Geometric reasoning in motion planning

C. Laugier

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 932

Programme 6

GEOMETRIC REASONING IN MOTION PLANNING

Christian LAUGIER

GROUPE DE RECHERCHE
GRENOBLE

Décembre 1988

RR_0932

Raisonnement géométrique en planification de mouvements
- Geometric reasoning in motion planning -

Christian LAUGIER*

Septembre 1988

Résumé

La planification de mouvements est un problème clé de la programmation automatique des robots. Nous présentons dans ce rapport les modèles géométriques et les techniques de raisonnement que nous avons développés dans le cadre du projet SHARP. Après avoir analysé les caractéristiques des modèles requis, nous montrons comment ces modèles ont été utilisés pour implanter deux formes de raisonnement géométrique: le raisonnement spatial qui permet de calculer des trajectoires sans collision pour le robot, et le raisonnement morphologique orienté vers la planification des mouvements qui mettent en jeu des contacts et des contraintes d'incertitude (i.e. mouvements impliqués dans les opérations de saisie et de montage). Le raisonnement spatial opère dans l'espace des configurations du robot. La résolution du second problème nécessite de raisonner sur les propriétés morphologiques des objets manipulés, afin de construire une représentation exploitable des contacts et des contraintes de mouvements qui leur sont associées.

Abstract

Automating the programming of assembly robots necessitates to develop methods for planning robot motions. In this paper we describe the geometric models and the reasoning techniques we have implemented as part of the SHARP system. We first present which modelling facilities are required for constructing a suitable representation of the robot world. Then, we show how this representation has been used for implementing two classes of reasoning functions: functions aimed at computing collision free trajectories for the robot and its payload, and functions allowing to automatically generate contact based motions under uncertainty constraints (i.e. motions involved in grasping and in part-mating operations). Our method for solving the first motion planning problem operates in the configuration space of the robot. Solving the second planning problem makes it necessary to construct an explicit representation of the involved contacts along with their associated moving constraints. It leads to reason on the morphological properties of the manipulated objects.

*LIFIA/IMAG, 46 Avenue Felix Viallet, 38031 Grenoble Cedex, France

1 Introduction:

Programming a robot for a specific assembly task requires to determine which robot actions have to be executed, which sensing operations are necessary, and the way actions and sensing are to be combined. The main problem to solve for automating this programming process consists in determining all the parameters of the involved robot motions (trajectory, velocity, accuracy, involved forces and torques ...). A general formulation of this problem can be stated as follows: *given a description of the initial and of the goal situations along with a complete model of the robot world, find a robot motion allowing to reach the goal situation without generating any collision between the robot (the arm and the payload) and the objects belonging to the robot workspace; moreover, the generated solution must verify various constraints (contacts, accuracy, velocity, robustness ...) depending on the context of the motion to be executed.*

The major difficulty that have to be faced when solving this problem, relies in the high algorithmic complexity which is associated to the find path problem. On a theoritical point of view, Schwartz and Sharir [49] have shown that there exists polynomial time algorithms for solving the trajectory planning problem for any type of manipulator. However, the execution time associated with these hypothetical algorithms have terms which make them impractical, even in relatively simple cases. Fortunately, a more "practical" work done in the context of automatic robot programming [34] [33] [40] [30] [41] has shown that it is possible to consider three main instances of the problem which can be separately solved using more specific approaches:

(1) *Transfer motions.* They represent large motions which are executed in a moderately occluded environment. Planning such motions requires to take into consideration the whole robot arm, in order to generate "safe trajectories" for the robot and its payload (i.e trajectories which can be executed with a high velocity without generating any collision). This means that the computed solutions are located far enough from the obstacles in order to avoid unexpected collisions caused by the control errors. Consequently, it is possible to approximate objects by more simple shapes for reducing the amount of geometric computations.

