Planning robot actions under position and shape uncertainty

Christian LAUGIER*
LIFIA/IMAG
46 Avenue Felix Viallet
38031 Grenoble Cedex, FRANCE.

Abstract

Geometric uncertainty may cause various failures during the execution of a robot control program. Avoiding such failures makes it necessary to reason about the effects of uncertainty in order to implement robusts strategies. In this paper, we first point out that a manipulation program has to be faced with two types of uncertainty: those that might be locally processed using appropriate sensor based motions, and those that require a more global processing leading to insert new sensing operations. Then, we briefly describe how we have solved the two related problems in the SHARP[1] system: How to automatically synthesize a fine motion strategy allowing the robot to progressively achieve a given assembly relation despite position uncertainty ? How to represent uncertainty and to determine the points where a given manipulation program might fail ?

1 Introduction

1.1 Statement of the problem

A robot and its working space constitute a complex mechanical system that cannot be completely modelled. This means that both the environment and the actions executed by the robot cannot be exactly predicted at programming time, and that the resulting uncertainty may cause the program to fail. Consequently, it is necessary to determine the points where uncertainty it too high according to the required precision, and then to reduce this uncertainty using appropriate *sensory based strategies*. The purpose of this paper is to discuss this problem, and to answer as far as possible the two following questions: How to automatically synthesize a fine motion strategy (i.e. a manipulation strategy combining sensing operations with small robot movements) allowing the robot to progressively achieve a given assembly relation despite position uncertainty ? How to represent uncertainty and to determine the points where a given manipulation program might fail ?

The first question is related to the fact that some position errors can be directly taken into account by the system when planning contact based motions. The basic hypothesis in this case, consists in assuming that the *local* environment of the task can be considered as a "geometric guide" for the robot. The second question is motivated by the fact that some other position errors have a more *global scope*, since they are directly related to the interaction that exists between the actions of the manipulation program. For example, a grasping operation generating a too large uncertainty on the position of the chosen grasping points, may lead to a failure during the next part mating operation. The basic strategy in this case consists in first propagating the uncertainty terms through the program in order to determine possible failure points, and then to amend the program by inserting appropriate sensory based operations.

*Senior Researcher at INRIA

[1]SHARP is an automatic robot programming system currently under development at LIFIA

1.2 Planning fine motion strategies

Several types of techniques have been developped for dealing with uncertainty in robot programming. Some of these techniques are aimed at executing "compliant motions" involving both force and position parameters in the command [23] [15] [8]. The other techniques were developped for the purpose of constructing complete fine motion strategies. A first approach for solving this problem consists in generating a solution by instanciating some predefined "procedure skeletons" using error bounds computations [19] [12] [14], or by assembling a set of partial strategies using learning techniques and expert rules [4]. The major limitation of this approach comes from the fact that it relies on the following hypothesis: any assembly operation can be unambiguously associated to a more general assembly class that can be processed using a single type of strategy. Unfortunately, a slight modification of the local geometry of the involved workpieces may drastically change the strategy to apply. This means that it seems impossible to identify a reasonable set of assembly classes.

A more general approach consists in constructing the fine motion strategies by reasoning on the geometry of the task [13] [6] [9] [22]. This approach leads to consider the local environment of the workpieces to assemble as a "geometric guide" for the robot. Then, planning a fine motion strategy may be seen as the determination of an ordered sequence of well chosen contacts and of intermediate situations. The formal bases of this approach are given in [13], and a first attempt of implementing it in a polygonal world is described in [6]. But one suspects a too high algorithmic complexity for making the problem manageable in real situations. This is why we have developed a method allowing to reduce the size of the search graph by heuristically guiding the geometric reasoning.

This method is briefly described in section 2. The basic idea consists in deducing a fine motion strategy from an analysis of the different ways in which the assembled parts may be theoritically dismantled. This analysis is executed on an explicit representation of the contact space. It leads to successively construct a *state graph* representing the set of potential solutions, and to search this graph in order to find a "good reverse path" defining a feasible fine motion program.

1.3 Reasoning on position uncertainty

This problem has already given rise to several theoritical and practical developments aimed at achieving two different goals: (1) dealing with position uncertainty when programming a manipulation robot [19] [1] [14] [17], and (2) combining the uncertain data provided by sensors while updating or constructing the model of the robot environment [3] [18] [7] [5]. In spite of their different formulations, these problems have led to the development of similar approaches for reasoning on position uncertainty (although the applied computational models are different). The basic mathematical tools involved in this reasoning are described in [16] and [17].

