を PUTDOWN とする）をそれぞれコマンド定義する例を**図1**に示す．

3.3 繰返し処理の記述

FOR コマンドは，コマンドを繰り返し実行するときに使用し，表2に示すように二つの使用法がある．①の形式は複数の仮パラメータに対応するそれぞれの実パラメータの各値でコマンドの実行を繰り返すものである．②の形式は一つの仮パラメータに対してコマンド

PICKUPの定義

```
INPUT COMMAND ? DEFINE PICKUP WORK L
                1. A=WORK+(0,0,50,0,0,0)
                2. MOVE A
                3. L1=L+30
                4. OPEN L1
                5. MOVE WORK
                6. CLOSE L
                7. MOVE A
                8. END
INPUT COMMAND ?
```

PUTDOWNの定義

```
INPUT COMMAND ? DEFINE PUTDOWN WORK L
                1. A=WORK+(0,0,50,0,0,0)
                2. MOVE A
                3. MOVE WORK
                4. L1=L+30
                5. OPEN L1
                6. MOVE A
                7. END
INPUT COMMAND ?
```

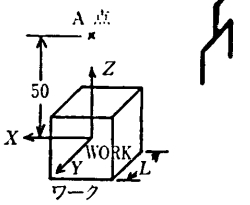
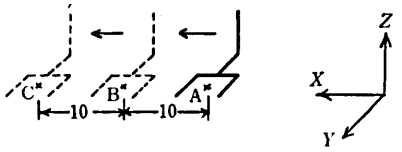


図 1 コマンド定義例
Fig. 1 Example of defining command.
WORK：ワークの掴点の位置と姿勢
L：ワークをつかむときの指の幅
A：approach 点*, departure 点*

* 作業対象物の近くで不用意にロボットを動かすと，他の対象物などと衝突することがあるので，対象物に接近したり離れたりする場合に，ロボットが必ず通過するようにした点のことである．



VALで記述した例

```
   SETI N=1
10 MOVE A
   SETI N=N+1
   IF N GT 3 THEN 20
   SHIFT A BY 10.0
   GO TO 10
20 ・・・
```

IMLで記述した例

```
DATA=FOR I/(1,3) A+((I-1)*10,0,0,0,0,0)
FOR I/DATA MOVE I
```

図 2 繰返し処理の記述の比較例
Fig. 2 Comparison of the program of iterative process.

の実行を繰り返す（FOR のネストも可能）.

パレタイジング作業や多層溶接作業のように多点の類似繰返し動作をさせる場合，従来のロボット言語ではループを用いる方法をとる．しかしこれはプログラムの作成と修正の効率から考えると望ましくない．したがって，IML ではコマンド言語の特徴を生かすために，APL にみられるようにループをなるべく用いないでプログラムを作成できるようにした．

たとえば，繰返し処理の記述例として，**図2**にA点からB，C点へロボットを動かすプログラムを VAL と IML で記述した例を示す．この図で IML の場合，まず A，B，C点の位置データを作成しこのリストの各要素で MOVE コマンドの実行を繰り返す．これは，VAL がループの中で使用されるデータをループの中で作成しているのに対して，IML ではデータを前もって作成しておきそのリストを引数として関数（コマンド）に作用させる考え方をとっている．図2からわかるように IML による記述のほうが，カウンタセットのための変数や条件分岐もなくプログラムが簡潔である．図2では MOVE コマンドの繰返し例であるが，複数のコマンドを繰り返して実行する場合には，それらのコマンド群をコマンド定義しておきそのコマンド名を指定すればよい．こうした繰返し処理の記述方法では，入力データをかえることによってプログラムの内容（コマンド定義の内容）を変えることなしに別の作業ができることがある．このことは，動作手順の記述部分とその入力データ部分とが分離しているといえる．そして，入力データを CAD から生成されるデータとすることもできよう．

3.4 教示データの作成と管理

IML は，ロボットを動かすことによって教示点を教え，そのときの腕の関節角からロボットあるいは作業対象物の位置・姿勢を計算する機能を備えている．しかし，パレタイジング作業のように規則的な位置にロボットを動かすような場合，いちいち教示することはめんどうである．このようなときには，代表点を教示した後図2の例のように FOR コマンドを使ってデータを作成すればよい．各作業座標系で教示したデータ（座標値）は**図3**に示すような階層構造で管理される．そして，作業実行時に教示したときと作業座標系が異なる場合，教示のときの作業座標系から作業実行時の作業座標系への座標変換を行って教示データを出力する．

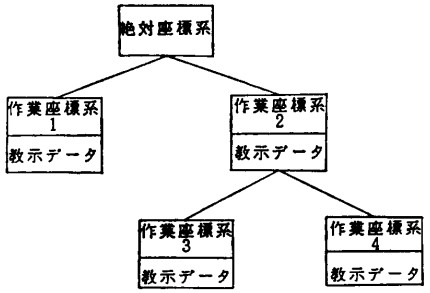


図 3 教示データの構造
Fig. 3 Structure of teaching data.