Two classes of approaches have been developped for solving this motion planning problem: the local approaches and the global approaches. The *local approaches* lead to progressively construct a safe trajectory by reasoning at each step on the local characteristics of the encountered situations. In this case, the local decisions are made using either a "generate and test" scheme [47] [1] or a function allowing to associate some repulsive fields to the obstacles [24] [15]. On the other hand, the *global approaches* operate on an explicit model of the position constraints imposed by the physical environment. Most of them are based on the "configuration space" scheme allowing to represent a safe trajectory for the robot arm by a connected set of free configuration values expressed in a n-dimensionnal space (where n is the number of degrees of freedom of the robot)

[36]. Several instances of the problem have been solved using such an approach: moving a bidimensionnal object in the plane [35] [9] [5], moving a polyhedral object in the three dimensionnal space [36] [6] [10], dealing with rotating joints [56] [18] [14] [29] [39]... A more detailed analysis of these methods can be found in [4] and in [32].

(2) *Grasping operations.* These operations involve small motions which are executed in a very constrained environment located in the vicinity of the robot gripper. These movements allow to both reach the chosen grasping position in the initial environment, and to remove the gripper from the final environment once the planned manipulation have been achieved. The related trajectories are often very simple, but they are constrained by several factors like the type of the involved contacts, the stability of the gripped object and the accessibility of the selected features.

Four different aspects of the automatic grasping problem have been studied in the literature [54]: (1) *The determination of the grasping features* is done by symbolically reasoning on the geometry of the object to be grasped; this reasoning is either based on sensing (visual or tactile) informations [19] [2] [22] [55] [52], or on a complete geometric model [34] [59] [25] [28] [27] [60]. (2) *The stability* of the object in the gripper have been studied using either mathematical models of static and friction [60] [16] [7] [44], or heuristics leading to apply some simple geometric computations [25] [54] [20]. (3) *The accessibility analysis* is aimed at determining if a given solution is reachable in the initial and in the final environments. The applied methods generally operate in the configuration space of the gripper [34] [27] [55]. (4) The problem of the *compatibility* of the selected grasp with the planned manipulation have currently received very few attention [20] [42].

(3) *Fine motions.* They represent sequences of small movements which are executed in a very constrained environment located in the vicinity of the manipulated object. Such motions have to be guided by the sensory data, since the position uncertainty may make the robot fail. The applied technique consists in considering the local environment as a "geometric guide" for the robot. The related trajectories are very simple, but they are constrained by the involved contacts and by the limited accuracy of the robot command and of the sensing operations.

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 [58] [43] [48] [17]. 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 predifined "procedure skeletons" using error bounds computations [50] [34] [40], or by assembling a set of partial strategies using learning techniques [11] [12]. A more general approach consists in constructing the fine motion strategies by reasoning on the geometry of the task [37] [13] [31]. As we will see further, our fine

motion planner is based on this approach.

In this paper we describe the geometric models and the reasoning techniques we have implemented as part of the SHARP system (SHARP is an automatic robot programming system currently under developpement at LIFIA). We first present in section 2 which modelling facilities are required for constructing a suitable representation of the robot world. Then, we show in section 3 how this representation has been used for implementing two classes of reasoning functions: functions aimed at computing collision free trajectories for the robot and its payload, and functions allowing to automatically generate contact based motions under uncertainty constraints (i.e motions involved in grasping and in part-mating operations). Our method for solving the first motion planning problem is described in section 4. It is based on two types of techniques aimed at computing the valid ranges of values associated with some selected motion directions, and at constructing and searching a graph representation of the free space. Solving the second planning problem makes it necessary to construct an explicit representation of the involved contacts along with their associated moving constraints. It leads to reason on the morphological properties of the manipulated objects. This point is developped in section 5. Finally, section 6 describes the motion planners which have been implemented in the SHARP system.

2 World modelling:

Several aspects of the world model have to be constructed and maintained by the system in order to make possible the geometric reasoning involved in motion planning (see [32]): the geometry and the physical properties of objects, the evolution of objects relationships, the robot motions, and the robot states (including sensory informations) associated to each world state. Most of these informations are not explicitly represented in the initial CAD models. Consequently, one of the first task of the system is to construct a more suited model of the robot world combining three main representations: a geometric model including topological informations on objects, a structured representation of world states, and a model of robot motions. All these informations are used at the planning time for constructing a problem oriented representation of the robot world. We will call "planning space" such a representation (see section 3.1).

