SURVEY OF ADVANCED GENERAL-PURPOSE SOFTWARE FOR ROBOT MANIPULATORS

J.C. LATOMBE,
IMAG, BP 53X, 38041 GRENOBLE Cedex, FRANCE

ABSTRACT

Computer-controlled sensor-based robots will become more and more common in industry. This paper attempts to survey the main trends of the development of advanced general-purpose software for robot manipulators. It is intended to make clear that robots are not only mechanical devices. They are truly programmable machines, and their programming, which occurs in an imperfectly modelled world, is somewhat different from conventional computer programming.

INTRODUCTION

In the early 60's, a few Artificial Intelligence laboratories (MIT, Stanford, Edinburgh) started "hand-eye" projects. These projects concerned robots as general-purpose programmable machines able to perceive their environment and to decide their actions. Although high cost, low reliability and poor efficiency restrained these machines from practical application, an important concept, which may greatly affect our future, was technically born.

During the last twenty years, industry has developped flexible machines known as industrial robots. Although most of these machines would not have been called "robots" by the tenants of the hand-eye projects, they have considerably evolved under socio-economic pressure and rapid progress of basic technology. It has become more widely recognized that robots with better sensing result in cheaper environments and that more competent robots facilitate automation of more complicated tasks.

Today, the dream of some of the early researchers in Artificial Intelligence becomes a reality. There are more and more sophisticated computer-controlled robots operating on a wide variety of tasks such as painting, welding and assembly. Although many of them still do not have advanced sensing capabilities, such as touch and vision, the use of sensor-based robots is increasing. In addition new applications outside manifacturing, in areas like undersea work and agriculture, are being considered, and they push for new progress.

All these developments raise many crucial software problems :

- Programming robots, which was very easy when tasks were as simple as pick and place, becomes an important issue. How to describe an assembly task to one or several robots ?
- Visual sensors provide an enormous amount of potentially useful data. How to interpret digitized images in terms of the real world ?

Computer programming which was non-existent in the early age of industrial robotics becomes one of the central issues of modern robots.

This paper surveys recent progress in robot advanced general-purpose software. It focuses on robot manipulators and does not directly address mobile robots. In addition, issues in "low-level" software, like servoing, trajectory computation, coordinate transforms and image processing are not treated.

The paper is divided into three main sections : planning, programming and vision. Although these sections complete each other, they may be read in any order.

I ROBOT PLANNING

Planning tasks

Robot planning is the activity that underlies the mental construction of sequences of actions (plans) to achieve goals. Programs performing this kind of activity are called planners.

Robot planning has been a very popular research area during the 70's. Many planners have been designed during that period including STRIPS [Fikes 71], ABSTRIPS [Sacerdoti 73], LAWALY [Siklossy 73], BUILD [Fahlman 74], NOAH [Sacerdoti 75], HACKER [Sussman 75] and NONLIN [Tate 77].

Some typical planning tasks considered by researchers are formulated in a simple abstract world, the "blocks world", which has the following semantics :

- it consists of several moveable blocks and a fixed table,
- when a block is not moved, it rests either on the table, or on top of an other block,
- there cannot be more than one block directly on top of an other,
- the table is large enough so that all the blocks can directly rest on it,
- a robot can be used to move one block (and only one) at a time.

Within this world, a planning task is to generate a sequence of block moves that will hopefully transform an initial configuration of the world into a final one (figure 1).

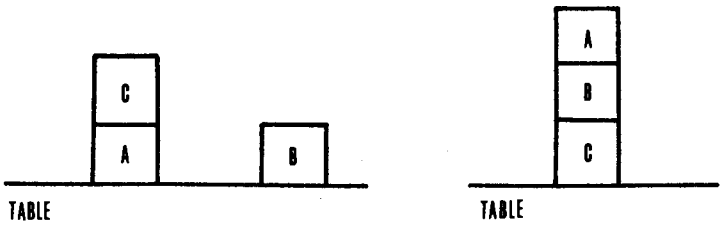


Figure 1 : A manipulation task

Planner's knowledge

Clearly a planner must have some knowledge about the world, the goal of the task and the robot's possible actions. A classical representation, which has influenced most of the works in the area, is the following one [Fikes 71] [Nilsson 80] :

- A configuration of the world is modelled by a conjunctive set of variable-free predicate atoms. Such a set is called a state. The initial state describing the initial configuration of the world is given to the planner (figure 2).

{ON(A,TABLE), ON(C,A), CLEAR(C), ON(B,TABLE), CLEAR(B)}

Figure 2 : The initial state for the task of Figure 1

- In addition to the initial state, the task is specified by a goal formulated as a set of atoms (figure 3). The goal is achieved in a state S if it can be unified with a subset of S.

{ON(B,C), ON(A,B)}

Figure 3 : The goal of the task of Figure 1

- Each possible action of the robot is represented by an operator made of three parts : the precondition, the delete list, and the add list. Each one is a set of atoms.

An operator F = (P,D,A) is applicable to a state S if its precondition P is achieved in S. If it is so, applying F to S produces a new state S' = (S-D)UA. Usually the operator contains variables (parameters) that are instantiated by the substitution unifying P with a subset of S.

Figure 4 shows operators for the blocs world. MOVE(x,y,z) -read "move x from y to z"- is applicable to the state of figure 2 with x=C, y=A and z=B. It transforms this state into the following one :

{ON(A,TABLE), CLEAR(A), ON(B,TABLE), ON(C,B), CLEAR(C)}.

Note that, since the table is always "clear" (it is large enough to support all the blocks), we must consider separately the cases when y=TABLE and z=TABLE. The special predicate DIFF is used to differentiate among the cases.

```
MOVE(x,y,z)
 P = {DIFF(y,TABLE), DIFF(z,TABLE), CLEAR(x), CLEAR(z), ON(x,y)}
 D = {ON(x,y), CLEAR(z)}
 A = {ON(x,z), CLEAR(y)}

MOVE(x,TABLE,z)
 P = {DIFF(z,TABLE), CLEAR(x,) CLEAR(z), ON(x,TABLE)}
 D = {ON(x,TABLE), CLEAR(z)}
 A = {ON(x,z)}

MOVE(x,y,TABLE)
 P = {DIFF(y,TABLE), CLEAR(x), ON(x,y)}
 D = {ON(x,y)}
 A = {ON(x,TABLE), CLEAR(y)}
```

Figure 4 : Operators for the blocks world

Given the knowledge of Figures 2, 3 and 4, a plan for the task of figure 1 is:

[MOVE(C,A,TABLE); MOVE(B,TABLE,C); MOVE(A,TABLE,B)].

Obviously there are many different ways to axiomatize knowledge in a planner.

Plan generation

Plan generation can be regarded as a graph searching problem [Nilsson 71]. The initial state and the operators define an implicit graph whose nodes and arcs are labelled by states and instantiated operators respectively. A solution plan is a path from the initial state to a state in which the goal is achieved.

However such a crude approach is subject to the combinatorial explosion whenever the plans to be generated contain many actions. This is why most of the work in robot planning has been aimed at finding good strategies for refining the search space.

A well-known strategy is "means-ends analysis" [Ernst 69] [Fikes 71].

In principle it works as follows :

(1) Initialize STATE to the initial state, STACK to the goal (STACK is a stack), and PLAN to the empty list.

(2) Set G to the goal on top of STACK.

(3) If G is achieved in STATE then :
- Remove G from STACK.
- If STACK is empty, then G is the goal of the task. The problem is solved and the solution plan is contained in PLAN. Exit.
- If STACK is not empty, then G is the precondition of an operator F. Append F to PLAN, apply F to STATE and store the new state in STATE, and return to step (2).

