

ロボットビジョン言語

～照合レベルのロボットビジョン言語の試作～

松下俊夫, 佐藤知正

(電子技術総合研究所)

1. はじめに

近年、ロボットへの教示法の効率化や、センサ情報を利用した柔軟な制御記述の必要性からロボット言語の開発が盛んである。一方、視覚処理においても、ロボットへの応用をはじめとして広くアプリケーションユーザに利用されるためには、使い易くかつ十分な記述能力を備った視覚処理用言語(ビジョン言語)が必要である。我々は、組立て作業等に用いられるロボット用ビジョンシステムとして、距離プロフィールを用いた能動的バリフィケーションビジョンを提案している⁽¹⁾。現在このシステムの記述を目的としたロボットビジョン言語(以下RVLと略す)の研究を進めているが、その考え方の枠組を定め、実験においても基本機能の作動を確認したので報告する。本資料では、まず我々のRVLの視覚処理プロセス上での位置づけを行い、現在開発を進めている言語の基本仕様を概説する。次に実験システムと開発の現況を報告し、最後に使用例を述べる。

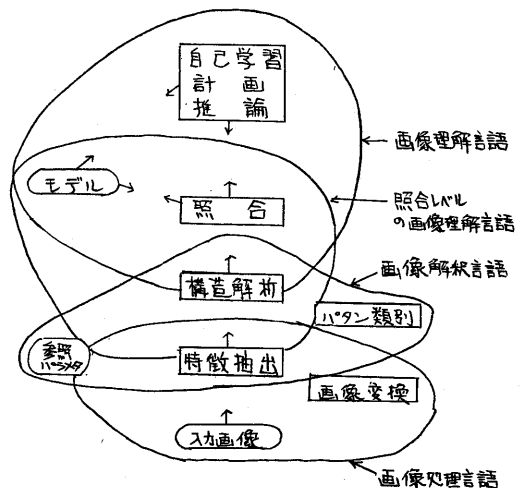
2. ビジョン言語研究の現況とRVL

ビジョン言語の開発には従来より表1のような幾つかの系統がみられる。RVLはロボットユーザ用の言語を目指しており、低レベルの画像処理記述よりも上位のプロセスを記述することを表1. ビジョン言語開発の系統

1. 画像処理パッケージ+コマンド言語	PICASSO, PIC ⁽³⁾
2. 画像処理用高級言語	PAL, Parallel PASCAL ⁽²⁾
3. 問題向き画像解釈言語	PILS ⁽⁵⁾
4. ロボット言語への視覚機能の付加	RAIL, VAL ⁽⁷⁾
5. 画像DB照会言語	GPE, GRAIN ⁽⁸⁾
6. CAD用言語	

目標とし、かつ汎用的な3次元作業に対応するため、対象物のモデル化機能を備える必要がある。一方、作業の実時間要求に従従するためには、視覚対象物がある程度単純でなくてはならない。現在視覚処理機能を組み込んだロボット言語も現れ始めてきているが、主流はRAILに見られるごとく2次元のシルエットを対象としており、単純性と高速性の要求に沿うものの3次元への拡張には距離があり、またモデル構成機能が十分でない。図1に画像理解に至るプロセスと視覚処理用言語のレベルを整理した。各レベルの境界は厳密なものではないが、使い易さという点から境界内の下の機能に行くほど詳細がユーザに隠されるよう設計するのが一般的であろう。逆に上部の機能を実行するためには処理シーケンスの細部までプログラムしてやる必要がある。通常画像処理用高級言語と呼ばれているものは、特徴抽出のプロセスまでの記述の合理化を目的としており、また

図1. 画像理解と言語レベル



RAIL型の視覚機能は、抽出特徴を用いたパターン類別の水準を対象としている。RVLは画像理解言語のうちAI的な機能を除外した照合レベルに位置する言語であり、構造解析に基づく照合プロセスの記述と対象物モデル記述機能を備えている。

ところで、視覚の対象世界は極めて多様性に富み、画像一般を言語の対象とすることは非常に困難であり、また処理も複雑となって高速性の要求されるロボットビジョンには適さない。RVLでは視覚データを対象物の1次元距離断面(距離プロフィール)に限定し、これらの問題を解決している。