Work done in the context of robot programming is aimed at explicitly representing the error bounds associated to the position variables of a manipulation program, in order to propagate them through the model and to compute their values in any point of the program. This approach was initially devised for computing the values of the parameters associated to a set of predefined manipulation procedures [19] [14]. We will see further how it can be applied for evaluating the correctness of a manipulation program, according to the precision constraints imposed by the task. Some other researchers have considered the problem of reasoning on position uncertainty as a computational subproblem of the interpretation of sensory data. In order to avoid as far as possible to over-estimate the errors when combining such data, the developed approaches have been based on statistical models [18] [5] [7].

The statistical approach is probably well adapted to sensor fusion. But its computational characteritics seem to limit its applicability domain to rather simple situations involving very few contacts. Unfortunately, assembly programs include a great number of multiple contacts and of complex assembly relations. This is why we have first chosen to implement an approach based on error bounds

computations (see section 3.2). This approach was consistent with the main characteristics of our problem: checking for the correctness of a sequence of actions, very few propagation operations have to be applied for each assembly step, several terms of the initial uncertainty are quickly reduced by contacts. But a more recent implementation based on statistical models [17], has shown that the results obtained using the two approaches are very similar (in this particular context). This means that the choice of one of these models is not of a prime importance for us, and that this choice has no real consequences on our verification/ correction method.

2 Planning fine motion strategies

2.1 Outline of our approach

As mentioned above, the applied method for planning fine motion strategies leads to consider the local environment of the workpieces to assemble as a "geometric guide" for the robot. Then, planning a fine motion strategy may be seen as the determination of an ordered sequence of well chosen contacts. Each selected contact leads to decrease the "distance" to the goal situation by reducing the position uncertainty. It gives rise to the execution of a guarded compliant motion. All the parameters of the related motion command (motion direction and amplitude, force and termination conditions) are finally computed using various accessibility and reliability criteria.

More practically, our method *deduces a solution from an analysis of the different ways the assembled parts may be theoretically dismantled.* This approach has been motivated by the fact that the existing contacts constrain the relative movements of the two parts, and reduce this way the number of hypotheses to formulate. An important characteristic of our method is to progressively guide the search choices, by successively analysing more and more detailed constraints drawn from the geometry of objects. As we will see further, a practical way for doing that consists in separating the computation of potential reachable positions and valid movements, from the determination of those which are really executable by the robot. This approach has been implemented in our system by a two phases algorithm leading to successively construct a *state graph* representing the set of potential solutions, and to search this graph in order to find a "good reverse path" defining a feasible fine motion program:

- *The analysis phase* constructs the state graph by reasoning on a fictitious dismantling of the assembly. For that purpose, the system determines at each step the different contact situations which can be reached from the current situation by applying a single motion. Only the local moving constraints associated to the contacts are examined at this step. The applied method leads to progressively decrease the number of contacts.

- *The search phase* determines a "reverse path" in the graph, i.e. a path starting from a node having an empty set of contact and ending at the node corresponding to the final assembly. This search phase is based on heuristics which attempt to optimize the selected solution in terms of both efficiency (number of operations) and reliability (robustness of the selected motions). In case of failure –for example one contact situation cannot be achieved because of the control errors– the graph is locally refined by introducing some new potential motions and contacts [9].

This approach allows us to reduce the size of the search graph. It also leads to apply costly geometric computations only when intricate situations have to be processed. The applied method makes use of a symbolic representation of contacts which includes three types of information [11]: the geometric entities involved in the contact, the topology of the contact and the associated geometric parameters.

2.2 Reasoning on motion constraints

Each contact reduces the number of d.o.f of the moving object. In order to determine the next motion to execute from a given contact situation, the system must reason on the moving directions which are constrained by the contacts. Consequently, it is necessary to explicitly represent the valid movements which can be *locally* associated to a contact situation. In order to simplify the computations, we will consider that these movements are either pure translations or pure rotations which are defined independently of their possible amplitudes. If A is a mobile object in contact with B, we will define a potential motion for A as *"a motion having an amplitude greater than the maximum control error, and generating no collision between the features in contact"*. In practice, this definition has led us to develop an analytic support allowing to explicitly represent the forbidden motions, those which preserve the contacts and those which break them [11]. The applied method leads to represent a set of possible translating motions as a particular domain on a unitary sphere. The intersection of such domains is computed using simple functions of the type $\vartheta = \arctan((N_x \cdot \cos\varphi + N_y \cdot \sin\varphi)/ - N_z)$, where (N_x, N_y, N_z) is the normal external vector to the related contact plane [20].

Similar representations are also used for dealing with curved supports. But in this case, the completeness property of the representation is preserved by using a first order approximation, leading to locally represent the potential compliant motions as a set of tangential motions. Rotating motions are modelled using a different method leading to group together all the rotation axes which generate an "homogeneous behavior" relatively to the contacts (for example: rotations which maintain the topological properties of the contact).