2.1 The geometric models:

The geometric models required for motion planning are basically the same that those previously developped in the context of off-line robot programming [27] [53]: boundary representations for solid objects made of polyedra, parallelepipeds, cylinders, cones and spheres; numerical data specifying geometric parameters (radius, dimensions, coordinates ...) and geometric transforms.

Since motion planning requires to apply a lot of geometric computations involving volumic and topological properties, these models have been completed by two types of constructions aimed at reducing the amount of computations [54] [32]: a hierarchy of elementary surrounding volumes (parallelepipedic boxes), and an explicit representation of the topology of objects (spatial hierarchy, local structures of the type "winged edge", and matter distribution in the vicinity of the elementary elements like faces or edges). Such constructions allow the system to apply fast interference checking algorithms, and to easily extract local informations on object shapes. For example, the grasp planner can check for the possibility of establishing a contact between a jaw of the gripper and a particular feature of the object, by applying very simple algebraic computations involving a small number of geometric entities (for example: a face and its normal external vector, and the counterclock oriented edges located in the vicinity of the analyzed object feature).

2.2 The world states:

Since the robot modify its environnement when operating, the world model has to be changed according to the executed actions. If we make the assumption that the robot operates in a "closed world" (i.e any world modification is the consequence of a robot action), it is possible to associate each world change to a particular robot sensing or manipulating operation. Since the purpose of motion planning is to find sequences of

actions allowing the robot to progressively reach a goal state from an initial state, it is possible to only represent the world states obtained after each elementary robot action.

In our system, each world state is represented by a directed graph where nodes denote cartesian frames associated to objects, and arcs represent objects relationships. Position informations on objects are expressed in terms of nominal geometric transforms and of associated uncertainties (see [32]). The basic structure of the graph is a hierarchy where the root is the reference frame of the robot world. The other arcs represents physical constraints existing between couples of objects (joints, contacts ...). In the current implementation of the system, contact relations are automatically determined and updated using a set of "demons" (see [51]).

2.3 Modelling the robot motions:

2.3.1 The motion parameters:

In order to guarantee that the planned motions will achieve their expected goals, the system must reason on a model including physical informations like forces and frictions, control and sensing errors, characteristics of the applied command, and termination predicates. In our system, we have implemented very simple models for representing these parameters [32]: friction cones based on a rough approximation of the static coefficients, error bounds for control and sensing, generalized spring type of command, and termination predicates represented by cartesian products of the type $P_a \times F_a$, where P_a is a subset of position in $\Re^3 \times SO^3$ and F_a is a set of reaction forces. Such a model is sufficient for solving the assembly problem we are concerned with, if we make the assumption that the involved contacts can be unambiguously identified using position and force data. More complex models may be required for applications having possible sensing interpretation ambiguities (see [13] and [8]).

2.3.2 The configuration space:

Geometric aspects of motions have also to be modelled in order to make motion planning possible. The developped representation is based on the *configuration space* scheme, first introduced in [36].

Definition 2.1 *Let A be a mobile system composed of l ($l \geq 1$) rigid elements moving in a cartesian space $\Re^k$ ($k = 1, 2 ou 3$). A configuration c of A is a minimal set of parameters, allowing to unambiguously specify the position and/or the orientation in $\Re^k$ of each rigid componant of A.*

A configuration c of A is represented by a vector in a n-dimensionnal space. We will say that A has n degrees of freedom (d.o.f), and that $A(c)$ represents the "position" of the whole system A in $\Re^k$ when A is in the configuration c.