3. 距離プロフィールを用いた能動的バリエーションビジョンシステム

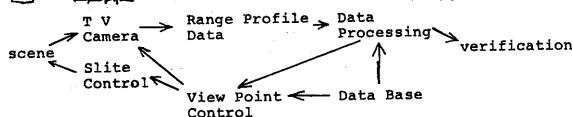
このシステムは機械部品の組立て作業等を行うロボット用のビジョンシステムである。腕作業との結合には位置情報の取得が必要である点と、画像処理速度が高速である必要があることから、投射スリット光のステレオ視により1次元データである物体の距離断面を抽出し、これに基づいた処理を行っている。システムには対象物と環境とに対する知識が十分与えられているものとし、内部のモデルに従い存在検証のための適当なプロフィールの抽出を試みる。検証の結果が不十分な場合には、投光方向やカメラ位置を変化させてプロフィールの単純さを補い、適用範囲を拡大することを意図している。(図2)

4. RVLの言語仕様

4.1 ホスト言語

RVLは電総研で開発されたLisp系言語PETL(9)上にインプリメントさ

図2 距離プロフィールを用いた能動的バリエーションビジョン



表現レベル データタイプ 処理プロセス

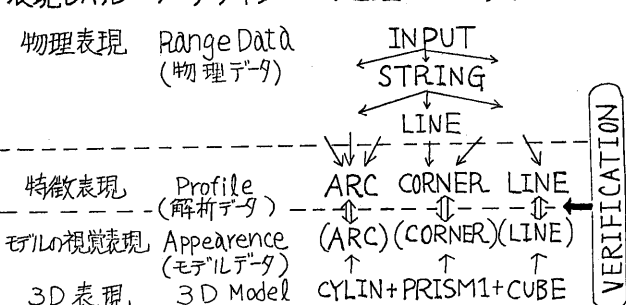


図3 RVLにおけるデータ階層

れており、本言語もLispの関数評価形式を利用している。

(2) データタイプ

RVLが内部で使用している固有のデータタイプとして、

- ・プロフィール/距離性データ
 - ・プロフィール記述リスト
 - ・Appearanceデータ
 - ・VIEW FRAME, OBJECT FRAME (カメラ, 物体の座標系)
 - ・TRANSFORM (座標系の変換子)
- 等がある(未インプリメントのものを含む)。

(3) 対象物体の記述

対象物体のモデル構成法は照合とその結果に基づく物体の計測値抽出の手法を規定するため、RVLにおける最も重要な部分である。RVLでは対象物体にかかめる視覚データを図3のような層(ないし4層)の階層構造でとらえている。すなわち、視覚世界の物理データとして距離データがスカされ、これが解析され特徴表現としてプロフィールに変換される。一方、対象物の視覚モデルとしては見元のモデル(Appearance)を考える。対象物体の同定とは、ProfileとAppearanceとの照合をとることには他ならない。

Appearanceの設定はユーザが直接指示する場合と、3次元モデルからグラフィックス的手法で自動的に行う場合と二つの方法が考えられる。

照合検証の方法には profile と appearance の 2 つの表現を作り両者のマッチングをとるやり方も考えられるが、RVL では appearance 記述に基づき距離データのプロフィールの変換を試み appearance の条件と適合するプロフィールが得られるか否かで検証を行っている。3次元要素モデルによる環境記述の方法をシンタックス風にかくと次のようになる。ここで、

```

WORLD      :: OBJECT/(LOCATE WORLD OBJECT TRANSFORM)
OBJECT      :: COBJECT_FRAME,MODEL
OBJECT_FRAME :: CFRAME_ORIGIN,FRAME_DIRECTION
FRAME_ORIGIN :: CX,Y,Z
FRAME_DIRECTION :: CX,Y,Z
MODEL       :: ELEMENT/(BIND MODEL ELEMENT TRANSFORM)
ELEMENT     :: NIL/CYLINDER/CONES/SPHERE/CUBE/....
TRANSFORM  :: LTRANS,ROTJ

SCENE       :: CVIEW_FRAME,WORLDJ *(VIEW VIEW_FRAME WORLD)
APPEARANCE :: EVIEW_FRAME,OBJECTJ *(VIEW VIEW_FRAME OBJECT)

