移動機械の指令制御システム

中村達也，福井郁生，小鍛治繁（機械技術研究所）

1．はじめに

人工知能の分野では自然言語処理やCADを対象として，特に推論及び知識表現に関する研究が盛んであり，そのための道具としての言語もPLANNER[1]やCONNIVER[2]を初めとして優れた言語及びアイデアが開発されている。一方，ロボットの分野では主としてハードのメカニズムやその制御に研究の重点が置かれていて，言語に関しては組み立て言語AL[3]などの研究はあるがあまり十分になされていないように思われる。そこで筆者らは人工知能のアプローチを移動機械の制御に導入して，より高度の知能をもちしかも実用面を考慮した移動機械の指令制御システムの研究を行っている。

このようなシステムを実現するための問題点を明らかにし，解決するための実験システム（MEL.DEIC-1と呼んでいる）を作製したので，その概要とその情報処理系について述べる。ここで移動機械は単に物の運搬だけでなく，それに加工工具をとりつけて例えば造船でみられるような大きな鉄板を切断するなどの，移動機能をもつ工作機械[4]を目標としたものである。人から与えられた指令（設計図面など）をもとに指令制御システムは作業計画を作成し，又その計画にもとづいて移動機械の制御方法とパラメータを決定する。実行に際しては異常事態の監視と異常時における対処を行う。

このため実験システムの設計方針として以下を採用した。

(1) サーボなどの実時間の制御はマイクロコンによって処理し，作業計画の作製などの知的処理はミニコンによって行うという，並列かつ階層的なシステムとする。

(2) 知的処理は本来記号処理であるとの考えから，ミニコンのシステムとしてはLISP-likeな言語を利用する。

(3) 実験システムであることを考慮して，出来る限り容易にインプリメントする。このため，純数値計算はアセンブラヌはフォートランで書き，それをLISP関数として登録する。

(4) データベース（ディスクファイル）をもち，データの検索はパターンとの照合によって行う。

2．システムの概要

2.1 システム構成　システム構成を図1に示す。移動機械は図2に示すように，DCモータで駆動される2個の駆動輪と2個のキャスターをもつ。移動機械上に位置計測用のTVカメラと制御用のマイクロコンピュータが積載されている。TVカメラは移動機械の中心位置に上向きに置かれ，水平面内で回転するミラーを通して外部環境を観測する。マイクロコンには約500語のモニタが常駐しており，ミニコンとの交信，ミニコンからマイクロコンへのプログラムのローディング，若干の実時間処理機能を行う。マイクロコンとミニコンは無線送受信機で接

続しており，伝送速度は 600 bps である。ミニコンには既製の RDOS の下に，マイクロコンとの交信を行う交信プログラム（約 100 語）と作業計画などの作成を行う記号処理プログラム LISP-DB（約 5 K 語）が並列に存在している。残りはディスクファイルのコア領域とスタックなどの作業領域である。

なお図2の移動機械は完成したばかりである。以前は1個の駆動輪と舵取り用の2個の車輪とからなる移動機械を使用し，本実験システムはそれを対象に作製された。現在新しい機械向きにシステムの修正を行っている。なおミニコンは ECLIPSE S/200（コア 32 K 語），マイクロコンは旧は TOSMIC-12（500 語の ROM（モニタ），3.5 K 語 RAM），新は μ-NOVA（4 K 語 RAM）を使用している。

2.2 移動機械の制御

2.2.1 位置の計算 移動機械の位置は外部環境を TV カメラで観測することにより決定する。この種の研究では火星探査ロボット[5]のように自然環境を対象にしたものが多い。しかし自然環境を対象にする場合には高度のパターン認識能力を必要とし，それを可能とするほどのパターン認識の分野の急速な進展は望めそうにない。そこで筆者らは容易に位置測定ができるような工夫を環境に施す方法をとる。