Definition 2.2 *One call* **configuration space** C_A *of a mobile A, the set of the possible values of the configuration vector c. One call* **free space** EL_A *the set of configurations c such as A(c) do not generate collision between the componants of A and the objects belonging to the environment of A.*

EL_A is a subset of C_A, and C_A is a n-dimensionnal set defined by the cartesian product $I_1 \times I_2 \cdots I_n$, where I_i is the set of possible values for the parameter p_i of c. If A is a solid object in $\Re^3$, C_A may be seen as a subset of $\Re^3 \times SO^3$ (SO^3 is the group of the orthogonal rotations); if A is six d.o.f revolute arm, C_A is a subset of a 6-dimensionnal space often called "6-tore" because of its particular topological structure. Using the notations introduced in [36], the image $CO_A(B_i)$ of an obstacle B_i in C_A (called C-obstacle) and the free-space EL_A may be characterized as follow:

$$CO_A(B_i) = \{c \in C_A ; \quad A(c) \cap B_i \neq \emptyset\}$$
$$EL_A = C_A - \cup_{i=1}^{m} CO_A(B_i)$$

The techniques which have been developped in SHARP for computing the C-obstacles and the free-space associated with an articulated robot operating in $\Re^3$ are described in section 4.

3 Outline of our motion planning approach:

3.1 The concept of planning space:

The models described in the sections 2.1 and 2.2 cannot be directly used by the motion planners, because they do not explicitly contain all the needed informations. Consequently, the system has to construct a more suited representation of the robot world before starting to plan. This representation is problem oriented, and it is called the *planning space* associated with the motion planning problem to solve. It characterizes *the sets of robot states that can be realized using the commands provided by the system, and that can be identified using the available sensor devices.* Only position and force sensory informations are considered in our system. Consequently, we chose to represent a *robot state* E_p associated with a commanded position p ($P \in \Re^3 \times SO^3$), by the set of sensory couples (p^*, f^*) which can be read on the position and the force sensors, once the command has been executed by the robot. If $\xi(\Re^3 \times SO^3)$ represents an Euclidian approximation of $\Re^3 \times SO^3$ [13], a robot state E_p can be defined as follow [32]:

$$E_p = \{(p^*, f^*) : p^* \in B(p, \varepsilon_p + \varepsilon_m) \ \wedge \ f^* \in cone(n, \phi + \vartheta_f)\}$$

where n is the external normal vector in p to the contact surface, and ϕ is the angle associate with the friction cone. ε_p, ε_m and ϑ_f are respectively the position control error bound, the position sensing error bound, and the force sensing error bound. Then, the planning space E can be defined as set of robot configurations p in C_{robot} which verify the following properties:

- $E_p \cap E_{p'} = \emptyset \quad \forall p, p' \in E$
- $C_{robot} = Closure\ (\cup_{p \in E}\ P_p)$

Two major difficulties have to be faced when constructing such a space [8]: the set of robot states is not enumerable, and the dimension of the space do not allow to exactly represent all the constraints drawn from the object surfaces. This is the reason why we will construct an approximate model, by grouping together the robot states which can be considered as "equivalent" relatively to the type of motion to be planned. In order to be exploited by the planners, this model is structured as a *state graph*, where each node n_i defines a set E_i of "equivalent states" and each arc a_{ij} represents the motions allowing the robot to move from any state e_i in E_i to any state e_j in E_j. Using this representation, it becomes possible to consider the motion planning problem as an instance of the graph search problem.

Since motions involving contacts necessitate to take into consideration both position and force criteria whereas free-space motions (i.e motions executed at a distance

to obstacles greater than $\varepsilon_p + \varepsilon_m$) only rely on position criteria, two types of representations have to be considered when planning. In our system, these representations are constructed using the following criteria [32]:

- A class of "equivalent states" for free-space motions is defined as a *convex set of robot configurations which generate no collision with the environment.* An important property of this approach is to guarantee that any trajectory corresponding to a straight line (in the C-space) between two configurations of a given class is collision free. As we will see in section 4.5, the classes constructed in SHARP are hyperparallelepipeds obtained by discretizing the joint space.

- A class of "equivalent states" for motions involving contacts is defined as a *connex set of robot configurations which generate similar reaction forces* [8]. Using this definition, a face, an edge or a vertex of an object may be considered as a potential basis for defining a set of equivalent states (the condition to verify in this case is that each sensory couple (p^*, f^*) can be unambiguously associated to a single set of contact points). Several levels of details may be required depending on the characteritics of the motion to be planned. In our system, we choose to operate on a minimal set of classes, by only considering the different contacts that can be realized between the mobile and the concerned fixed objects (see section 5).