2.3 Dealing with the state graph

The sets of contacts and of their associated potential motions are combined in order to construct the *state graph associated to the fine motions.* This graph represents all the combinations of motions and robot states which have been selected as *potential elements of solution* for the problem to be solved. It is represented by a directed graph $G(A/B)$, where each node represents a robot state E_p, and each arc defines a motion allowing the robot to move from one state to an other one [11].

Our analytic representation of potential motions allows us to characterize the whole set of movements which are potentially feasible from a given state E_p. But this representation cannot be directly used by the motion planner, since each constructed domain D represents an infinite set of possible solutions (and consequently an infinite set of possible arcs for each node in the state graph). A classical technique dealing with this problem, consists in *discretizing the sets of potential solutions.* This technique leads to split each domain D into a finite set of small spherical domains of the type $\Delta\varphi \times \Delta\vartheta$. Each obtained domain ΔS represents a set of motions which will be "globally" analysed by the system. This approach requires that all the motion directions in ΔS allows to theoritically achieve the same symbolic contact situation, when executing these motions from a given position P. If the objects are polyedra and the motions are pure translations, such a constraint may be evaluated using a "visibility" analysis technique [2].

But the high algorithmic complexity of this approach along with its inability to deal with rotations and curved surfaces, has led us to make use of an heuristic based approach. The basic idea consists in analysing a subset of the possible solutions, by selecting in D the most promising motion directions. This approach is consistent with the fact that most of the required movements for mating two mechanical parts, are executed along some *privileged directions* defined by the contact surfaces. Then, the state graph may be constructed using the following algorithm:

1. Create the node GS and insert it in the list $OPEN$.
2. If $OPEN = \emptyset$ then return $G(A/B)$, else process the node x located at the head of the list $OPEN$.

Choose n directions $d_1, d_2 \cdots d_n$ in D_x. Create an arc a_{xi} for each chosen direction d_i.

3. Create a node y_i for each arc a_{xi}. If the state E_i associated to y_i is already represented by a node z in $G(A/B)$, then merge y_i and z.

4. Insert the new nodes having an empty set of contacts in IS; insert the other nodes in the list $OPEN$. Goto (2).

$OPEN$ represents the list of the next nodes to process, and D_x is the set of potential motions associated to the node x. GS is the goal state (the parts A and B are assembled), and IS is the set of the possible starting states for the fine motion strategies (states having an empty set of contacts). The geometric functions which have been developed for computing the parameters of the graph items (contacts, potential motions and sensory identification for each node; valid ranges of positions, sliding surfaces and sticking surfaces for each arc) are described in [10] and [11].

The moving directions are selected in D_x according to an heuristic function. For example, four directions will be initially generated by a couple of non-parallel planar contacts: $d_1 = N_1 \wedge N_2$, $d_2 = -d_1$, $d_3 = d_1 \wedge N_1$ and $d_4 = d_2 \wedge N_2$, where N_1 and N_2 are the external normal vectors to the contact faces F_1 and F_2. d_1 and d_2 define two compliant motions allowing to maintain the contacts; d_3 and d_4 define two motions leading to respectively break the contact associated to F_2 and to F_1. These moving directions are considered by the system only if they are included in D_x.

2.4 Constructing a fine motion strategy

2.4.1 Searching the state graph

Let IS be the set of nodes of $G(A/B)$ which have an empty set of contacts, and GS the state corresponding to the situation where A and B are assembled. Any path in $G(A/B)$ starting from a node s in IS and ending at the node GS, may be considered as a fine motion strategy allowing to assemble A on B. Then searching for a solution in $G(A/B)$ can be done using the following algorithm:

1. Search for a path SG starting from a node s in IS and ending at the node GS.

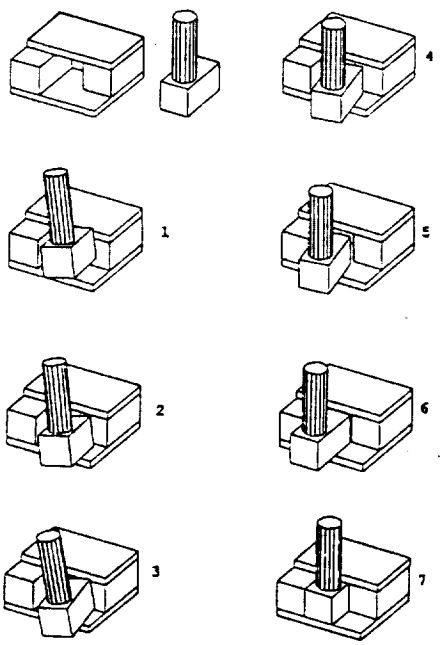
2. Verify that each arc in SG represents a feasible motion (no collision, reachability of the goal). Refine $G(A/B)$ in case of failure, and goto (1).