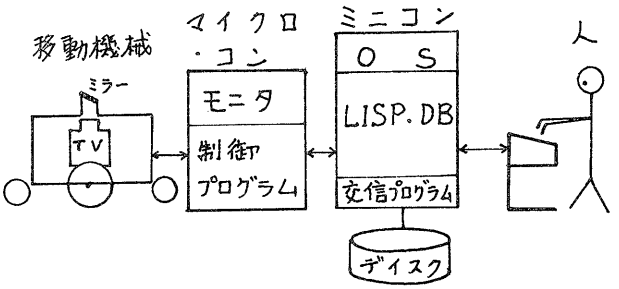


図1. システム構成

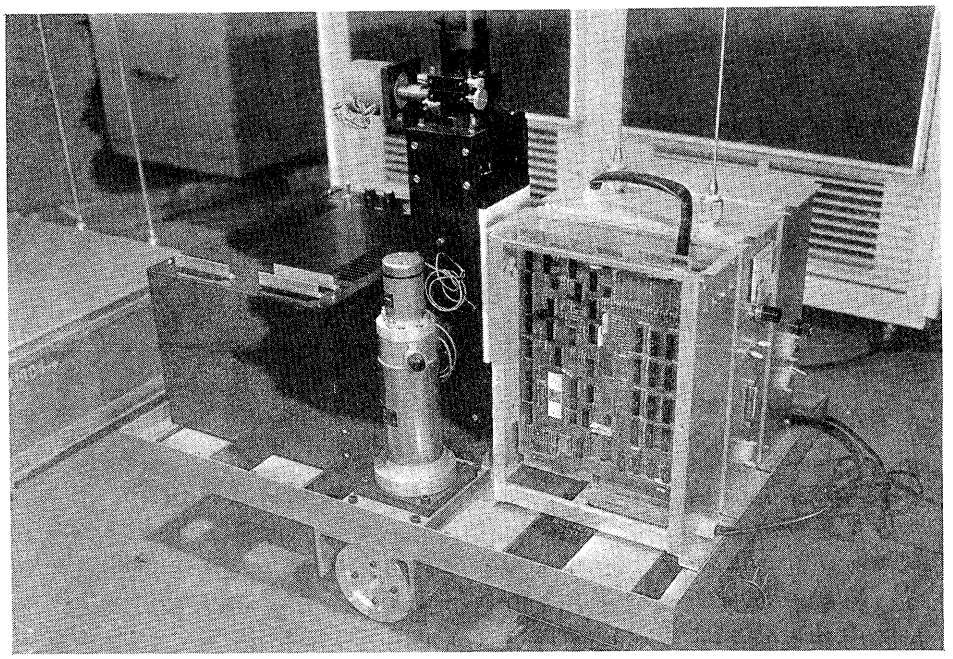


図2. 移動機械

本システムでは環境側に基準点として特定のマークを設置し，図3のように移動機械上に固定した軸 $\overrightarrow{CA}$ とマークへのベクトル $\overrightarrow{CB}$ とのなす角度 $\theta=\angle ACB$ を測定する。θ だけでは平面内の位置を決定できないので，移動機械の運動モデルを利用し積分により位置を計算する。

運動モデルは低速移動の仮定をおき，ダイナミックスを無視した静的モデル，即ち移動の軌跡は曲率 k と速度 v によって定まる，を用いている。従って運動方程式は

$$\begin{aligned}\dot{x} &= k\dot{y}\\ \dot{y} &= -k\dot{x}\\ \dot{x}^2+\dot{y}^2 &= v^2\end{aligned} \qquad (1)$$

となる。式(1)で k，v を一定にとれば移動軌跡は等速円運動となる。

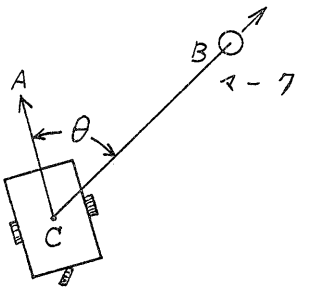


図3、位置の計測

角度 θ の1次元情報だけで x, y を計算する方法も得られているが，相当の計算を必要とするのでマイクロコンで行う訳にいかない。そこで本システムでは別の情報として速度 v が既知であるとしている。速度 v は現在のところ測定していないで，DC モータへの入力電圧より v を計算している。将来は v の測定も行うことを予定している。θ および v を利用した位置の計算式は行列形式で，

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix}=\frac{1}{\sqrt{x^2+y^2}}\begin{pmatrix}x & -y\\ y & x\end{pmatrix}\begin{pmatrix}-v\cos\theta\\ v\sin\theta\end{pmatrix} \qquad (2)$$

と表わされる。実際には式(2)を差分式にして逐次計算により位置の計算を行っている。ただし，初期の x, y は既知であるとしている。

<u>2.2.2 位置決め制御</u> 制御変数は k および v で，k は舵取り電圧に相当する舵取り角をポテンショメータにより設定しおり，v は DC モータの入力電圧により設定する。移動機械の位置決めは角度を入力とする比例制御で行っている。制御方法を point-to-point の位置決めを例にとり簡単に説明する。(図4(a)参照)