(4) If G is not achieved in STATE then :
- Select a subgoal (atom) SG of G that is not already achieved in STATE.
- Select an operator F the add list of which contains an atom unifiable with SG.
- Record the precondition of F on top of STACK as a new goal.
- Return to step (2).

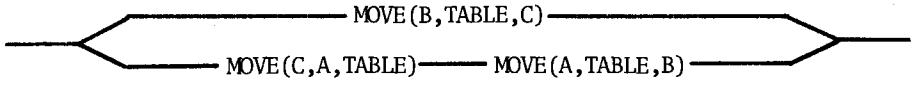
In addition the procedure includes a backtracking mechanism to deal with the alternatives that may occur at step (4).

Means-ends analysis faces combinatorial explosion by mixing top-down (goal-driven) reasoning -- operators are selected only if they are relevant to achieve the current goal -- and bottom-up reasoning -- selected operators are applied as soon as they are applicable. However the strategy is not complete in the sense that it does not produce a solution plan whenever one such plan exists. Indeed, it is not able to deal with interacting subgoals.

Interacting subgoals

Let consider the task defined by Figures 2, 3 and 4. The means-ends analysis strategy considers the two subgoals ON(B,C) and ON(A,B) in sequence. Suppose it attempts to achieve ON(B,C) first. When it tries to achieve ON(A,B), the current value of PLAN will be [MOVE(B,TABLE,C)] . To attain ON(A,B) more actions will then be appended to this plan and we will get a plan like the following :
[MOVE(B,TABLE,C) ; MOVE(B,C,TABLE) ; MOVE(C,A,TABLE) ; MOVE(A,TABLE,B)]
which clearly does not achieve the conjunctive initial goal. ON(B,C) and ON(A,B) are said to be interacting subgoals because they cannot be solved separately.

A method to deal with interacting goals, which has attracted considerable interest, is non-linear planning [Sacerdoti 75] [Tate 77]. With such a method we would attempt to achieve ON(B,C) and ON(A,B) in parallel and we would obtain a plan like the following :



This plan contains potential conflicts between parallel actions :

- MOVE(B,TABLE,C) undoes a precondition of MOVE(C,A,TABLE),
- MOVE(A,TABLE,B) undoes a precondition of MOVE(B,TABLE,C).

These conflicts are solved by constraining the ordering on the actions.

Another way to face interacting subgoals is to use regression techniques [Waldinger 75] [Nilsson 80]. On our example, the techniques basically work as follows :

- A plan to achieve one of the two subgoals, let say ON(A,B), is generated first :

[MOVE(C,A,TABLE); MOVE(A,TABLE,B)].

- Then we try to achieve ON(B,C) by appending new actions to this plans, without violating ON(A,B). Clearly, that is impossible and we compute the weakest precondition that should be achieved immediately before MOVE(A,TABLE,B) in order to guarantee that ON(B,C) will be achieved immediately after. Here this precondition is trivially ON(B,C).
- Finally we achieve the precondition ON(B,C) by inserting MOVE(C,TABLE,C) before MOVE(A,TABLE,B) while checking that CLEAR(A), a precondition of MOVE(A,TABLE,B) established by MOVE(C,A,TABLE), is not violated by this new action.

Neither non-linear planning, nor regression completely solve all the difficulties introduced by interacting subgoals. However, some powerful planners based on such techniques have existed for a while [Sacerdoti 75] [Tate 77]. A recent paper [Wilkins 82b] brings some improvements to the previous non-linear planning techniques.

Current research

After slowing down during the late 70's, research on robot planning seems to become active again. Some of the new research topics involve :

- planning for several agents and distributed planning [Rosenschein 82] [Konolidge 80],
- resources sharing [Wilkins 82a] [Wilkins 82b],
- taking time into consideration [McDermott 82].

Application of robot planning

Most of the research in AI on robot planning has focused on methods for producing intelligent behavior. As a result the planners work in highly stylized worlds with simple well-defined semantics. They ignore such issues as geometry, position uncertainty, object touching, which are critical in manipulator control.

To a limited extend BUILD [Fahlman 74] considered contact, gravity and force notions. However it was reported that most of the programming effort (80 %) was devoted to coding programs dealing with these notions, rather than implementing general-purpose planning methods.

STRIPS [Fikes 71] is one of the very few planners that controlled a real robot. Again it required a large engineering effort to write the lower-level programs implementing the set of actions considered by STRIPS.

Such experiences suggest that, to be applicable to practical manipulation tasks, the logic-based methods sketched in the previous sections need to be completed by what we may call a "general-purpose intermediate-level software" dealing with geometry, collision, ... This is precisely what some of the current research reported in the next main section tends to achieve.

When this software is available, an important application of planning will probably be the control of groups of robots cooperating on several simultaneous tasks. The planners will have to decide of the subtasks (including information gathering) sequencing and allocation. They will also have to consider resource (parts, tools, sensors) sharing. Some work in this direction has already been done [Vernet 80]. Similar applications in domains close to robotics have also been worked out with some success [Descotte 81] [Bullers 80]. In particular, [Descotte 81] describes a system that generates plans for machining mechanical parts. These plans define the sequence of cuts to be performed and the machines to be used for the cuts.

It has also been suggested that planners could be useful for emergency recovery when a failure occurs in the execution of a task [Gini 75].

Non-robotics applications of planning methods include air fleet control [Thorndyke 81] and DNA sequencing [Stefik 80].

II ROBOT PROGRAMMING

Interactive guiding

A common method for programming industrial robots is the "teaching" or "interactive guiding" method. Using an interaction device such as a joystick or a button box, the programmer teleoperates the manipulator to perform a sample execution of the task. The motions of the joints are recorded so that they can automatically be played back later.

This method requires almost no specialized knowledge from the user. But it presents severe drawbacks that restrict its application to simple tasks :

- Repetition of the same sequence of motions with minor changes at each iteration (e.g. palettizing) is fastidious to "program",
- Recorded "programs" are difficult to edit,
- Sensory data cannot be used to face uncertainties of the environment,
- Coordinating several robots is impossible.

Interactive guiding applies repetability constraints to the task, the robot and the environment, which restrict its use to rather simple tasks such as spray painting and spot welding. Achieving these constraints on more complex tasks is often costly or impossible. For that reason methods based on the use of symbolic languages for programming robots receive more and more attention.

Robot programming languages

A robot system provides primitive mechanical and sensory capabilities which implicity determine the task domain to which it can be applied. During the last ten years, these capabilities have improved impressively to become applicable to a wide range of complex tasks including mechanical assembly [Salmon 77]. Simultaneously there has been an increasing need for an appropriate formalism for specifying how they are to be applied to the tasks. Robot programming languages have been designed since the mid 70's to satisfy this need [Latombe 79].

Most robot programming languages are extensions of classical languages like BASIC, FORTRAN and ALGOL. They include the basic data structures and instruction set one uses to find in these languages. In addition they provide new constructs suitable for specifying robot tasks. The most important ones deal with world modelling, end-effector motions and operations, sensor data gathering, synchronization of several robots. Some of them are illustrated on the procedure shown at Figure 5. Actually this program is written in LM [Latombe 81], but similar programs could have been written in other languages.