3. Synthesize the fine program represented by SG.

Since the current version of the system discards the potential motions which may stop in different contact situations (because of control errors), each selected solution SG is represented by a single path in $G(A/B)$ –and not by a "complete subgraph" as discussed in [10]–. Such paths are computed using a combination of a *cost function* and of a set of *dynamic advices* implemented using production rules [9] [20]. The cost function exploits the heuristic weights associated to the graph items, in order to both minimize the number of operations and to maximize the reliability of the selected motions. The dynamic advices are activated when some impractical situations are detected by the system (for example: adjacent contacts or closed obstacles). They lead to locally refine the graph. This approach allows us to reduce the algorithmic complexity by only exploring "in detail" the branches of the graph which are really significant according to the selected solution.

2.4.2 Synthesizing a fine motion program

Synthesizing a fine motion program from a path SG requires to first check for the validity (collision and reachability criteria) of each involved motion, and secondly to apply rewriting rules for generating the program. The validity tests are executed using the following computations [10]:



```
(REALIZE-S
    (R-robot ON R0)
    (VIA C1 C2 ... Cn))

(REALIZE-C
    (point-P1 ON face-B1)
    (ALONG (TRANSLAT :VECT vector-V1)))

(REALIZE-C
    (edge-A1 ON face-B1)
    (ALONG (ROTATION :AXE (make-axe :PT point-P1 :VECT vector-V2)))
    (BY-MAINTAINING (point-P1 ON face-B1)))

(REALIZE-C
    (face-F1 ON face-B1)
    (ALONG (ROTATION :AXE (make-axe :PT point-P1 :VECT edge-A1)))
    (BY-MAINTAINING (edge-A1 ON face-B1)))

(REALIZE-C
    (edge-A2 ON face-B2)
    (ALONG (TRANSLAT :VECT vector-V3))
    (BY-MAINTAINING (face-F1 ON face-B1)))

(REALIZE-C
    (face-F2 ON face-B2)
    (ALONG (ROTATION :AXE (make-axe :PT point-P2 :VECT edge-A2)))
    (BY-MAINTAINING (face-F1 ON face-B1)(edge-A2 ON face-B2)))

(REALIZE-C
    (point-P3 ON face-B3)
    (ALONG (TRANSLAT :VECT edge-A1))
    (BY-MAINTAINING (face-F1 ON face-B1)(face-F2 ON face-B2)))
```

Figure 1: A fine motion strategy computed by the system.

$Sweep(A, d) \cap Gros_\varepsilon(B) = \emptyset \Rightarrow$ no collision
$A(p) \cap Gros_\varepsilon^{-1}(B) \neq \emptyset \Rightarrow p$ is reachable

where $\varepsilon = 2(\varepsilon_p + \varepsilon_i)$ and the terms ε_p and ε_i represent respectively the control and the sensing error bounds; $Sweep(A, d)$ is the volume swept by A when moving along d, $Gros_\varepsilon(B)$ and $Gros_\varepsilon^{-1}(B)$ respectively represent the obstacles B grown and shrunk according to ε, and $A(p)$ is the object A in the configuration p. Then the missing motion parameters of each selected movement are computed, in order to synthesize a sequence of guarded compliant motions of the type:

MOVE <*objet-A*> ALONG <T> BY-MAINTAINING <C> UNTIL <A>

where T and C are the symbolic motion parameters recorded in the graph, and A represents the set of contacts which may stop the movement. The numerical values associated to these parameters at the execution time are computed using the geometric model and some predefined thresholds. For example, a face belonging to A will generate a condition of the type "$F_v > threshold$", where F_v is the projection of the reaction force on the moving direction v, and v is assumed to be included in the friction cone (this condition is associated to the termination predicate). A similar computation is executed for the compliant parameters, but the needed thresholds are currently tuned by the operator.

3 Dealing with global uncertainty constraints

3.1 Outline of our approach

The main problem to solve is to decide if a manipulation program produced by the system is guaranteed to work in the real world (according to the known world model), i.e. if the precision constraints imposed by the task are "compatible" with the error terms associated to the position variables of the program. This means that both nominal positions and error terms have to be computed in any point of the program, in order to be compared with their associated constraints. Since errors are modified

by robot actions, this computation must be executed using an appropriate *propagation mechanism*. Let us consider as an example an object which has been grasped and moved by the robot. Its initial position error have been reduced along some directions by the grasping operation, before beeing grown by a term representing the inaccuracy of the motion command. This type of computation is used by the system for both determining possible failure points and possible correction points. The related verification/ correction process operates in two modes [17]:

- In the verification mode, the system applies a *forward propagation mechanism* for computing the resulting uncertainties at any point of the program, in order to compare them with the uncertainty constraints imposed by the task.
- In case of failure (an uncertainty term does not verify the associated constraint), the system switches to the correction mode in order to determine the possible correction points and the type of corrective action to apply. For that purpose, it applies a *backward propagation mechanism* leading to compute the uncertainty constraints which should be verified in the precedent steps of the manipulation program, for avoiding the studied failure.