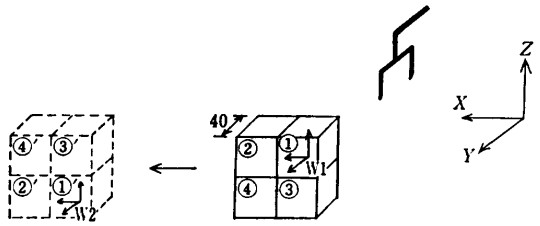


図 4 パレタイジング作業
Fig. 4 Palletizing.

```
INPUT COMMAND ? DEFINE CARRY LOCA1 LOCA2
                1. PICKUP LOCA1 40   } 図1で定義したコマンドを使う.   } CARRYコマンドの定義
                2. PUTDOWN LOCA2 40  }
                3. END
INPUT COMMAND ? HERE W1・・・ロボットの手先をW1の位置へ動かし,ワーク①の握点を教示する.
INPUT COMMAND ? DATA1=FOR I/(1,2) FOR J/(1,2) W1+((J-1)*40,0,-(I-1)*40,0,0,0)
                          ↑── ワーク①～④の位置データを作成する.
INPUT COMMAND ? HERE W2・・・ロボットの手先をW2の位置へ動かし,①′の位置を教示する.
INPUT COMMAND ? DATA2=FOR I/(1,2) FOR J/(1,2) W2+((J-1)*40,0,(I-1)*40,0,0,0)
                          ↑── ワーク①～④の移動先①′～④′の位置データを作成する.
INPUT COMMAND ? FOR EACH LOCA1/DATA1 LOCA2/DATA2 CARRY LOCA1 LOCA2
                        ↑── 作業を実行する
INPUT COMMAND ?
```

図 5　プログラム例
Fig. 5　Palletizing program.

4. 作業例と IML による記述

ここではいくつかの作業例をあげ IML による作業プログラム例を示す.

4.1 パレタイジング作業

この作業は，ある位置にあるワークを取り上げて別の位置に置く作業である．いま，**図4**に示すように①から④のワーク（立方体）をそれぞれ ①′～④′ の位置に置くことを考える．この作業は，繰返し処理の記述例であり，そのプログラムを**図5**に示す．図5では，まず CARRY コマンドの定義をする．次に現在のワークの位置データと移動先の位置データを作成する．作成したデータのリストの各要素で CARRY コマンドの実行を繰り返す．図5から FOR コマンドとコマンド定義機能を使うことによってプログラムが簡潔に記述できることがわかる．また，①～④を**図6**の①′～④′の位置に置くには，CARRY の内容を変更することなく

DATA 2=FOR I/(1, 4)　W 2+((I −1)*40, 0, 0, 0, 0, 0)

とすればよい.

4.2 かくはん作業

この作業は**図7**に示すように，二つのグラスにはいっている水を，薬品が入っている二つのビーカにそれぞれ入れて，かくはん棒でかきまぜる作業である．これは教示したデータを言語の中に組み込んでプログラムを作成する例である．その作業プログラムの作成例を**図9**に示す．プログラムを作成する場合，まず**図8**のように作業座標系 BEAKER で水をビーカに注ぐ作業（軌道名を POUR とする）と，かくはん棒でかきまぜる作業（軌道名を MIX とする）を TRAJ コマンドを使って教示しておく．次に，ビーカ，水が入っているグラス，かくはん棒が置かれ

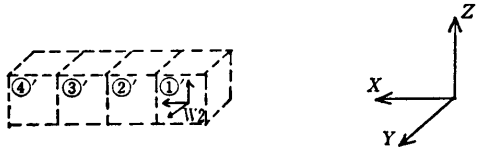


図 6　ワークの移動先
Fig. 6　Position of work.