The geometry of the world is sketched by cartesian frames attached to the objects of the task. Several frames are a priori known by the language interpretor. They include a fixed frame, which serves as the user's reference frame, and moving frames attached to the robots end-effectors. In the programs, one can define new frames by applying transforms (combinations of rotations and translations) to already known frames. The transforms are described using vectors. The language includes frames, transforms and vectors as basic data types, and provides operations to work on them : cross-product of two vectors, extraction of the transform between two frames, etc.

```
FUNCTION BOOLEAN INSERT(FRAME PIN, HOLE; REAL FZMAX, EPS; INTEGER NMAX);
INTEGER N; VECTOR LATF; FRAME HOLE1;
BEGIN
   N:=0; HOLE1:=HOLE:TRANSLATION(-VZ,10);
   WHILE N<NMAX DO
      MOVE PIN BY TRANSF(PIN,HOLE1) UNTIL FZ>FZMAX;
      IF DISTANCE(PIN,HOLE)<1
         THEN RETURN(TRUE)
         ELSE
            LATF:=VECTOR(FX,FY,0.);
            MOVE PIN BY TRANSLATION(LATF,EPS);
            EPS:=EPS/2; N:=N+1;
      ENDIF;
   ENDDO;
   WRITE "INSERTION FAILED"; RETURN(FALSE);
END;
```

Figure 5 : A robot program to insert a pin into a hole

Frames can be affixed to other frames. Two affixed frames remain in the same relative position in the interpretor's model of the world. If the position of one frame is known to be modified, let say by a motion statement, then the position of the other frame is automatically updated in the world model. Such a facility clearly simplifies programming when several frames are attached to the same object. For example, a frame F attached to an object may be convenient to define the position of the object, while other frames Gi are more suitable to model possible grasping positions of the object (that is where the frame attached to the end-effector should be when the object is to be grasped). If F and Gi are affixed together, then they will "move together". Affixment also allows the description of motions directly in terms of object frames. A more clever use of this facility deals with the automatic correction of the position of an object when contact information is obtained [Taylor 76].

Each motion statement applies to the frame M which the user considers as the most convenient to describe the motion. M must be affixed to the frame attached to the end-effector of a robot. The final destination is defined by a frame D. M will coincide with D at the end of the motion. To attain this goal position, the origin of M is translated along a straight path. In addition if M and D do not have the same orientation, the axes of M are rotated around a fixed direction. The interpretor does all the computations, involving trajectories planning and generation, and cartesian to joint coordinates transformations, and generates the servo set points [Paul 81]. The user can control the speed of the motion. He also can describe non-linear trajectories by stipulating that M should traverse some intermediate positions before it reaches its final destination.

A motion may be guarded by a boolean condition involving sensory data such as exerted forces. This condition is periodically evaluated during the motion and the robot is stopped when the condition becomes true. Languages like AL [Finkel 76] and PAL [Paul 80] also include options to control both the position and the forces exerted by the robot during motion, for instance :

```
MOVE OBJECT TO HERE WITH FORCE(VY) = YBIAS.
```

Synchronization of several robots is offered by a few languages only. For instance AL includes a COBEGIN-COEND construct. LM permits the user to get the control back to its program when a robot is moving.

Current status

The earliest robot language was WAVE [Paul 77] at the Artificial Intelligence Laboratory of Stanford. Since then many (too many?) other languages have been developped including AL [Finkel 76], LM [Will 75], EMILY [Will 78], SIGLA, LM [Latombe 81], ROBEX [Weck 81], VAL, AML, RAIL, LAMA-S [Falek 80], MAL [Gini 79] and PAL [Paul 80]. Several of them are now commercially available : SIGLA (Olivetti), VAL (Unimation), AML (IBM), RAIL (Automatix), LM (SCEMI). They have been used on a wide variety of manipulation tasks including arc welding and mechanical assembly [Latombe 82a] [Paul 77].

Aids for developping and debugging robot programs have also been implemented. POINTY [Gini 78] is an interactive tool for constructing AL programs using guiding techniques. Graphic simulators have been developped as off-line debugging tools [Soroka 80] [Sata 81]. They are becoming popular in firms where one can take advantage of existing CAD facilities [Querones 82] (figure 6). Recent languages, like RAPT [Popplestone 78] and LM-Geo (an extension of LM) [Mazer 1982], also make use of geometric models of objects to permit the users to define positions of frames in terms of symbolic geometric relations rather than by quantitative transforms.

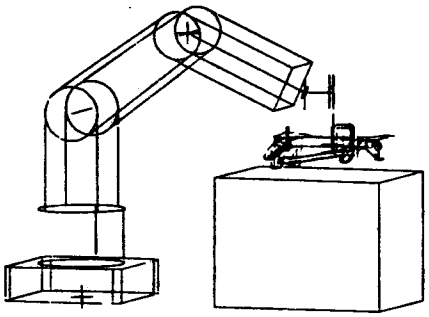


Figure 6 : A graphic display of a robot at work [Querones 82]

The availability of robot programming languages at a commercial level is probably the most significant aspect of the current evolution of industrial robots. It enlights the growing importance of sophisticated software for robotics.

Task-level languages

The programming languages introduced above are often called manipulation-level languages. Although they represent an important breakthrough in industrial robotics, they are not quite easy to use. The programmer must define all the motions to make and prevent possible collisions. He also has to face various kinds of uncertainties requiring to monitor sensory conditions. These drawbacks clearly are inherent to the level of description of the tasks.

More recent research has focused on task-level programming systems in the domain of mechanical assembly. A task-level specification consists of two parts :

(1) a sequence of assembly relations to achieve, such as inserting a pin into a hole and fitting an object onto an other,

(2) a data base of object models, which may have been provided by a CAD system.

The goal is to automatically translate this specification into a manipulation-level program [Latombe 79].

In addition to making users programmation easier, task-level systems have other potential interests :
- they can simplify the interfacing of manipulation and vision through the object data base,
- AI methods for robot planning may be applied to generate task-level programs for complex tasks involving several robots.

Some years ago two task-level systems, AUTOPASS [Lieberman 77] and LAMA [Lozano-Perez 76], have been defined, but they hardly have been implemented. The initial report on the AL language [Finkel 74] also included a task-level subset of instructions which was never implemented. Although they make use of explicit geometric models, languages like RAPT [Popplestone 78] and LM-Geo [Mazer 1982] are basically manipulation-level languages.

In fact translating task-level programs into manipulation-level programs turns out to be a very difficult problem involving geometric, strategic and physical interweaving ramifications. The trend in current research is to solve more delimitated subproblems, typically path finding, grasping and fine motion generation. Enough results have been obtained so far to predict that true task-level systems will be operating in some laboratories before long.

The find-path problem

The find-path problem can be stated as follows [Brooks 82] : given an object with an initial position, a goal position, and a set of obstacles located in space, the problem is to find a continuous path for the object from the initial position to the goal position which avoids collisions with obstacles.

Clearly this is a central problem that has to be solved not only for manipulation robots, but also for mobile robots. However, for mobile ones, it usually can be approximated as a bidimensional problem [Giralt 79] [Moravec 79] [Chatila 81].

Until recently most of the works have focused on algorithms useful for computing collisions among solids [Boyse 79] [Ahuja 80] [Powell 82]. Various techniques have been devised (2D projections, decomposition of volumes into cells, parallepipedic and spherical approximations ...), but they do not directly solve the more difficult problem of finding paths. However in uncluttered environments these techniques can probably be used to check that trajectories generated by using crude heuristics are collision-free.

A relatively common approach to the find-path problem is the configuration space approach. It has been first reported in [Udupa 77] and much developped in [Lozano-Perez 79] [Lozano-Perez 80a] [Lozano-Perez 80b].

Basically it consists of shrinking the moving object (let call it A) to a point while expanding the obstacles (let call them Bi, i=1, ...) inversely to the shape of the object. This principle can be easily explained in two dimensions with a fixed orientation of A. Then the position of some reference point P of A is sufficient to specify the position of A. The obstacles Bi restrain P to be outside some regions Bi' (Figure 7). The Bi' can easily be computed if both A and the Bi are convex polygons. Then finding a path for A is equivalent to finding a path for P among the Bi'. This new problem can be solved by traditional searching method. The graph to be searched is formed by connecting all pairs of Bi' vertices

that can be connected by a straight line not intersecting any of the Bi'.