3.2 Modeling uncertainty

The *nominal position* of an object A is a theoretical value which is used for programming the robot, but which is never reached at execution time because of various positioning errors. The associated variable in the program is a geometric transform T_a verifying the relation $R_a = Base \star T_a$, where R_a is the frame associated to A, and $Base$ is the reference frame of the robot workspace. Then, the gap existing between the nominal position of R_a and its real position, may be represented by a geometric transform ε_a verifying the relation $R_a^* = Base \star T_a \star \varepsilon_a$, where R_a^* represents the real position of R_a. The same property holds when considering the relative positions of two objects A and B.

As explained in section 1.3, we have chosen to represent position uncertainties using error bounds. Then, the uncertainty I_{ab} associated to the position of R_b relatively to R_a, is defined as the set of all possible errors ε_{ab}. Each transform ε_{ab} may be characterized by a triplet (t, u, α), where t is a translating vector in $\Re^3$, u is a unitary vector representing the rotation axis, and α is the rotating angle. Then, the uncertainty I_{ab} may be seen as a subset E of the cartesian product $\Re^3 \times S(1) \times [-\pi \ +\pi]$, where $S(1)$ is the unitary sphere [17]. Since these subsets are generally difficult to compute, we will make use of approximations leading to first project E on each space $\Re^3$, $S(1)$ and $[-\pi \ +\pi]$, and then to approximate the obtained sets Tr, U and D using simple surrounding sets. This approach widely simplifies the involved computations. It is consistant with the verification/ correction scheme, which leads in this case to reason on the "worst case hypothesis". Using this approach, it becomes possible to represent an uncertainty by a set $Tr \times U \times D$, where Tr is either a sphere, a disc or a bounded straight line; U is either the unitary sphere $S(1)$ or a vector; D is an interval $[-\alpha \ +\alpha]$. For example, the position uncertainty associated to an object lying on an horizontal plane $z = a$, is represented by a set of the type: $Disc(z_o, \varepsilon) \times \{vector\ z_o\} \times [-\alpha \ +\alpha]$. In case of a vertical cylindrical contact, the first item (the disc) is replaced by an interval $[b - \varepsilon \ \ b + \varepsilon]$ in the z direction (b is the nominal position of the object).

Remark: In the statistical approach, the uncertainty I_{ab} is modeled using the covariance matrix of ε_{ab}, when the six parameters (translation and rotation vectors) of ε_{ab} verify a gaussian law [17].

3.3 The world model

A *world state* is represented by a directed graph, where each node represents the reference frame associated to an object, and each arc denotes a geometric transform and its associated uncertainty term. The basic structure of this representation is a tree having the reference frame of the robot

workspace as root. This root usually represents the fixed base of the robot. Then, each arc defines either a spatial relation between two objects, or a physical relation created by a particular action of the robot (for instance: a contact between two objects or a functional link between a sensor and the sensed object). It is characterized by a couple $\{T_{ab}, I_{ab}\}$, where T_{ab} is a nominal geometric transform, and I_{ab} is the associated uncertainty.

This dynamic structure is fundamental for dealing with uncertainty, because it allows the computation of error terms at the points where they are really significant. For example, creating a contact between two objects A and B leads to reduce the *relative position uncertainty* of A and B. Then, it makes sense to explicitly represent this information in the world model, and to propagate its effects through the graph.

Lets consider the manipulation example shown in figure 2. The link $Arc(R_o, R_a)$ have been suppressed in the world state W_3, because it represents a "wrong" way for computing I_{oa} which only depends in this case on the terms I_{ba} and I_{ob}. Conversely, a new link $Arc(R_b, R_a)$ has been created after having computed T_{ba} and I_{ba}. T_{ba} represents the composition of several geometric transforms ($T_{ba} = T_m^{-1} \star T_{ob}^{-1} \star T_{oa}$, where T_m is the executed motion); I_{ba} is a similar expression, but it is computed using appropriate operators leading to combine sets of geometric transforms (these sets represent the uncertainty terms). In general, the related computations are difficult to implement [17]. Fortunately, the approximations which have been applied for representing uncertainty sets (spheres, discs and bounded straight lines), allow the implementation of simpler algorithms. For example, the composition of a sphere $S(\varepsilon_1)$ and of a disc $D(axis, \varepsilon_2)$ representing two translating uncertainties, will give rise to a new term represented by a sphere $S(\varepsilon_1 + \varepsilon_2)$.