```

LOCATE, BIND はそれぞれ空間配置関数と結合関数を表している。また VIEW は VIEW_FRAME に基づいてスリットによる切断面を生成する関数である。RVL は現在 3次元モデルを持っておらず、関数 VIEW も存在しないが、appearance を求めるのに常に 3次元モデルを必要とするのは不都合な面もある。RVL ではユーザが次の構文に基づき appearance の生成を行う。

```

APPEARANCE :: APPEAR_TYPE
              / (BIND APPEARANCE APPEAR_TYPE TRANSFORM)
APPEAR_TYPE :: NIL/LINE/CORNER/ARC/....

```

なお appearance による物体記述は、BIND のような関数が存在しなくとも、LISP の属性リスト機能を用いて行うことが可能である。(付録参照)

(4) 主な関数

(TAKERD no) により撮像が行われ距離データが抽出されて、no の番号が与えられる。物体の検出は (FIND 'object) により行うことができる。現在インプリメントされているのは、ある APPEAR_TYPE を持つ単一要素物体検出機能だけであるが、object の APPEARANCE 指定に基づき距離データを解析して対象物の検出に成功すると値 T を返す。この間 object のプロフィールを記述するデータ構造が形成される。FIND された object に対し、

(MEASURE 'object 'target) を実行すると、object に関し target で示した測定値が値として返される。FIND や MEASURE は一種の関数として定義されており、object の記述を参照し、その指示に基づき PETL のキーインデックスタイプのファイル(データベースと呼ばれる)に登録された処理系統の実装を評価する。このためデータベースに登録された記述の修正により容易に RVL 機能の拡充を行い得る。一般的なモデル作成関数はまだ組み込まれていないが、簡易版として (DEFVIEW 'object '(appear-type list)) により OR の形で APPEARANCE を定義し、(SETPARA 'object 'appear-type 'parameter-name 値) でプロフィールの幾何学的仕様を指定するものとしている。

(5) 中間言語に対応した関数

後述する中間言語の命令コードに対応した関数が PETL 関数として定義されている。したがって実験者は RVL 上で画像の入出力、プロフィール処理、プロフィール要素リストのソータ等に対話的に行うことができる。これらの関数と PETL 本来の関数を組み合わせて、ユーザは独自の関数を自由に定義することが可能である。

5. 中間言語とサブルーチンパッケージ

RVL の母体である PETL は LISP 系言語なので、PETL 上で画像処理をはじめとする数値処理を行うのは、処理速度とプログラム開発の面で不利である。このため、画像処理、プロフィール処理、その他のユーティリティ機能等大部分の機能は FORTRAN で書かれたサブルーチンパッケージを利用して行っている。RVL 上での関数は、サブルーチン起動のコマンドコードである中間言語コードに

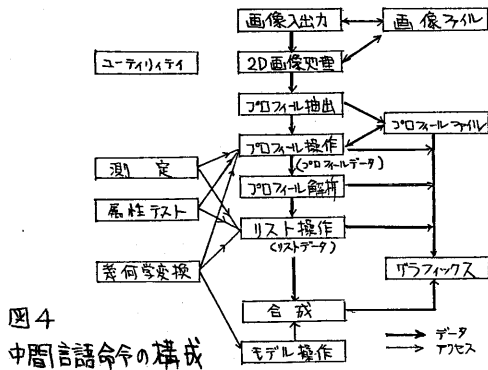


図4

中間言語命令の構成

落とされ、中間言語インタプリタからサブルーチンがコールされる。この部分は画像処理用サブルーチンパッケージに対するコマンドラングージタイプの画像処理言語に対応している。図4に中間言語命令構成をグループごとのブロックとして示した。一般に画像処理のためには、画像入出力、前処理、特徴分析、ファイル操作、結果の表示と数多くの要素機能が必要で、中間言語がこれらを一通りサポートしていることが、上位システムでのプログラミング負担を減らすために必要である。現在、中間言語コマンド数は約100種あり、さらに増強中である。