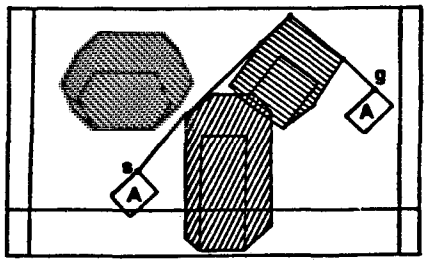


Figure 7 : Expanding obstacles in the 2D world [Lozano-Perez 80a]

The method can easily be extended to handle a moving object and obstacles approximated by sets of possibly overlapping polygons. Its generalization to problems involving three dimensional polyhedra and rotation of the moving object is far more complicated [Lozano-Perez 79] [Lozano-Perez 80a].

[Brooks 82] recently proposed another approach to the find-path problem based on a representation of the free space using "generalized cones". It works fast in relatively uncluttered environments.

Automatic grasping

The problem here is to find out how to grasp an object with a robot "hand". It includes the find-path problem as a subproblem, but the goal position of the hand on the object, i.e. the grasping position, is now to be determined. This requires to take into account accessibility constraints applied by both the object to be grasped and the task environment ; for instance, a pin to be inserted should not be grasped lengthwise. In addition the object should be stable in the hand.

Several researchers have attempted to automate grasping. One approach [Lozano-Perez 76] has been to consider a small number of grasping position classes defined on simple elementary volumes like rectangular parallepipeds and cylinders (Figure 8). Another one [Wingham 77] is to search for parallel faces in a polyhedron. Below we present a somewhat more complete approach defined in [Laugier 81] and [Pertin 82].

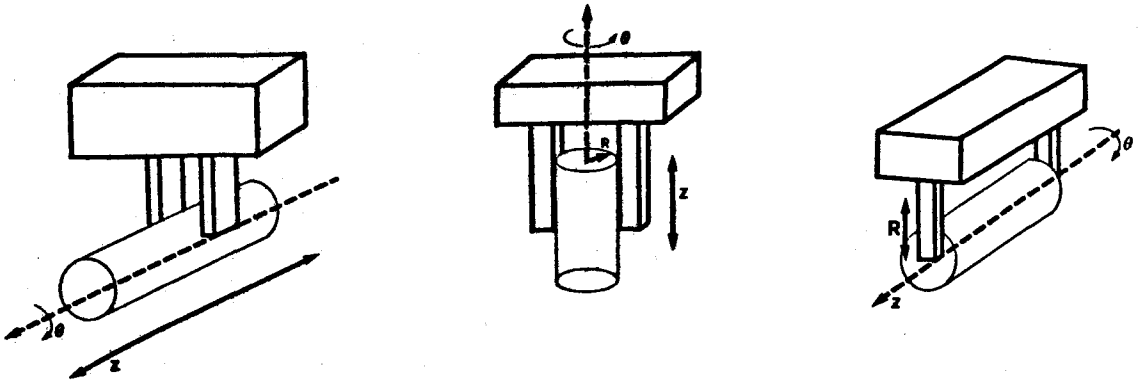


Figure 8 : Grasping classes on a cylinder [Lozano-Perez 76]

This approach brakes the problem down into two steps :

a) Initially each pair of object faces, edges and vertices is regarded as a potential basis for grasping (the hand is supposed to be made of two parallel jaws). Simple local properties of the object are used to determine a small subset of grasping pairs.
For instance :
- Any pair containing a concave edge is rejected.
- Any pair made of non-parallel faces is discarded.
- Any pair made of faces which are not "in front of" each other is rejected.
- Etc...

b) During the second step, a grasping position is computed by analyzing global accessibility constraints. The grasping pairs are ordered according to the potential stability of the object in the hand, for instance : parallel plane faces provide more secure grasping conditions than a face and a vertex, a "good" grasping pair should permit the center of mass of the object to be between the grippers axis. These heuristics are usually sufficient when the objects are small with respect to the hand.
The ordered grasping pairs are considered in turn until a grasping position has been computed. This computation is based on a method very similar to the configuration space approach introduced in the previous section. To simplify the application of the method the hand is supposed to move along a straight path. Heuristics using the values of the local properties computed in the first step are used to suggest tentative paths.

Another very different approach to automatic grasping has been proposed by [Boissonat 81]. It does not use a prior geometric model of the object, but relies on a 3D visual sensor to extract object geometrical patterns appropriate for grasping. This approach is more suited for bin picking than for automatic programming.

Fine motion synthesis

An important operation in assembly tasks is part mating. It is a tight-tolerance operation which is made difficult by several sources of imprecision :

- there are bounds to the accuracy of a robot manipulator,
- copies of the same object are identical only within specified tolerances,
- feeders deliver objects with some position uncertainty.

Therefore, part mating requires several sensor-based corrective motions. For instance when inserting a pin into a hole, the pin may first hit the hole chamfer. This contact can be detected by using a force sensor and a corrective motion then consists of displacing the pin along the lateral force on a short distance (see program of Figure 5). Fine motion generation is a difficult part of robot programming. Some mechanical devices known as compliant wrists avoid the need of programming fine motions. However they are very specific-purpose tools.

Until now there have been very few works aimed at synthesizing fine motions programs automatically. [Taylor 76] presents an approach based on representing accuracy information in a world model data-base, and a numerical methods to propagate location constraints through relations among object features. A very different approach based on inductive learning is described in [Latombe 82a] and [Dufay 82]. A system has been developped to infer programs

from execution traces. This system makes use of user-input rules to generate sample executions of the task. These rules are of two kinds : planning rules, which associate situation and goal conditions to motions to be performed, and scene analysis rules, which permit the system to infer the current situation from sensor conditions. Each execution trace is a sequence alternating states and motions. The generated traces are merged into a finite-state automaton by inductive transforms. Programs similar to that of Figure 5 have been synthesized by this system.

III. VISION IN ROBOTICS

Applications

The purpose of a vision system applied to robotics is to transform the rough data contained in -- a digitized image -- typically a 512x512 array of "pixels" -- into a description about world objects, which is to be used by programs that control the arm motions. Each pixel is usually an integer coding light intensity ("grey-level"). But it may also be a vector coding 3D coordinates or color information.

There are many potential applications of vision to robotics, including : object recognition and location, control of robot operations and inspection.

Vision in the blocks world

Most of the early works in computer vision have concerned scenes made of polyhedral objects (Figure 9). Historical presentations of these works are given in [Ballard 82] and [Cohen 82]. Some of the main results have been a theory for line drawing understanting and a constraint-propagating labelling algorithm.

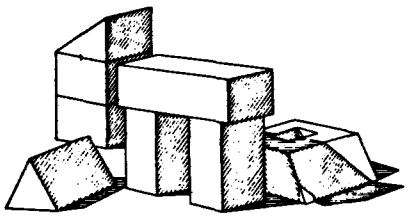


Figure 9 : Scene in the blocks world [Waltz 75]

Each line of a scene in the blocks world can be labelled by a + (convex edge), a - (concave edge), or by a > (occluding edge). The occluding matter is to the right of the line looking in the direction of the > (Figure 10). Each line can have only one label (i.e. no line may change its interpretation between vertices).