Since world changes are caused by robot actions, two types of operators have to be developed for updating the world model: those devoted to the computation of the uncertainty associated to a combination of existing relations (*composition* of two uncertainties and *inversion* of an uncertainty), and those which are used for computing the uncertainty terms which have been modified by robot actions (*projection* and *fusion* operators). The projection operator is used for computing the terms which have been shrunk by a contact; the fusion operator is used for computing the resulting uncertainty when a sensing operation have been executed. For example, the resulting uncertainty I_{ba} in W_3 is obtained by projecting the expression $I_{ob}^{inv} \otimes I_{oa}$ on $P \times N \times [-\pi \ + \pi]$, where P is a plane parallel to the jaws of the gripper, and N is a direction normal to P; $\otimes$ and "*inv*" are respectively the composition and the inversion operators mentionned above.

3.4 Modeling robot actions

Propagating uncertainties through a manipulation program requires to precisely know the effects of robot actions (in terms of nominal positions and of associated uncertainties). Since the existing robot programming languages provide instructions which cannot be fully interpreted in these terms, we have developed a geometric based language aimed at avoiding such ambiguities [10]. All the same, the required "predictibility" property is not verified for motion commands having several possible termination conditions, and for commands involving compliant motions. This is why we have chosen to only apply the verification/ correction scheme on linear sequences of operations of the following types: free space motions, motions aimed at achieving or at breaking a contact (this includes grasping and dropping operations), and sensing operations aimed at locating an object. Then, more complex constructions implementing fine motion strategies are supposed to be globally correct. For that purpose, they will be assimilated to "macro-actions" that can be used for achieving complex assembly relations (in this case, the final uncertainty is directly deduced from the known assembly clearances).

In our geometric language, the semantics associated to a robot action is defined by the modifications that this action applies to the world model: nominal positions and their associated uncertainties

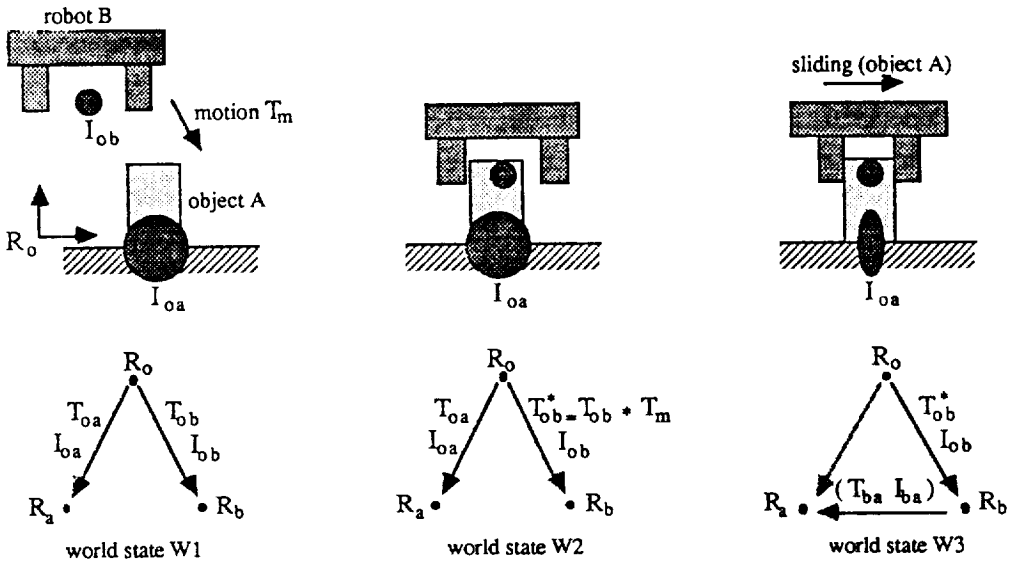


Figure 2: World model modifications generated by a grasping operation.

(nodes), spatial and physical relations (arcs). Free space motions lead to modify the position parameters, according to the executed movements and to the accuracy of the robot. Motions involving contacts lead to both modify the position parameters and to locally change the graph structure, as explained in section 3.3 (Figure 2 illustrates).

3.5 The verification/correction scheme

3.5.1 Representing a constrained manipulation plan

According to the previous hypotheses, a manipulation plan may be seen as a linear sequence of actions $A_1\ A_2\ \cdots\ A_n$. These actions lead to progressively move from an initial world state W_o to a final one W_n. Each world state W_i is represented by a graph as explained in section 3.3; its contents depends on both the last executed action A_i and the previous world state W_{i-1}. But the action A_i is guaranteed to produce the expected result (i.e. the world state W_i), only if some conditions hold before its execution (i.e. in W_{i-1}). For example, a grasping operation may fail if the uncertainty associated to the initial position of the object is too high relatively to the width of the jaws. Then, checking for the correctness of a manipulation plan necessitates to reason on such conditions. This means that the plan must contain an explicit representation of the *position constraints* CO_i that have to be verified in each world state W_i, $i = 0, n$. Such constraints are associated to the position variables of the plan. They are expressed using couples of the type $(T^i_{ab} = t_o\ ,\ I^i_{ab} \prec i_o)$, where T^i_{ab} and I^i_{ab} are respectively the nominal transform and the uncertainty term associated to the relative position of the objects A and B in W_i; according to our model, the relation $\prec$ is defined using the classical subset operator: $I_{ab} \prec I_o \ \Leftrightarrow\ (Tr_{ab} \subset Tr_o) \wedge (U_{ab} \subset U_o) \wedge (D_{ab} \subset D_o)$.