中間言語の具体例を幾つか示す。プロフィール処理を進めて行くためには、プロフィール要素の物理量の計算や、要素間(単項演算も含めて)の関係判別を行う機能が必要となる。Vを頭文字に持つ中間言語コマンドは測定値又は計算値を求め、Jで始まるコマンドは要素間の多値状態を判別し、Tのものは2値関係の判別のために作られている。その例を示すと(引数は略)、
VRDOM...要素の距離方向の存在範囲
VEIL(M,R)...要素の左端(中,右)点座標
VITEML...要素の長さ
VCRSP...2線分の交点座標
JCCVX...要素の凹凸性
JHOR...要素間の水平位置関係
JRAR... " 遠近 "

TGAPI...要素間の距離が閾値以下か
TIPDS...要素位置が指定区間内か
TCMPL...要素長が指定範囲であるか
TFIT...直線(M)当てはめ誤差が閾値以下か
等々である。

6. 実験システム

実験システムの構成を図5に示す。主計算機はVAX11/780であるが、PETLがCOSMO上にあるため図の構成となっている。PDP11/45は画像入出力装置のコントロールを行う。画像入力は通常のエ-TVカメラにより256×240点のサンプルを行いVAX上で距離データに変換しているが、スリット光の位置抽出の専用装置で前処理を行い高速化を図る予定である。PETLプロセスとVAX側の交信は端末回線で行っており、その制御はVAX上の中間言語コードインタプリタ("VMON"—View Monitorと呼ぶ)で行っている。交信は8組データを用いており、中間言語コード、引数値、処理結果が転送される。ユーザは端末より直接VMONへ中間言語コードを送ることも可能で、RVLと独立に画像処理機能のチェツクを行える。

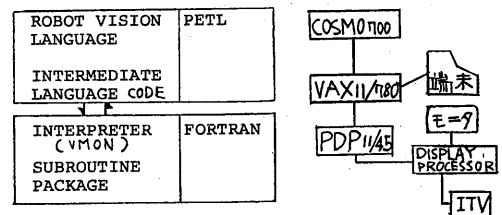


図5 実験システム構成(ソフトとハード)

7. RVLによるプログラム例と実験

(1) 対象物の記述

距離プロフィールを用いたビジョンシステムは、対象物として機械部品ののように基本的な幾何学的形状で合成されたソリッドな物体を想定している。写真1はガソリンエンジンの側面カバーと排気ポートにスリット光を当てた状態である。今エンジンとカメラの位置



スリット光



図6. エンジンの記述

写真1

関係がほぼこの

ように拘束されていると仮定すると、排気ポートの検出を目的としたエンジンの記述はその APPEARANCE を図6のように構造化することにより以下のように書き得る。

```
(CREATE 'COVER) (1)
(DEFVIEW 'EDGE1 'LINE1) (2)
(SETPARA 'EDGE1 'EDGE 2000) (3)
(DEFVIEW 'EDGE2 'LINE1) (4)
(SETPARA 'EDGE2 'EDGE 1500) (5)
(BIND 'COVER 'EDGE1 (NILTRANS)) (6)
(BIND 'COVER 'EDGE2 '((-3000) (NILROT))) (7)

(CREATE 'PORT) (8)
(***** (DEFINITION OF SIDE1 AND SIDE2)
(BIND 'PORT 'SIDE1 (NILTRANS)) (9)
(BIND 'PORT 'SIDE2 '((-1900 0) (NILROT))) (10)

(CREATE 'ENGINE) (11)
(BIND 'ENGINE 'COVER (NILTRANS)) (12)
(BIND 'ENGINE 'PORT '((-1600 3500) (NILROT))) (13)
```

(1), (8), (11) はそれぞれ新しい OBJECT 定義のための初期化であり、(2)~(5)で COVER の構成要素を記述し、(6), (17)で COVER に結合している。同様に PORT を定義し、COVER と PORT を ENGINE に結合すると ENGINE は図6のような tree 構造で記述されることになる。