A systematic investigation [Huffman 71] [Clowes 71] [Waltz 75] showed that there are only a few types of possible line junctions, and that for each type of junction there are a few valid labellings. [Waltz 75] developped a constraint-propagating labelling algorithm which quickly eliminates incoherent labels. Initially the lines at each junction are given a set of labels depending on the type of the junction. Since a line can only have one label, a junction interpretation is rejected whenever it is incoherent with all the interpretations of an adjacent junction. Eliminating an interpretation at a junction may render a neighboring interpretation incoherent as well, ... In many cases this algorithm eliminates all the

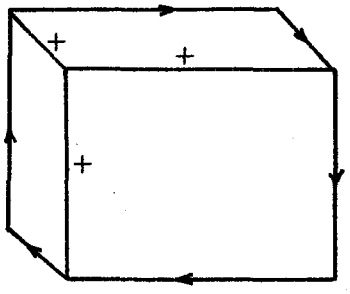


Figure 10 : A labelled polyhedron

impossible labellings except the correct one, so that search is usually not needed. Similar contraint-propagating algorithms, sometimes named relaxation algorithms, have been used in many computer vision systems.

Vision in the blocks world also contributed to develop the "gradient space" representation [Mackworth 73] [Shafer 82b].

Binary 2D vision

Since the mid 70's several works in robot vision put emphasis on effective cost, speed, reliability and simplicity of use. A number of general-purpose systems known as binary 2D vision systems have emerged during that period [Agin 75] [Yashida 77] [Horaud 81]. They are suited for recognizing and locating isolated objects resting on an horizontal surface.

These systems take a binary image of the scene by thresholding the video signal of a camera. Such an image is an array of bits. A connectivity algorithm groups the points into topologically connected "black" and "white" regions [Veillon 77]. The white regions are interpreted as the background, and the black ones as the objects (it could be the reverse). The values of some global properties of each main black region are computed, for instance : area, length of perimeter, number of holes, compactness. A recognition algorithm (nearest-neighborhood, decision tree) compares them with stored models and decides the identity of the corresponding object. Then the centroid of the region determines the object position. Other properties with respect to the centroid (maximal and minimal radius, position of holes centroids) are used to compute the orientation of the object. With appropriate electronic operators, these systems can identify and locate objects within fractions of seconds.

Object models can be constructed by a program-by-showing training method. Each object in each stable position is shown individually to the camera. The properties to be used at run time are computed and a decision tree is generated. Some simple statistics (e.g. standard deviation) may also be gathered in order to face variations of properties values over several images because of camera noise and quantization errors. [Lieberman 79] proposed a completely self-trainable system. Given a geometric model of an object, the system predicts its two-dimensional possible appearances. First it computes its convex hull. Then it determines all the stable configuration of the object on an horizontal surface. For each configuration, it computes the 2D vertical projection of the contour.

Since a few years, several systems based on the above principles have been commercialized by companies like MIC, Automatix and Brown-Boveri. This development is another very

important aspect of the current evolution towards polyvalent industrial robots.

However, binary 2D vision is based on very simple computations which require many constraints to be achieved in order to operate properly :

- A contrasting background is necessary, so that thresholding separates the objects from the background correctly.
- The camera must look straight down to the objects in order to minimize the change in appearance of the objects when the objects are moved on the horizontal surface.
- The lighting should avoid shadows and specular reflections that may produce unclassifiable regions.
- The objects should be isolated so that the properties of the regions in the image can be directly used to identify objects.

Although binary 2D vision systems can be very useful in a wide range of applications, there are still more numerous applications in which the above constraints are difficult or impossible to achieve.

Locating partially visible objects

The limitation of vision with binary images drove some researchers to use grey-level images. A typical application that has been considered is the location of partially visible objects.

A comprehensive work on this problem is described in [Perkins 77] [Perkins 78]. Like the methods introduced in the above section, it applies only to 2D scenes (objects features are supposed to keep the same visual appearance when objects overlap). The Hueckel edge operator is scanned over the image to extract edge points, which are then chained into lines. A curve fitting technique examines curvature to segment those lines into straight lines and circular arcs (Figure 11). Finally, a matching program compares the segmented lines extracted from the image with an object model in the same format to identify and locate the object in the scene.

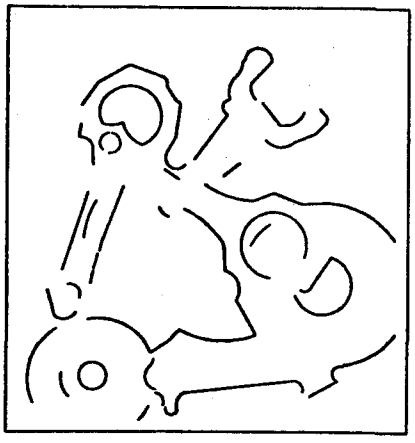


Figure 11 : Line segments in an image [Perkins 77]

The matching process is made of three successive steps :

- First, general features of the model and the image lines such as total length of radius of arcs, magnitude of total angular change,...,are compared to order line pairs according to their matching likelihood.
- Second, the pairs are examined in turn. The model line of the first pair is slid to find the best alignment with the image line (the two lines may not have the same length). A poor correlation results in rejecting the pair and trying the next one. A good one produces a tentative transformation (x, y, θ). In case of symetry (e.g. rotational symetry) of a line, several pairs may be used to establish the transform.
- Third, the transform is applied to all the model lines. A more global test is applied to validate the transform.

The method has been implemented successfully to locate objects (steering knuckles) in scenes with considerable occlusions and visually noisy conditions. Response time was rather long (several tens of seconds), but over half that time was spent in scanning for edge points. The efficiency could easily be improved by using special-purpose operators. Another way to improve such a methods is to guide scene analysis by a better control strategy such as prediction-verification.

Locating partially visible objects have also been performed using binary images by considering local features of the regions frontiers. [Bolles 80] describes a system using corners and small holes as local features, and a maximal clique method for matching. [Riad 81] proposes another system based on the use of features like straight lines and circular arcs with a syntactic method for matching.

Prediction-verification

The purpose of the prediction-verification strategy is to focus processing on hopefully relevant image regions. Basically, it works as follows :

(1) Hypotheses about the scene are constructed by mapping those features which have been extracted so far from the image into object models.
(2) The most plausible hypothesis is selected to predict the type and approximate location of new features in the image.
(3) Lower-level procedures are called to extract the predicted features. Success increases the plausibility of the hypothesis. Failure decreases this plausibility.

These three steps are executed iteratively until a hypothesis gains sufficient support. Failure in finding expected features at step (3) may cause to return to step (1).

PVV [Souvignier 81] [Latombe 82a] is a typical system based on this strategy. Like the systems described in the above section it works on 2D scenes made of overlapping objects. The goal of the vision task is to identify and locate a particular object. First a few segments of contrast lines (straight segments, circular arcs) are extracted. For instance, only points with high gradient values may be considered at that stage. Then, hypotheses about the possible locations of the object are generated by considering every pair of extracted segments and comparing them with pairs of segments of the object model. Each hypothesis is rated by a plausibility factor depending on the lengthof the segments which support the hypothesis. The best-rated one is considered, and the system looks for new expected features in appropriate

windows of the image. If the hypothesis gains plausibility above some level, then it is returned as the solution of the vision task. Otherwise, either new features are predicted and looked for (when a hypothesis keeps sufficient plausibility the system may take the risk of looking for features with lower contrast), or the hypothesis is abandonned. In order to deal with overlapping objects, the system can match several short image segments against a longer model segment.