The motion planners that we have developped on the basis of these models are *consistant* since the generated solutions are guaranted to work despite sensory and control errors. Conversely, they are not *complete* since our representation eliminates two types of solutions: those which are ambiguous because of the interpretation mechanisms, and those which are missed because of the applied approximations. But the completeness property is not of prime importance, as long as the missed solutions do not make the system fail in practical cases. Fortunately, the processed mechanical assemblies generate a large set of possible solutions that can be found by the system, provided that the applied approximations are accurate enough. A refining process may be applied in case of failure.

3.2 The geometric reasoning involved in motion planning:

The problem to solve is to automatically generate the various robot motions which are involved in the basic manipulation operations (transfer, grasping and part-mating). As mentioned above, two types of motions have to be considered at planning time: free space motions and contact space motions. The reasoning techniques required for planning these different types of motions are called "spatial reasoning" and "morphological reasoning".

The spatial reasoning techniques are aimed at computing safe trajectories for the robot (i.e trajectories which are both collision free and robust relatively to sensing and control errors). A first class of computational tools have been developped for solving this

problem in the context of grasping and part-mating operations. These tools are aimed at computing the "valid ranges of positions" which can be achieved by the gripper and the manipulated object without colliding with the other objects. They operate on an "exact" representation (relatively to the used world model), since involved contacts do not allow to apply too large approximations. In this case, the computed solutions are composed of simple trajectories leading to create or to destruct a set of contacts. The other class of computational tools have been developped in the context of transfert motions. These tools operate on a complete model of the robot configuration space. But in this case the algorithmic complexity inherent in the general trajectory planning problem is reduced by applying various approximations.

The morphological reasoning techniques are aimed at computing the robot states involved in grasping and part-mating operations. The computational tools developped for that purpose allow to compute three main properties of contacts [32]: local accessibility, mechanical constraints and motion constraints. They are based on simple geometric and topological computations performed using local informations on object shapes.

The planning of free space motions can be done by applying pure spatial reasoning techniques, whereas the planning of motions involving contacts requires to combine morphological and spatial reasoning techniques in order to master the algorithmic complexity. The basic idea consists in progressively guiding the search choices, by successively analysing more and more detailed constraints drawn from the geometry of objects. This method has been implemented in SHARP by applying an ordered set of *simple geometric filters.* It leads to separate the computation of potentially reachable positions and valid movements, from the determination of those which are really executable by the robot. The two related reasoning phases are the following:

1. In a first step, the system computes a set of *potential solutions* by analysing the local properties of the implicated contacts. For exemple, it will generate a set of "potential grasps" by studying potentials contacts between the gripper and the object to be grasped. It also will determine "potential moving directions" associated with a set of contacts, by analysing the related topological constraints. This computation is useful when generating a sequence of compliant motions aimed at mating two parts. All the computational techniques applied during this phase are called morphological reasoning techniques, since they lead to reason on local morphological properties of objects.

2. The second phase leads to evaluate the *global accessibility* of the suggested solutions. It analyses the global constraints drawn from the task, in order to reject or to validate the previous choices. The retained solutions are then refined and completed according to the results of the applied computations. For example, the obstacles which constrain the movements of the jaws of the gripper, will lead to prune the set of possible grasping positions. The computation of the valid ranges

of position associated with a potential moving direction, will also lead to specify the missing motion parameters. All the computational tools applied during this phase are derivated from the spatial reasoning techniques, since they lead to reason on the spatial constraints drawn from the robot environment.

4 Spatial reasoning techniques:

4.1 Presentation and notations:

The purpose of this section is to describe the basic techniques which have been developped in SHARP for computing the images $CO_A(B)$ of obstacles B in the configuration space C_A of A, and for constructing an explicit representation of the free-space EL_A. We will make use within the presentation of the basic notations and definitions developped in [36].