(2) 中間言語命令を用いたプロフィールの抽出

次の例はデータを1回サンプルし、一定長以上の STRING を抽出してそれを線分で近似する過程を中間言語レベルの関数で表記したものである。

```
(VAR IS IE) / SEARCH DOMAIN
(SETQ IS IE 256) / DATA ARRAY ID.NO
(VAR IR IH ID) / LIST ID.NO
(VAR L1 L2 L3) / PARAMETERS USED IN SUBROUTINES
(SETQ L1 10 L2 11 L3 12)
(VAR DT1 DT2 LNG) / INPUT RANGE DATA
(SETQ DT1 250 DT2 500 LNG 10) / DIFFERENTIATE IR
(TAKED IR) / SET HORIZONTAL COORDINATE VALUE
(ODIFI IR ID) / SEARCH STRING
(OSTETH IH) / SELECT GREAT STRING
(ASSTRG IR L1 IS IE DT1 0) / FIT LINE TO STRINGS
(AGSTRG IR L1 L2 DT2 0)
(ASLINE IR IH ID L2 L3 LNG 0)
```

写真1の状態のデータにこのプログラム

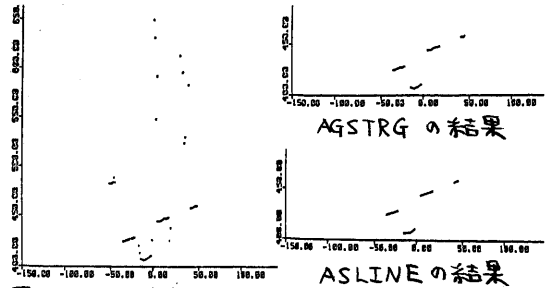


図7 スカ距離データ

4を適用した例を図7に示す。

(3) 関数 FIND の適用例

円柱、三角柱、四角柱の存在する環境で四角柱を検出する場合を例にとる。最も単純なプログラムは次のようになる。

```
(DEFVIEW 'BOX1 '(CORNER1 LINE1)) / BOX1'S APPEARANCE
(SETPARA 'BOX1 'CORNER1 'EDGE '(2500 2500))
(SETPARA 'BOX1 'CORNER1 'ANGL 9000)
(SETPARA 'BOX1 'LINE1 'EDGE '(2500 2500))

(TAKED 1)
(FIND 'BOX1)
```

処理経過を図8に示す。この例では BOX1 は CORNER1 として検出されている。コーナの検出はまず線分がありはめを行い、次に隣接する線分間の角度を調べて行われる。条件に合致するプロフィールが検出されると図9のようなリストで登録される。物体相互のオクルージョン

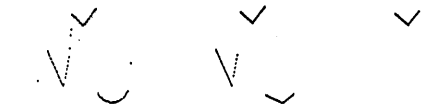


図8 直角のコーナ型プロフィールの検出

この状態によってはこのように単純なプログラムではうまくいかないが、そのような場合には視点の移動や腕による前方物体の除去等も必要となる。

(4) 関数 MEASURE の適用例

腕を用いた物体の取り上げ作業等では、物体の座標や大きさの情報が必要

図9

プロフィール
リストの要素
item 例

Line List Item	Pointer	Corner List Item	Pointer
0	1	0	1
1	2	1	2
2	3	2	3
3	4	3	4
4	5	4	5
5	6	5	6
6	7	6	7
7	8	7	8
8	9	8	9
9	10	9	10

となる。次の例では円柱を定義して検出し、中心位置と半径を出力させている。FIND に失敗した時はNILが出力される。図10の例では中心座標(-1.8, 450.7), 半径(17.4)(mm)が抽出結果として円弧のリストに登録されている。

```
(DEFVIEW 'CYLIN 'ARC)
(SETPAR 'CYLIN 'RADI 2000) / CYLIN'S APPEARANCE

(TAKEPD 1)
(FIND 'CYLIN)
(UAR CEN RAD)
(SETQ CEN (MEASURE 'CYLIN 'CENTER))
(SETQ RAD (MEASURE 'CYLIN 'RADIUS))
(PRINT CEN)
(PRINT RAD)
```