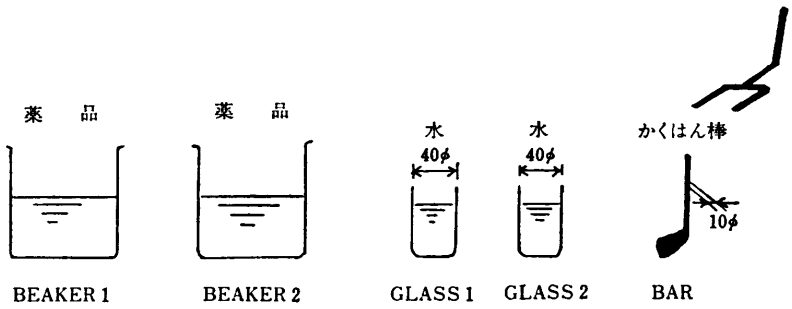
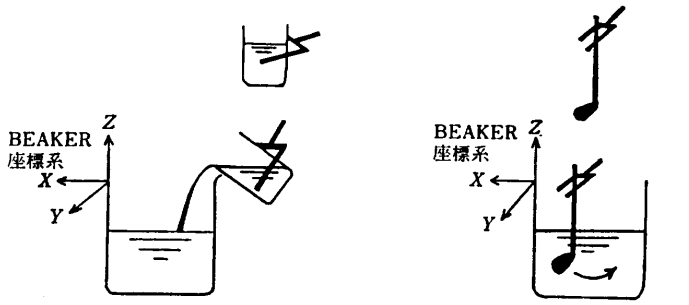


図 7　かくはん作業
Fig. 7　Mixing by stirring.



(1) 水を注ぐ作業(POUR)の教示　　(2) かくはん作業(MIX)の教示

図 8　軌道の教示
Fig. 8　Teaching of trajectory.

```
INPUT COMMAND ? DEFINE STIR
              1. COORD BEAKER          ・・・ロボットの手先をビーカーの位置へ動かし，作業座標系BEAKERを定義する．
              2. WITH BEAKER           ・・・以後の作業座標系をBEAKERとする．
              3. TRAJ POUR             ・・・水を注ぐ作業の教示開始
                   ロボットを動かしながら水をビーカーの中に入れる作業を教示する．(図8の(1))
              4. TRAJ MIX              ・・・かくはん作業の教示開始
                   ロボットにかくはん棒を持たせてかくはん作業を教示する．(図8の(2))
              5. WITH ABSOLUTE         ・・・以後の座標系を絶対座標系とする．
              6. COORD BEAKER1         ・・・ロボットの手先をBEAKER1の位置へ動かし，作業座標系BEAKER1を定義する．
              7. COORD BEAKER2         ・・・ロボットの手先をBEAKER2の位置へ動かし，作業座標系BEAKER2を定義する．
              8. HERE GLASS1           ・・・ロボットの手先をGLASS1の位置へ動かし，GLASS1の握点を教示する．
              9. HERE GLASS2           ・・・ロボットの手先をGLASS2の位置へ動かし，GLASS2の握点を教示する．
             10. HERE BAR              ・・・ロボットの手先をBARの位置へ動かし，BARの握点を教示する．
             11. PICKUP GLASS1 40      ・・・GLASS1を持ち上げる．
             12. POUR BEAKER1          ・・・水を注ぐ作業をBEAKER1の作業座標系で再生する．
             13. PUTDOWN GLASS1 40     ・・・GLASS1を元の所へ置く．
             14. PICKUP GLASS2 40      ・・・GLASS2を持ち上げる．
             15. POUR BEAKER2          ・・・水を注ぐ作業をBEAKER2の作業座標系で再生する．
             16. PUTDOWN GLASS2 40     ・・・GLASS2を元の所へ置く．
             17. PICKUP BAR 10         ・・・かくはん棒BARを持ち上げる．
             18. MIX BEAKER1           ・・・かくはん作業をBEAKER1の作業座標系で再生する．
             19. MIX BEAKER2           ・・・かくはん作業をBEAKER2の作業座標系で再生する．
             20. PUTDOWN BAR 10        ・・・かくはん棒を元の所へ置く．
             21. HOME                  ・・・原点位置へ戻る．
             22. END
INPUT COMMAND ? STIR                   ・・・プログラムを実行する
```

軌道の教示 / 作業対象物の位置に関する教示 / 動作手順の記述

図 9 プログラム例
Fig. 9 Program of mixing up.