現在位置を P，マークを M，移動目標点を Q とするとき，移動機械の軸 PR と目標への直線 PQ となす角度 $\alpha=\angle QPR$ に比例した舵取り角をとる。もちろん比例定数は実験等により適当に選ぶ。図4(b)のように定められたコースを追跡する場合も，同様に移動機械から特定の距離だけ前方のコース上に Q を定めればよい。このとき Q 点は車の移動とともに移動する。実験によれば，カーブにおいて一定のコースズレが起るので，精度を上げるためには速度を遅くするか，制御に補償をする必要がある。図5は位置決めの軌跡をディスプレイ画面上に表示したものである。同図は計算上のものであるため，ディスプレイ上では Q 点に到達しているが，実際の移動機械は Q 点より若干ずれている。

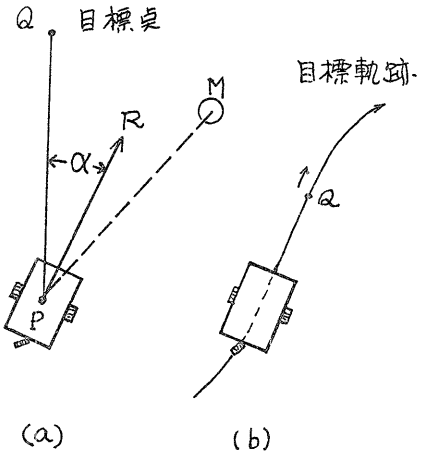


図4　位置決め制御方法

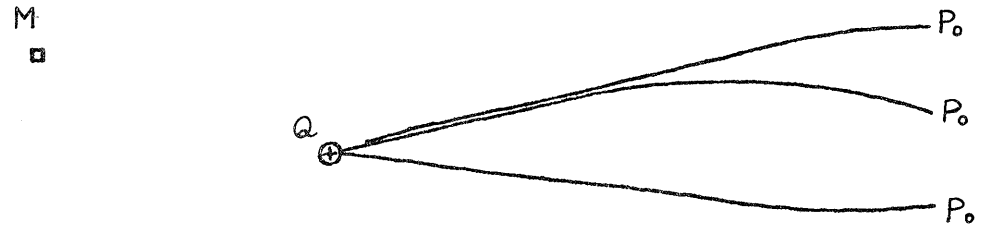


図5、 Point-to-Point の位置決め軌跡

3. 情報処理系

3.1 マイクロコン・モニタ　マイクロコンで行う処理を，1)タイマーによって制御される処理，2)上位のミニコンからの要請で行う処理，3)のべつ幕無く行う処理に分類した。具体的には，1)は一定時間毎に角度θを測定して行う位置の計算，2)は車の始動，停止，パラメータの変更，3)は位置決め制御である。モニタはこの3種の処理を行うプログラムを独立に書くことを可能にしている。3種の処理の優先順位は1)，2)，3)の順である。プログラムの実行中に優先順位の高いプログラムに起動がかかると，レジスタの内容等を退避し現在実行中のプログラムを中断して順位の高いプログラムを実行する。処理が終了すれば以前中断したプログラムに再び制御が戻る。

ミニコンからマイクロコンへのコマンド及びマイクロコンのプログラムからのシステム・コールを以下に説明する。なおミニコンとマイクロコンとの交信はASCII符号で行っている。

コマンド：

1) L＜プログラム＞：＜プログラム＞をミニコンからマイクロコンにロード。
2) S＜アドレス＞：　＜アドレス＞からプログラムを実行する。
3) P：現在実行中のプログラムを中断して，作業用の変数，レジスタ等の内容を退避する。
4) R：現在実行中のプログラムを止め，以前中断したプログラムを復帰する。
5) E：タイマで制御されるプログラムの登録を取消す。

システム・コール：

6) IN：ミニコンからデータを入力する。
7) OUT：ミニコンにデータを出力する。
8) DELAY：何秒後にどのプログラムを実行するか指定して，タイマーで制御されるプログラムを登録する。
9) TTI, TTO：テレタイプの入出力。

3.2 ミニコン用システム　ミニコン側で行う処理としては
1) 人からの指令で与えられた作業目標を達成するための作業計画の作成。
2) 作業計画を実行するための制御方法及びパラメータの決定。
3) マイクロコンへの制御プログラムの転送。
4) 作業実行の監視，緊急時の処理，作業の評価による学習・適応。
が考えられ，これらを実現する言語には次の機能が要求される。
1) 移動機械が置かれている環境の地図，作業計画や制御方法を決定するためのアルゴリズムやノウハウを記憶するデータベース。

2) データを利用して作業計画等を作成するための推論メカニズム.
3) 作業の実行中の処理を行う実時間機能とそのための制御構造.