This kind of strategy is typical of systems working with locally ambiguous data, for instance speech understanding systems [Woods 78]. With several variations, it has been implemented in a wide range of vision systems, including [Bolles 77], [Yashida 77], [Freuder 77], [Ballard 78], [Shirai 78], [Brooks 81] (these are only a few of them). The VV system [Bolles 77] is to be used in repetitive assembly tasks. The environment should remain very similar between two executions of the same task so that the system can have a great deal of prior knowledge about the scene. Its goal is to verify and refine the location of one or more objects in the scene, for instance gain more precise information about the relative displacement between a screw and the screw hole. VV uses a Bayesian estimation scheme to assign extracted image features (corner-like features abtained by local correlation techniques) to model features. It applies a least-square fitting method to combine the position information provided by feature assignments. [Yashida 77] describes a very simple use of the strategy to distinguish among objects of similar appearance by predicting discriminatory local features.

Prediction-verification is only one kind of strategy for controlling the vision process. Plan generation for anticipating cost-effective sequences of operators [Garvey 76] is another one, which probably have not received enough attention until now.

Surveys of computer vision systems applying control strategies can be found is [Binford 82] and [Latombe 82b].

3D vision

With a few exceptions, the systems introduced in the above three sections can only be used to interpret "two-dimensional" scenes. This does not mean that the objects have to be laminar. But the type of features considered by the systems and the relatively simple matching techniques they use require that each object may only be viewed under a finite number of angles.

3D computer vision concerns the use of explicit three-dimensional information in the image interpretation process [Shirai 79]. The goal is to devise systems with sufficient competence to analyze scenes made of objects in arbitrary position and orientation, with specular reflections and shadows. Obviously such a goal is relevant to both robot manipulators and mobile robots [Moravec 79].

Three-dimensional information about a scene can be obtained with very different methods, e.g. :

- by computing the surfaces orientations from a grey-level image using reflectance models (see for instance [Woodham 78]),
- by correlating two images provided by two cameras (stereo-vision) [Grimson 81] [Moravec 79],
- by finding out how three-dimensional models of objects can project on a 2D plane to produce lines similar to those appearing in a grey-level image [Brooks 81],

- by using a modulated laser beam and computing the elapsed time of the reflected light from the phase shift [Nitzan 77],
- by matching shadows contours with objects contours [Shafer 82].

In robotics, a very popular method to obtain 3D information on a scene has been "light striping" [Popplestone 75] [Latombe 82b] [Bolles 81b] [VanderBrug 79]. A plane of light is projected into the scene, which causes a stripe of light to appear on the objects (Figure 12). A camera is used to obtain an image of the stripe. The XYZ coordinates of each illuminated point are determined by intersecting a line (the line of sight corresponding to the image point) and a plane (the plane of light). The correspondance between the image points and the XYZ coordinates is established by a calibration procedure.

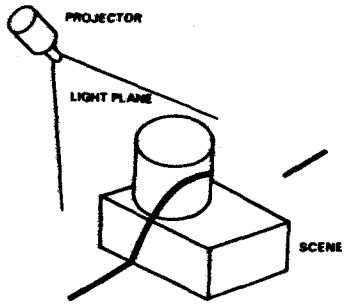


Figure 12 : Light striping [Popplestone 75]

An easy way to get a plane of light is to use a laser with a cylindrical lense [Mezin 82]. Then the camera is equipped with an appropriate filter and provide a clean image of the stripe. With some minor arrangements, a typical application of such a system is seam tracking for arc welding [Mezin 82] [Villers 82] [Bamba 81]. The vision sensor scans the area a few centimeters ahead of the weld and computes successive profiles of the seam weld. Important points of the profile can be obtained by a simple polygonal segmentation [Borianne 81].

In the above application one light stripe is processed at a time. There are many others which require to have a complete 3D map of a scene. This can be done by using a deflector to move the plane of light over the scene. Another way is to move the whole sensor or to move the scene (for instance the objects may rest on a conveyor belt).

A recent paper [Faugeras 82] presents a related vision system based on active stereovision. It uses two cameras to yield images from two different viewpoints. A laser beam overilluminates one point is the scene, which is seen by both cameras. The system has been experimented to gather 3D data about complicated industrial castings, and used to detect planar, cylindrical and conical parts of the object using the Hough transform [Ballard 82].

Thus, 3D vision systems based on active lighting are under development in many places. It is very likely that they will soon be implemented in industrial sites. Applications include seam tracking, automatic recognition of objects and volumic inspection.

Perspective

Although important progress has been done in robot vision during the last few years, most of the existing systems work in very restricted domains with carefully engineered environments. Augmenting the competence of these systems requires [Tenenbaum 79] :

- better low-level feature extraction,
- reasoning about several levels of representation of the scenes,
- more efficient control strategies based on prior knowledge.

Quicker and cheaper hardware pushes for such developments. In addition more basic research in image understanding is likely to be applied to robot vision soon [Bolles 81a].

An important area of robot vision will be the analysis of dynamic scenes. Until now it has attracted little attention within the robotics field. [Gennery 82] describes a method of visually tracking a known solid object determining its three-dimensional position and orientation rapidly as it moves. [Agin 77] explored the use of a camera mounted on the end-effector of a manipulator for bolt insertion. In fact, the problem with motion vision is that it requires more voluminous input than static vision. However basic research has been active for a while [Ullman 79] [Ballard 82].

Finally, robot vision systems being more complex, it will become crucial to develop tools to easily "program" them to accomplish the tasks. Until now the simple training methods used for binary 2D vision systems have not been extended to more complex systems.

CONCLUSION

This survey has presented some of the main trends in advanced software for robot manipulators. At this stage, it should be obvious that there has been considerable progress during the last few years. Several research products have become available in industry, for instance manipulation-level programming languages and binary 2D vision systems. New important problems, such as task-level languages interpretation, multi-robots planning and 3D vision are under active development in several laboratories.

This paper should also have made clear that robot manipulators are truly programmable machines, but that programming computer-controlled sensor-based robots is somewhat different from the mere programming of a computer. Robot programs run in a world which is incompletely known and imperfectly modelled. It requires strategies to deal with uncertainties, to detect and prevent potential catastrophes like collisions, to recover from errors. In fact, robot programs are solutions to a unique combination of geometrical, physical and strategic problems.

REFERENCES

IJCAI : International Joint Conference on Artificial Intelligence

AAAI : American Association for Artificial Intelligence

ISIR : International Symposium on Industrial Robots

[Agin 75]
G.J. Agin, R.O. Duda : "SRI vision research for advanced industrial automation"
2nd USA-JAPAN Computer Conference, 1975.

[Agin 77]
G.J. Agin : "Servoing with visual feedback"
SRI International, AIC, TN 149, July 1977.

[Ahuja 80]
N. Ahuja, R.T. Chien, N. Bridwell : "Interference detection and collision avoidance among three-dimensional objects"
1st AAAI Conference, Stanford, August 1980.

[Ballard 78]
D.H. Ballard, C.M. Brown, J.A. Feldman : "An approach to knowledge-directed image analysis"
In "Computer Vision Systems", Edited by A.P. Hanson and E.M. Riseman, Academic Press, 1978.

[Ballard 82]
D.H. Ballard, C.M. Brown : "Computer vision"
Prentice-Hall, 1982.

[Bamba 81]
T. Bamba et al. : "A visual sensor for arc-welding robots"
11th ISIR, Tokyo, October 1981.

[Binford 82]
T.O. Binford : "Survey of model-based image analysis systems"
The International Journal of Robotics Research, Vol. 1, n° 1, Spring 1982.

[Boissonnat 81]
J.D. Boissonnat, F. Germain : "A new approach to the problem of acquiring randomly oriented workpieces out of a bin"
7th IJCAI, Vancouver, August 1981.

[Bolles 77]
R.C. Bolles : "Verification vision for programmable assembly"
5th IJCAI, Cambridge, August 1977.

[Bolles 80]
R.C. Bolles : "Locating partially visible objects : the local feature method"
1st AAAI Conference, Stanford, August 1980.