The main difficulty associated with the computation of C-obstacles $CO_A(B)$ comes from the fact that these sets represent hypervolumes in a n-dimensionnal space which is generally non isomorphic to the Euclidian space $\Re^n$. In general, an exact determination of these sets requires to make use of mathematical tools, which can not be really applied because of their associated algorithmic complexity. This is the reason why we often reason on subspaces of C_A, in order to successively explore different subsets of the d.o.f of A. Such an approach allows to iteratively construct approximations of the sets $CO_A(B)$, when exact representations cannot be computed. We will represent by C_A^{xyz} the subspace of C_A associated with the three translations along axes x, y and z, and by CO_A^{xyz} the related C-obstacles. In the same way, we will represent by $C^q_{A_q,A_1(v_1)\cdots A_{q-1}(v_{q-1})}$ the subspace associated with the joint q of an articulated structure A, when the anterior joint variables are fixed to the values $v_1, v_2, \ldots v_{q-1}$, and the posterior links are not considered. In order to simplify the notations, we will represent by C_A^q this subspace, and by $CO_A^q(B)$ the related C-obstacles.

4.2 Computing valid ranges of positions:

The problem to solve consists in determining all the ranges of position of a mobile object A, that can be reached in a given direction without colliding with other objects B_i. A more formal definition may be stated as follows:

Definition 4.1 *One call valid ranges of position $V_A(q)$ associated with the mobile A moving along a direction q, the set of values q such as: $A(q) \cap B_i = \emptyset$.*

Definition 4.2 *Let c_j be a set of joint values of the type $(v_1, v_2, \ldots v_{j-1}, v_{j+1}, \ldots v_n)$. One call valid ranges of position $V_{A[c_j]}(q_j)$ associated with an articulated mobile A moving along a direction q_j of C_A, the set of values q_j such as: $A(v_1, v_2 \ldots v_{j-1}, q_j, v_{j+1} \ldots v_n) \cap B_i = \emptyset$*

The determination of the sets $V_A(q)$ requires to apply interference and collision checking algorithms, derived from those developped in [3]. These algorithms allow to determine the possible contacts between two polyhedral objects A and B, by analysing the intersections between the geometric entities of B, and the curves and the surfaces described

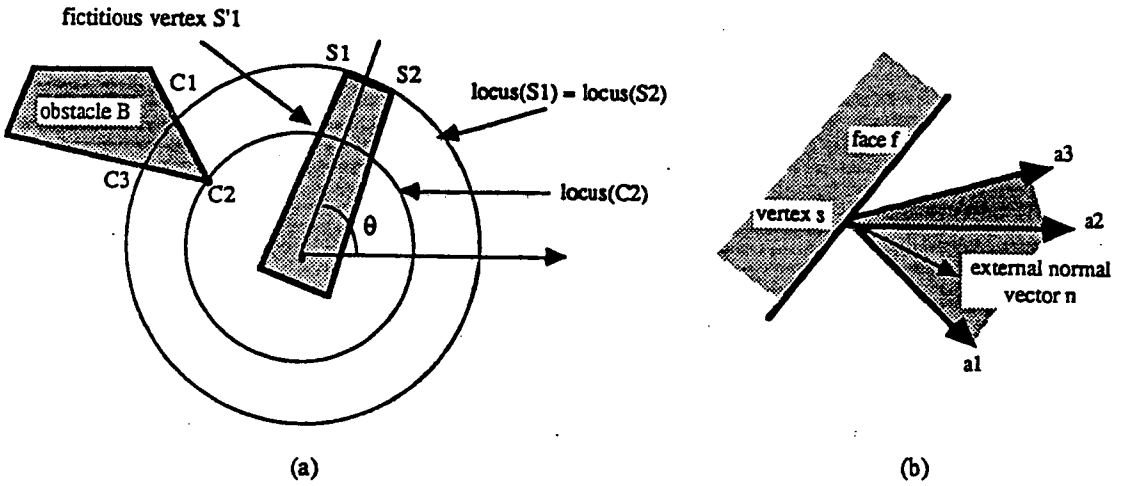


Figure 1: Computation of the contacts associated to a rotating object. (a) Potential contacts in a 2D exemple: (s_1, c_1), (s_2, c_2), (s'_1, c_2). (b) Contact (s, f) locally valid: $a_1 \cdot n \geq 0$ *and* $a_2 \cdot n \geq 0$ *and* $a_3 \cdot n \geq 0$.

by the vertices and the edges of A when A is translating or rotating. Each computed intersection is first considered as a potential contact between A and B (see figure 1). Then, very simple computations based on the vectorial calculus are applied in order to eliminate obviously wrong solutions (see figure 4.2).