そして，これらの機能をもった言語，もしくは言語の研究手段としてLISPが最も注目されている。最近の傾向としては，ファームウェア化[6]等による高速処理，コルーチン，バックトラックなどの高度の処理を可能にする制御構造[7]，パターンマッチング，記号処理の機能がある。

本ミニコン用システムとしては次にデータベースが必要であるとの立場から，データの検索に必要なパターンマッチングの機能をもつ，LISP 1.5 レベルの言語LISP.DBを作製した。そしてLISP.DBは上記の機能実現のための実験道具であるので，出来る限り容易に作製できることを設計方針とした。そこで通常のLISPのようなリスト表現をやめ，データの型は記号列のみとした。ただし，LISP 1.5 と同じ機能を持たせうるように，"(", ")" 及びスペースは特別に扱っている。さらにLISP関数の引数と値を授受するためのスタック以外には，レジスタの内容の退避・復帰，関数を呼び出したときの飛び先と戻り先等のコントロールは既製のフォートラン・スタック及びライブラリを利用してプログラムの作製を容易にした.

4. LISP.DB

データは記号列のみで扱っているので，中味は通常のLISPとかなり異っている。しかし形式上はほぼ同じである.

プログラムは図6のようにFORM，即ち

(<関数名> <引数> <引数> ···)

の形式でコンソールから入力する会話型となっている.

```
* (CAR (QUOTE (A B)))
* A
*
```

図6. 会話例

記憶領域及び処理時間を節約するためニーモニックは短く，例えばQUOTEの代りに単にQと書くなどの簡略化を行っている．しかし本稿では理解を容易にするため簡略化は行っていない.

4.1 データ　データはコアのデータ領域と同じ大きさのディスクファイルに格納され，ディスクファイルは夫々0,1,2,…の番号が付けられている．最初LISP.DBを起動するとファイル0がコアにロードされる．ファイルに格納されるデータは"("と")"の数が一致した記号列ならば何でもよい。ただし，スペース及び"(", ")"はデリミタとして特殊な扱いをうける。データは通常のデータ，例えば

(CAT MACHINE X=3 Y=4)

の他に，通常のLISPではproperty listに記憶されるEXPRの関数も

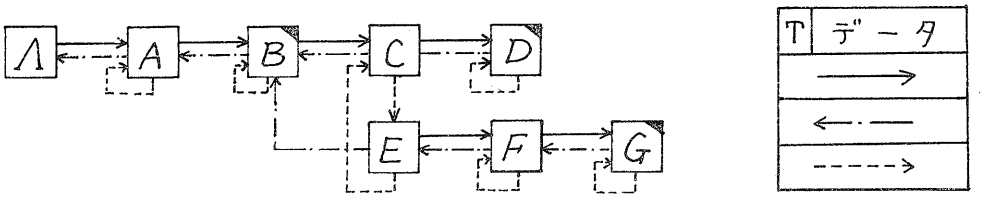
.FUNC CADR (X) (CAR (CDR X))

の形でファイル内に格納される．ここに.FUNCはユーザ定義のEXPR関数であることを意味する.

4.1.1 データの内部表現　データは全て記号列として扱っているが，データの検索を高速にするための試みとして1個の記号を1個のブロックにし，図7の様に

ポインタによって連結した構造によって表現している。図7(a)において左端のブロックがベースレジスタに相当し，右へ記号列の1番目の記号，2番目の記号…とポインタによって結合される．右肩が塗りつぶしてあるブロックはそこを終端とする記号列があることを示す。記号列 AB，ABCD，ABEFG は部分記号列 AB が共通であるので，共通のブロックを利用している．この様に最初の部分記号列の部分の重複分だけメモリが節約される．

又，データ領域内において，記号列は終端のブロックを示せば逆向きのポインタ（同図では一点鎖線）をたどることによって得られる．図7(b)はブロックの内容を示す．Tは1ビットのタグで，T=0 or 1 はそのブロックを終端とする記号列がない or あることを示す。



(a)　　(b)