[Bolles 81a]
R.C. Bolles : "An overview of applications of image understanding to industrial automation"
SRI International, AIC, TN 242, May 1981.

[Bolles 81b]
R.C. Bolles, J.H. Kremers, R.A. Cain : "A simple sensor to gather three-dimensional data"
SRI International, AIC, TN 249, July 1981.

[Borianne 81]
P.L. Borianne : "Perception tridimensionnelle : une étude de cas"
IMAG, Grenoble, Rapport de DEA, June 1981.

[Boyse 79]
J.W. Boyse : "Interference detection among solids and surfaces"
CACM, January 1979.

[Brooks 81]
R. Brooks : "Symbolic reasoning among 3-dimensional models and 2-dimensional images"
Artificial Intelligence, 1981.

[Brooks 82]
R.A. Brooks : "Solving the find-path problem by good representation of free space"
2nd AAAI Conference, Carnegie-Mellon University, August 1982.

[Bullers 80]
W.I. Bullers, S.Y. Nof, A.B. Whinston : "Artificial Intelligence in manufacturing planning and control"
AIIE Transactions, Vol. 12, n° 4, December 1980.

[Chatila 81]
R. Chatila : "Système de navigation pour un robot mobile autonome : modélisation et processus décisionnels"
Thèse de Docteur-Ingénieur, Toulouse, July 1981.

[Clowes 71]
M.B. Clowes : "On seeing things"
Artificial Intelligence, 1971.

[Cohen 82]
P.R. Cohen, E.A. Feigenbaum : "The Handbook of Artificial Intelligence"
(Volume 3) William Kaufmann, Inc. 1982.

[Descotte 81]
Y. Descotte, J.C. Latombe : "GARI : a problem-solver that plans how to machine mechanical parts"
7th IJCAI, Vancouver, August 1981.

[Dufay 82]
B. Dufay : "Inductive learning for fine motion synthesis"
IMAG, Grenoble (forthcoming).

[Ernst 69]
G. Ernst, A. Newell : "GPS : a case study in general problem solving"
Academic Press, 1969.

[Fahlman 74]
S. Fahlman : "A planning system for robot construction tasks"
Artificial Intelligence, Vol. 5, 1974.

[Falek 80]
D. Falek, M. Parent : "An evolutive language for an intelligent robot"
The Industrial Robot, September 1980.

[Faugeras 82]
O.D. Faugeras et al. : "Toward a flexible vision system"
12th ISIR, Paris, June 1982.

[Fikes 71]
R.E. Fikes, N.J. Nilsson : "STRIPS : a new approach to the application of theorem proving to problem solving"
Artificial Intelligence, 1971.

[Finkel 74]
R.A. Finkel et al. : "AL, a programming system for automation"
Artificial Intelligence Lab., Stanford, Memo AIM-243, November 1974.

[Finkel 76]
R.A. Finkel : "Constructing and debugging manipulator programs"
Artificial Intelligence Lab., Stanford, Memo AIM-284, August 1976.

[Freuder 77]
E.C. Freuder : "A computer system for visual recognition using active knowledge"
5th IJCAI, Cambridge, August 1977.

[Garvey 76]
T.D. Garvey : "Perceptual strategies for purposive vision"
SRI International, AIC, TN 117, September 1976.

[Gennery 82]
D.B. Gennery : "Tracking known three-dimensional objects"
2nd AAAI Conference, Carnegie Mellon University, August 1982.

[Gini 78]
G. Gini, M. Gini : "Object description with a manipulator"
The Industrial Robot, March 1978.

[Gini 79]
G. Gini, M. Gini, R. Gini, D. Giuse : "MAL : A multi-task system for mechanical assembly"
International Seminar on programming Methods and Languages for Industrial Robots, INRIA, June 1979.

[Giralt 79]
G. Giralt, R. Sobek, R. Chatila : "A multi-level planning and navigation system for a mobile robot. A first approach to HILARE"
6th IJCAI, Tokyo, August 1979.

[Grimson 81]
W.E.L. Grimson : "From images to surfaces"
The MIT Press, 1981.

[Horaud 81]
P. Horaud : "Extraction et représentation de contours dans une image. Applications à l'inspection automatique"
Thèse de Docteur-Ingénieur, INP Grenoble, December 1981.

[Huffman 71]
D.A. Huffman : "Impossible objects as nonsense sentences"
In Machine Intelligence 6, Edited by B. MELTZER and D. MICHIE, Elsevier 1971.

[Konolidge 80]
K. Konolidge, N.J. Nilsson : "Multi-agent planning systems"
1st AAAI Conference, Stanford, August 1980.

[Latombe 79]
J.C. Latombe : "Une analyse structurée d'outils de programmation pour la robotique industrielle"
International Seminar on Programming Methods and Languages for Industrial Robots, INRIA, June 1979.

[Latombe 81]
J.C. Latombe, E. Mazer : "LM : a high-level programming language for controlling assembly robots"
11th ISIR, Tokyo, October 1981.

[Latombe 82a]
J.C. Latombe : "Equipe 'Intelligence Artificielle et Robotique' : état d'avancement des recherches au 1er Octobre 1981"
IMAG, Grenoble, RR n° 291, February 1982.

[Latombe 82b]
J.C. Latombe, A. Lux : "Basic notions in knowledge representation and control for computer vision"
In "Computer Vision", O. Faugeras Ed., Cambridge University Press, 1982 (to appear).

[Laugier 81]
C. Laugier : "A program for automatic grasping of objects with a robot arm"
11st ISIR, Tokyo, October 1981.

[Lieberman 77]
L.I. Lieberman, M.A. Wesley : "AUTOPASS : An automatic programming system for computer controlled mechanical assembly"
IBM Journal of Research and Development, July 1977.

[Lieberman 79]
L. Lieberman : "Model-driven vision for industrial automation"
In "Advances in Digital Image Processing", Plenum Press, 1979.

[Lozano-Perez 76]
T. Lozano-Perez : "The design of a mechanical assembly system"
Artificial Intelligence Lab., MIT, AI-TR-397, December 1976.

[Lozano-Perez 79]
T. Lozano-Perez, M.A. Wesley : "An algorithm for planning collision-free paths among polyhedral obstacles"
Communications of the ACM, October 1979.

[Lozano-Perez 80a]
T. Lozano-Perez : "Spatial planning : a configuration space approach"
Artificial Intelligence Lab., MIT, Memo 605, December 1980.

[Lozano-Perez 80b]
T. Lozano-Perez : "Automatic planning of manipulator transfer movements"
Artificial Intelligence Lab., MIT, Memo 606, December 1980.

[Mackworth 73]
A.K. Mackworth : "Interpretation pictures of polyhedral scenes"
Artificial Intelligence, 1973.

[Mazer 82]
E. Mazer : "LM-geo : geometric programming of assembly robots"
IMAG, Grenoble, RR n° 296, March 1972 .

[McDermott 82]
D. McDermott : "A temporal logic for reasoning about process and plans"
Cognitive Science (Forthcoming), 1982 .

[Mezin 82]
G. Mezin : "Réalisation d'un système d'acquisition d'images tridimensionnelles pour la robotique"
IMAG, Grenoble, Mémoire CNAM, November 1982.

[Moravec 79]
H.P. Moravec : "Visual mapping by a robot rover"
6th IJCAI, Tokyo, August 1979.

[Nilsson 71]
N.J. Nilsson : "Problem solving methods in Artificial Intelligence"
McGraw-Hill, 1971 .

[Nilsson 80]
N.J. Nilsson : "Principles of Artificial Intelligence"
Tioga, 1980.