図11 円柱位置の測定

円弧リストの内容

```
NO PT TYPE IS IE      -1.8 450.7 17.4 -150.9 -15.7 9.088
1 22 3 111 143
      ×座標 Y座標 半径
```



以上がこれまでに開発したRVLを使用した例である。複合物体処理機能については現在インポート中であるが、PETLの再帰的使用性を用いて、プロフィール記述構造と柔軟に実現可能であり、また要素的機能については実験の結果は「満足すべき結果を得たので」、RVLの基本仕様の有効性は立証できたものと考えている。

8. むすび

距離プロフィールを対象とするロボットビジョン言語(RVL)の基本的な考えと仕様を示し、実験によりその有効性を確認した。RVLの特徴は次の点にまとめられる。

- ① 3次元物体を扱う照合レベルの画像理解記述言語である。
- ② 対象物のモデル記述能力、モデルを利用した対象物照合記述能力、画像解析記述能力を有している。
- ③ 光切断法に基づく距離プロフィールを処理の基本としこのため、記述の簡略化による処理効率の向上とユーザにとっての直観的理解の容易さが保持された。

RVLは現在まだ開発途中であり、今後の課題として

- ① 複雑な物体のモデル化機能を強化し有効性を実証すること、
- ② カメラの位置移動に対応する機能を扱えるようにすること、
- ③ データベース機能との結合を図ること等が考えられ、また実際のハンド=アイシステムに適用することにより使用性を高めて行く必要がある。

謝辞 本研究の機会を与えられた若松清司制御部長、赤坂 寛システム制御研究室長に謝意を表する。また、末広尚士技官には、同氏作製のロボット言語用プログラムの一部をVMDVのために使用させてもらったことを感謝する。

参考文献

- 1) 松下電: 距離プロフィール処理に基づく能動的なイメージングシステム 昭57年第2回SICE学術講演会, P346-367 (1982).
- 2) Kuiper, K.: PICASSO, PICASSO-SHOW and PAL-A Development of a High-level Software System for Image Processing, in Languages and Architectures for Image Processing, New York: Academic Press (1981).
- 3) Kohler, R.R., et al.: The VISIONS Image Operating System, Proceed. 6th Int. Conf. Pattern Recog., P11-74 (1982).
- 4) Maggiolo-Schettini, A.: Comparing some High-level languages for Image Processing, in the same book to 2) (1981).
- 5) 森本: 画像解析言語 PILS, 情報学論文誌, Vol. 23 P243-250 (1982).
- 6) Franklin, J.W., et al.: Programming Vision and Robotics Systems with RAIL, ROBOTICS VI Conf. Proceed. P 392-406 (1982).
- 7) Carlisle, B., et al.: THE PUMA / VS-100 ROBOT VISION SYSTEM, Proceed. 1st Int. Conf. on Robot Vision and Sensory Controls, P149-160 (1982).
- 8) Chang, M.S., et al.: Picture Query Languages for Pictorial Data-base Systems, COMPUTER Nov. 1981, P23-33 (1981).
- 9) 森本: PETL システム説明書-基本システム編-, 電研研 (1982).

(付録) OBJECT記述の仕様。PETLの属性リストを利用している。下位のインデックスにより上位のインデックスと右側に示したデータを登録する。

** Object Description **

```
(user specification)
OBJECT                                string
OBJECT.NO*                           integer
OBJECT.FRAME                          frame
OBJECT.VIEW                           list
      VIEW.PARTS                      list
        PART.NO*                     integer
        PART.NAME*                   string
        PART.TYPE                     appear_type
          TYPE.PARAMETERS             integer
            PARAMETER.MARGIN          integer
            PART.SPACE_RELATION       trans
      VIEW.PRIORITY                   list

(effect of FIND)
VIEW.LOCATION                         vector
VIEW.MATCH                           list
VIEW.RELIABLE                        integer
VIEW.PROFILE                         profile tablet
```

* optional attribute