図7　記号列AB，ABCD，ABEFGの表現

4.1.2　データの検索　データ領域からのデータの検索はパターンとの照合で行う。現在許されているパターンは，パターン変数を!x と書けば，

〈パターン〉＝〈記号列〉{△}!{〈変数〉}△〈記号列〉…

である。ここで，△はスペースを，{ }はオプションである。!〈変数〉は括弧のバランスがとれており，しかも終端の記号が元の記号列の終端であるか，又は次の記号がデリミタであるような部分記号列とマッチングする。データの検索は関数 FETCH で行う。

例：データ領域内にはデータ"(THIS IS A PEN)"だけが格納されているとする．

(FETCH (QUOTE !))：マッチングし値として"(THIS IS A PEN)"が返される。

(FETCH (QUOTE (TH!x)))：変数xに" IS IS A PEN"が代入される．

(FETCH (QUOTE (THIS IS A !x)))：変数xに" PEN"が代入される．

(FETCH (QUOTE (!x IS A PEN)))：変数xに"THIS"が代入される．

同一のパターンにマッチングする記号列がいくつもあり，夫々について逐次処理を行うために，マッチングの状態を記憶する場所が用意されている。それらは1，2，…と番号付けられている。この様に次々と検索したい場合には，FETCH の第2引数に状態を保存する場所を示す番号を与える。2回目以降のマッチングは関数 NEXT を用いる。

例：　(FETCH　〈パターン〉　1)　　(3)

　　(NEXT　〈パターン〉　1)　　(4)

ここで式(3)と(4)の〈パターン〉は同一でなければならない。マッチングするデータがない場合には＊F＊が値となる．

その他，データ領域にデータを格納及び削除する関数に ADD，REMOVE がある．

例：　(ADD 〈記号列〉)，(REMOVE 〈記号列〉)

4.1.3 ファイルの変更　ファイルの変更は
（CHANGE <ファイル番号>）
で行う。現在コアにあるファイルはディスクに格納され，<ファイル番号>のファイルがロードされる。

4.2 関数　関数の型はSUBR，FSUBR，EXPRである。FとSUBRは組み込み関数で，SUBRは実引数を評価してから呼び出す関数，FSUBRは評価しないで呼び出す関数，EXPRは引数を評価してから呼び出すデータ領域内に定義された関数である。ただし実引数の個数は任意である。実引数の値は仮引数に順々に割り付けられる。実引数の数が少い場合には残りの仮引数にはNIL空が割り当てられる。実引数の数が多い場合には余分の実引数は無視される。
EXPR関数は既製のLISPの使用経験から全てPROG形式とした。EXPR関数のデータ領域への登録は
（DEFINE <関数名> <仮引数リスト> AUX <補助変数リスト> <body>）
で行う。補助変数の初期値はNILである。<body>はLISP 1.5におけるのと同じである。

4.3 変数の束縛　変数の束縛にはスタックを用いている。

5. まとめ

移動機械の指令制御のための実験システムMEL.DEIC-1の概要とその情報処理系について述べた。ミニコン用システムLISP.DBは現在組み込み関数の拡充と改良を行っている。まだ実験によるシステムの評価も充分にはなされていないが，今後の改良の方針として以下を考慮している。
1) 関数型，データ型の拡張
2) オーバーレイの利用
3) パターンの拡張
4) コンパイラ，ローダの作製
又，将来的には以下が重要であると思われる。
5) pattern directed invocationの機能
6) LIFO以外の制御構造
7) 実用化のための高速化

参考文献

1) C. Hewitt, "Description and Theoretical Analysis of PLANNER", MIT No. D-0530 (1972)
2) D.V. McDermott and G.J. Sussman, "The CONNIVER Reference Manual" MIT AI Memo No. 259a (1974)
3) R. Finkel et al, "AL, A programming System for Automation", Stanford AI Memo AIM-243 (1974)
4) 中村，福井，小鍛治，小島，"大工システム（MEL.DEIC）"，機械技術協会会報 Vol. 29 No. 1 (1977)
5) Y. Yakimovsky and R. Cunningham, "DABI -- A Data Base for Image Analysis with Nondeterministic Inference Capability", Pattern Recognition and Artificial Intelligence,

Academic Press (1976)
6) 島田, 山口, 坂村, "LISP マシンとその評価", 信学論 Vol. J59-D, No. 6 (1976)
7) D.G. Bobrow, B. Wegbreit, "A Model and Stack Implementation of Multiple Environments", CACM Vol. 16, No. 10 (1973)