[Nitzan 77]
D. Nitzan, A.E. Brain, R.O. Duda : "The measurement and use of registered reflectance and range data in scene analysis"
Proceedings of the IEEE, 1977.

[Paul 77]
R.P. Paul : "WAVE : A model-based language for manipulator control"
The Industrial Robot, March 1977.

[Paul 80]
R.P. Paul et al. : "Advanced industrial robot control systems"
School of Electrical Engineering, Purdue University, TR-EE 80-29, July 1980 .

[Paul 81]
R.P. Paul : "Robot Manipulators : mathematics, programming and control"
The MIT Press, 1981.

[Perkins 77]
W.A. Perkins : "Model-Based vision system for scenes containing multiple parts"
5th IJCAI, Cambridge, August 1977.

[Perkins 78]
W.A. Perkins : "A model-based vision system for industrial parts"
IEEE Transactions on Computers, February 1978.

[Pertin 82]
J. Pertin : "Calculs d'accessibilité. Application à la saisie automatique"
IMAG, Grenoble, Rapport de DEA, June 1982.

[Popplestone 75]
R.J. Popplestone, C.M. Brown, A.P. Ambler, G.F. Crawford : "Forming models of plane-and-cylinders faceted bodies from light stripes"
4th IJCAI, Tbilisi, September 1975.

[Popplestone 78]
R.J. Popplestone, A.P. Ambler, I. Bellos : "RAPT : a language for describing assemblies"
The Industrial Robot, September 1978.

[Powell 82]
E.G. Powell : "An efficient collision warning algorithm for robot arms"
2nd AAAI Conference, Carnegie Mellon University, August 1982.

[Queromes 82]
J.G. Queromes : "Computer Aided Design and Robotics : A full promise of cooperation"
12th ISIR, Paris, June 1982.

[Riad 81]
A. Riad, M. Briot : "Identification and localization of partially observed parts"
11th ISIR, Tokyo, October 1981.

[Rosenschein 82]
J.S. Rosenschein : "Synchonization of multi-agent plans"
2nd AAAI Conference, Carnegie Mellon University, August 1982.

[Sacerdoti 73]
E.D. Sacerdoti : "Planning in a hierarchy of abstraction spaces"
3th IJCAI, Stanford, August 1973.

[Sacerdoti 75]
E.D. Sacerdoti : "A structure for plans and behavior"
SRI International, AIC, TN 109, August 1975 .

[Salmon 77]
M. Salmon : "Assembly by robots"
The Industrial Robot, June 1977.

[Sata 81]
T. Sata, F. Kimura, A. Amano : "Robot simulation system as a task programming tool"
11th ISIR, Tokyo, October 1981.

[Shafer 82a]
S.A. Shafer, T. Kanade : "Using shadows in finding surface orientations"
Department of Computer Science, Carnegie-Mellon University, CMU-CS-82-100, January 1982.

[Shafer 82b]
S.A. Shafer, T. Kanade, J.R. Kender : "Gradient space under orthography and perspective"
IEEE Workshop on Computer Vision, Rindge, New-Hampshire, August 1982.

[Shirai 78]
Y. Shirai : "Recognition of man-made objects using edge cues"
in "Computer Vision Systems", Edited by A.P. HANSON and E.M. RISEMAN, Academic Press, 1978.

[Shirai 79]
Y. Shirai : "Three-dimensional computer vision"
in "Computer vision and sensor-based robots", Edited by G.G. DODD and L. ROSSOL, Plenum Press, 1979.

[Siklossy 73]
L. Siklossy, J. Dreussi : "An efficient robot planner which generates its own procedures"
3rd IJCAI, Stanford, August 1973.

[Soroka 80]
B.I. Soroka : "Debugging robot programs with a simulator"
CADCAM-8 Conference, Anaheim, California, November 1980

[Souvignier 81]
V. Souvignier : "Prédiction et vérification en vision"
IMAG, Grenoble, Rapport de DEA, June 1981.

[Stefik 80]
M.J. Stefik : "Planning with constraints"
Computer Science Department, Stanford, Rep 80-784, 1980.

[Sussman 75]
G.J. Sussman : "A computer model of skill acquisition"
Noth Holland, 1975.

[Tate 77]
A. Tate : "Generating project network"
5th IJCAI, Cambridge, August 1977.

[Taylor 76]
R. Taylor : "A synthesis of manipulator control programs from task-level specifications"
Artificial Intelligence Lab., Stanford, Memo 282, July 1976.

[Tenenbaum 79]
J.M. Tenenbaum, H.G. Barrow, R.C. Bolles : "Prospects for industrial vision"
in "Computer vision and sensor-based robots", Edited by G.G. DODD and L. ROSSOL,
Plenum Press, 1979.

[Thorndyke 81]
P.W. Thorndyke, D. McArthur, S. Cammarata : "AUTOPILOT : a distributed planner for air fleet control"
7th IJCAI, Vancouver, August 1981.

[Udupa 77]
S. Udupa : "Collision detection and avoidance in computer controlled manipulators"
5th IJCAI, Cambridge, August 1977.

[Ullman 79]
S. Ullman : 'The interpretation of visual motion"
The MIT Press, 1979.

[VanderBrug 79]
G.J. VanderBrug, J.S. Albus, E. Barkmeyer : "A vision system for real-time control of robots"
9th ISIR, Washington D.C., March 1979.

[Veillon 77]
F. Veillon : "Une méthode de calcul en un passage de plusieurs propriétés topologiques et géométriques d'objects dans des images digitalisées"
IMAG, Grenoble, RR n° 67, January 77.

[Vernet 80]
M. Vernet : "Contrôle d'une équipe de robots à aptitudes multiples collaborant à l'exécution d'une même tâche"
Thèse de Docteur-Ingénieur, INPG, Grenoble, October 1980.

[Villers 82]
P. Villers : "Present industrial use of vision sensors for robot guidance"
12th ISIR, Paris, June 1982.

[Waldinger 75]
R. Waldinger : "Achieving several goals simultaneously"
SRI International, AIC, TN 107, July 1975.

[Waltz 75]
D. Waltz : "Generating semantic descriptions from drawing of scenes with shadows"
in "The psychology of computer vision", Edited by P.H. WINSTHON, McGraw-Hill, 1975.

[Weck 81]
M. Weck, W. Eversheim, D. Zuehlke : "ROBEX - an off-line programming system for industrial robots"
11th ISIR, Tokyo, October 1981.

[Wilkins 82a]
D.E. Wilkins : "Parallelism in planning and problem solving : Reasoning about resources"
SRI International, AIC, TN 258, January 1982.

[Wilkins 82b]
D. Wilkins : "Domain-independent planning : representation and plan generation"
SRI International, AIC, TN 266, August 1982.

[Will 75]
P.M. Will, D.D. Grossman : "An experimental system for computer controlled mechanical assembly"
IEEE Transactions on Computers, September 1975.

[Will 78]
P.M. Will : "Computer controlled mechanical assembly"
The Industrial Robot, March 1978

[Whingham 77]
M. Wingham : "Planning how to grasp objects in a cluttered environment"
M. Phil. Thesis, University of Edinburgh, 1977.

[Woodham 78]
R.J. Woodham : "Reflectance map techniques for analyzing surface defects in metal castings"
Artificial Intelligence Lab., MIT, AI-TR-457, June 1978.

[Woods 78]
W.A. Woods : "Theory formation and control in a speech understanding system with extrapolations towards vision"
in "Computer vision Systems", Edited by A.P. HANSON and E.M. RISEMAN, Academic Press, 1978.

[Yachida 77]
M. Yashida, S. Tsuji : "A versatile machine vision system for complex industrial parts"
IEEE Transactions on Computers, September 1977.