Then, a constrained manipulation plan will be considered as "correct", iff the constraints CO_i, for $i = 0, 1 \cdots n$, are verified in the related world states W_i resulting from the execution of the actions $A_1\ A_2 \cdots A_i$.

3.5.2 Checking for the correctness of the plan

Let C_i be the conditions holding in W_i on the position variables – a condition is represented by a couple of the type $(T_{ab} = t_o\ ,\ I_{ab} = i_o)$–. These conditions have been obtained after having "applied" the actions $A_1 \cdots A_i$, to the inital conditions C_o holding in W_o. For instance, the uncertainty on the position of an object A in W_i, depends on both the inaccuracy of the used feeder and the errors introduced by the manipulation operations executed on A.

Then, computing C_i requires to propagate the initial conditions C_o through the actions A_1 to A_i. Such a mechanism is refered as *forward propagation*. It leads to compute at each step k the *strongest postcondition* $POST_k(C_{k-1})$, obtained after having applied the action A_k to the conditions C_{k-1} holding in W_{k-1}. For example, the condition "$I_{oa} = i_o$" will give rise to the condition "$I_{oa} = i_o + I_{move}$", after having executed the action *"MOVE A BY T"*. Such a computation is executed using the semantics of the actions as explained in section 3.4. It is recursively applied to the plan, using the following property: $C_i = POST_i(C_{i-1})$. Then, the plan will be considered as "correct" iff C_i implies CO_i, for $i = 0, 1 \cdots n$.

Remark: A "correct" plan is guaranteed to work, provided that no unexpected physical fault occurs at execution time (an object fall for instance). But the method may sometimes lead to conclude that a quite reliable plan is not correct, because of the applied approximations.

3.5.3 Amending the plan

In case of failure (a constraint C is not verified in W_i), the system applies a *backward propagation mechanism* for determining the uncertainty constraints which should hold in the previous world states, in order to guarantee that C will be verified in W_i. The main idea consists in computing at each step k the *weakest precondition* $PRE_k(Ck)$ in W_{k-1}, which guarantee that the condition C_k will hold in W_k. For example, the condition "$I_{oa} < i_o$" will hold after execution of the action *"MOVE A BY T"*, if I_{oa} verifies the condition "$I_{oa} + I_{move} < i_o$" in the precedent world state. Such a computation is executed using the semantics of the actions as explained in section 3.4. It is recursively applied to the plan, using the property $C^i_j = PRE_{j+1}(C^i_{j+1}) \wedge CO_j$, where C^i_j represents the constraints of W_j $(j < i)$ which have been derived from CO_i using the backward propagation mechanism. Then, modifying W_j in order to verify the conditions C^i_j, leads to obtain a correct subsequence $A_{j+1} \cdots A_i$ (because each constraint in CO_k is verified, for $k = j \cdots i$).

The last step consists in determining *where* and *how* to amend the plan. Since any world state W_k (for $k \leq i$) may be initially chosen, it is necessary to determine where the amending operation has to take place. Unfortunately, it seems that no decisive criterion exists for guiding this choice.

The other problem is to determine the amending strategy which seems to be the more appropriate. In our approach, this operation is executed by *"patching"* the assembly plan, using a well suited sensory based strategy. But, no practical method has currently been devised for solving this problem.

4 Conclusion

In this paper we have described two complementary methods for dealing with position and shape uncertainty when planning robot actions. The first method operates at a *local level* for planning the fine motion strategies required for achieving the strongly constrained assembly relations. The other method is applied in a second time. It operates at a *global level* for amending the produced manipulation plan, when some uncertainty constraints are not verified.

The fine motion planner described in the paper has been implemented in LUCID-LISP on a SUN 260. Most of the experimentations have been executed in simulation. Some of them have given rise to real executions using a six d.o.f SCEMI robot equipped with a force sensor. This is the case for

the example shown in figure 1 which has been successfully executed by the robot. 9 mn of CPU time was needed for synthesizing the related fine motion program. The constructed state graph was made of about 50 nodes and 80 arcs.