ている現在の位置・姿勢を教示する．これらのデータは代入文で直接与えることもできる．このあと動作手順を記述して実行する．

一般に，教示データを作成する場合二つの方法がある．一つはロボットを直接手にとって腕や手先の動きを教える直接教示法であり，もう一つは手元のリモートコントロール装置（一般にティーチングボックスと呼ばれている装置）を操作してロボットを動かす間接教示法である．本実験で用いたロボットは直接教示法ができないので図8に示す作業の教示は間接教示法で行った．すなわち，ティーチングボックスとしてTSS端末のキーボードを操作しながら図8に示す作業を行い，その作業経路の主要点でのロボットの手先の位置・姿勢を教示した．

図9から，教示データを利用することによって作業プログラムの作成が容易となることがわかる．

5. システムの構成

IML は FORTRAN で書かれたリスト処理パッケージ[9]を使用しており，現在九州大学情報処理教育センターの M-360 上でインプリメントされている．システムの構成図を図 10 に示す．

実験に用いたロボットは三菱電機製のムーブマスター (RM 101)[10] である．このロボットは多関節型の人工の手で五つの自由度（指の開閉の自由度は除く）を有している．したがって任意の位置で，手先が任意

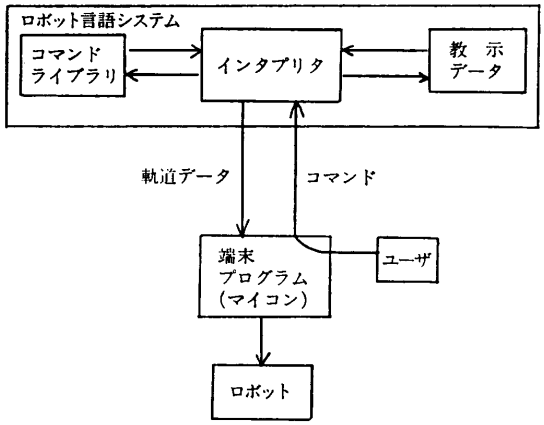


図 10 システムの構成
Fig. 10 System configuration.

の姿勢をとることはできない．ロボットはマイコン (FM-8) を介してホスト計算機 (M-360) とつながっている．マイコンは，端末としてロボットに作業を教示したりコマンドを入力するのに使用され，ホスト計算機の TSS 環境下で動く．コマンドが入力されると，インタプリタがそれを解釈し，コマンドが定義したコマンド名であれば展開し，教示データが必要ならばそれを利用し，端末に軌道データを与えロボットを動かす．作業対象物やロボットの位置・姿勢を教示するときはマイコンのキーボードをティーチングボックスとして用いるが，このキーをロボットの関節ごとに対応させると教示しにくいので，手首の位置の X, Y, Z 軸方向の動きに対応させている．ロボットの現

在値は原点位置からのパルス数から求めた.

6. あとがき

教示データと関数型言語の特徴を導入したロボット言語 IML について述べた. そして, いくつかの作業例を IML で記述し, その作業を実現することによってその有効性を確認できた. 現在, IML にはセンサからの情報を入力する機能はない. 一般に産業用ロボットの作業環境は比較的よく整備されているが, 精度が要求される作業や組立て作業を行うにはセンサ情報を処理することが必要であり今後の課題である.

参考文献

1) Mujtaba, S. et al.: AL User's Manual, Stanford A. I. Lab. (1979).
2) Takase, K. et al.: A Structured Approach to Robot Programming and Teaching, *IEEE Trans. Syst. Man Cybern.*, Vol. SMC-11, No. 4, pp. 274-289 (1981).
3) Inoue, H. et al.: Design and Implementation of High Level Robot Language, Proc. 11th ISIR, pp. 675-682 (1981).
4) Bonner, S.: A Comparative Study of Robot Languages, *Computer*, Vol. 15, No. 12, pp. 82-96 (1982).
5) User's Guide to VAL, Version 11, Unimation Inc. (1979).
6) Pratt, T. W.: *Programming Languages Design and Implementation*, Prentice-Hall, Englewood Cliffs, N. J. (1975).
7) 新井民夫: ロボット言語—記述形式と知的レベル—, 計測と制御, Vol. 21, No. 12, pp. 14-19 (1982).
8) 竹下 享: プログラム言語 APL, *bit*, Vol. 11, No. 1-Vol. 12, No. 5 (1979-1980).
9) 奥川伸一: 実験データ解析支援システムの開発, 九州大学修士論文 (1982).
10) 三菱電機: ムーブマスター RM-101 取扱説明書 (1982. 3).

(昭和58年10月12日受付)
(昭和59年2月14日採録)