The verification/correction module has been partly implemented in LUCID-LISP on a SUN 260. The two related propagation mechanisms have been tested on some simple examples, using both the error bounds representation and a gaussian based model. Despite the theoritical differences existing between the two approaches, the obtained results look very similar when dealing with simple manipulation plans involving very few data fusion operations. However, more realistic experiments are still needed for really evaluating the scope of our method. A complementary work is also needed for solving the plan amendment problem.

Acknowledgements:

This paper is based on the work done by Pascal Theveneau and Pierre Puget in the scope of the SHARP project of the LIFIA Laboratory. It was partly supported by the French National ARA project of the CNRS, the ADI agency, the ITMI company, and the INRIA institute.

References

[1] R.A.Brooks: *"Symbolic error analysis and robot planning"*, International Journal of Robotics Research, vol 1, no 4, December 1982.

[2] S.J.Buckley: *"Planning and teaching compliant motion strategies"*, Ph.D Thesis, Artificial Intelligence Laboratory, MIT, January 1987.

[3] R.Chatila, J.P.Laumond: *"Position referencing and consistent world modeling for mobile robots"*, IEEE International Conference on Robotics and Automation, St.Louis, 1985.

[4] B.Dufay, J.C.Latombe: *"An approach to automatic robot programming based on inductive learning"*, 1st International Symposium on Robotics Research, Bretton Woods, August 1983.

[5] H.F.Durrant-Whyte: *"Consistant integration and propagation of disparate sensor observations"*, International Journal of Robotics Research, vol.6, nb.3, 1987.

[6] M.Erdmann: *"On motion planning with uncertainty"*, AI-TR-810, Artificial Intelligence Laboratory, MIT,1984.

[7] O.D.Faugeras,M.Hebert: *"The representation, Recognition, and locating of 3D Objects"*, International Journal of Robotics Research, vol.5, nb.3, 1986.

[8] C.Gandon: *"Introduction de la compliance dans la programmation des robots"*, Thèse de 3ème Cycle, INPG, Grenoble, October 1986.

[9] C.Laugier, P.Theveneau: *"Planning sensor-based motions for part-mating using geometric reasonning"*, ECAI'86, Brighton, July 1986.

[10] C.Laugier: *"Raisonnement géométrique et méthodes de décision en robotique. Application à la programmation automatique des robots"*, Thèse d'Etat, INPG, Grenoble, December 1987.

[11] C.Laugier: *"Planning robot motions in the SHARP system"*, Published in "CAD based programming for sensory robots", Edited by Bahram Ravani, NATO ASI Series F, vol.50, Springer-Verlag, 1988.

[12] T.Lozano-Perez: *"The design of a mechanical assembly system"*, AI-TR-397, M.I.T. Artificial Intelligence Laboratory, Cambridge, December 1976.

[13] T.Lozano-Perez, M.T.Mason, R.H.Taylor: *"Automatic synthesis of fine-motions strategies for robots"*, 1st International Symposium of Robotics Research, Bretton Woods, August 1983.

[14] T.Lozano-Perez, R.A.Brooks: *"An approach to automatic robot programming"*, AI Memo 842, Artificial Intelligence Laboratory, M.I.T., April 1985.

[15] M.T.Mason: *"Compliance and force control for computer controlled manipulators"*, IEEE International Conference on Robotics and Automation, June 1981.

[16] I.Mazon, P.Puget: *"Modélisation des incertitudes de positionnement"*, Journées Géométrie et Robotique, Toulouse, May 1988.

[17] P.Puget: *"Vérification-correction de programmes pour la prise en compte des incertitudes en programmation automatique des robots"*, Thèse de l'Institut National Polytechnique de Grenoble, February 1989.

[18] R.Smith, M.Self, P.Cheeseman: *"A stochastic map for uncertain spatial relationships"*, IEEE International Conference on Robotics and Automation, Raleigh, March 1987.

[19] R.H.Taylor: *"Synthesis of manipulator control programs from task-level specifications"*, AIM 228, Stanford Artificial Intelligence Laboratory, July 1976.

[20] P.Theveneau: *Planification de mouvements fins de montage dans un système de programmation automatique de robots"*, Thèse de l'Institut National Polytechnique de Grenoble, November 1988.

[21] J.Troccaz, P.Puget: *"Dealing with uncertainties in robot planning using program proving techniques"*, 4th International Symposium of Robotics Research, Santa Cruz, August 1987.

[22] J.M.Valade: *"Raisonnement géométrique et synthèse de trajectoire d'assemblage"*, Thèse de Docteur-Ingénieur, Université Paul Sabatier, Laboratoire d'Automatique et d'Analyse des Systèmes, Toulouse, January 1985.

[23] D.E.Withney: *"Force feedback control of manipulator fine motions"*, Journal of Dynamic Systems Measurement and Control, June 1977.