This approach does not guarantee the faisability of the whole set of computed contacts when objects are not convex. In this case, the remaining ambiguities are processed by the last step of the algorithm which leads to construct the sets $CO^q_A(B)$ representing 1-dimensionnal C-obstacles in C^q_A along with their associated valid ranges of values $V_A(q)$ for $q \in D_q$ [32]:

$$V_A(q) = D_q - \cup_{i=1}^{n} CO^q_A(B_i)$$
$$CO^q_s(Bi) = \{q : s(q) \cap Bi \neq \emptyset\}$$
$$CO^q_s(Bi) = Closure\{\cup_{s \in A} CO^q_s(Bi)\}$$

where $CO^q_A(B_i)$ is a closed interval of $\Re$ representing the forbidden configurations of A defined by two consecutive contacts, and $s(q)$ is either a straight line (for translation) or a circle (for rotation). The figure 2 illustrates this computation. The algorithmic complexity in both the 2-dimensionnal and the 3-dimensionnal cases, may be represented by a term $\mathcal{O}(n^2)$, where n is the medium number of edges of A and B (see [32]).

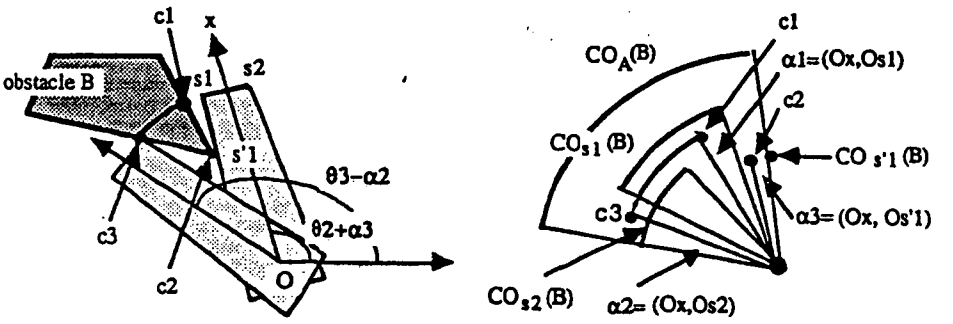


Figure 2: Computing $CO_A(B)$ associated with a polygonal rotating objects A.

4.3 Computing the C-obstacles associated with a rigid mobile:

Lozano-Perez shown in [38] that the C-obstacles of the type $CO_A^{xyz}(B)$ can be computed in a polyedral world in time $O(n^2 \log n)$, if n is the number of vertices of A and B. The applied method basically consists in constructing the convex hull of the set of vertices obtained by positionning the mirror image $\ominus A$ of A on the vertices of B (see figure 3):

$$CO_A^{xyz}(B) = conv(\{s \in \ominus A(s_i),\ \forall s_i \in B\})$$

But this method fails when A can also rotate, because the related C-surfaces are topologically very different from the initial real surfaces. In order to adapt his method to rotating objects, Lozano-Perez [38] makes use of several *slices of orientation* for computing approximations of the C-obstacles. But his approach is not really applicable when the moving object is an articulated revolute arm.

An other approach initially developped in [29] and in [39], consists in computing the sets of *valid orientations* of a translating and rotating object A, after having discretized the translation domain. This approach leads to construct 1-dimensionnal slices in C_A (i.e slices having one d.o.f in rotation), by applying the following operator [32]:

$$C_A^q = Reduction(C_A^{D_q})$$

where D is a domain in $\Re^2$ or $\Re^3$, and q is an angular sector. This operator leads to compute the ranges of orientation of A which generate no collision with the obstacles B, when the reference point of A moves in D. It can be expressed using the following functions:

$$\forall p \in D: \quad F_{A,B}(p) = \{\alpha \in [a\ b] :\ A(p,\alpha) \cap B = \emptyset\}$$
$$G_{A,B}(D) = \bigcap_{p \in D} F_{A,B}(p)$$

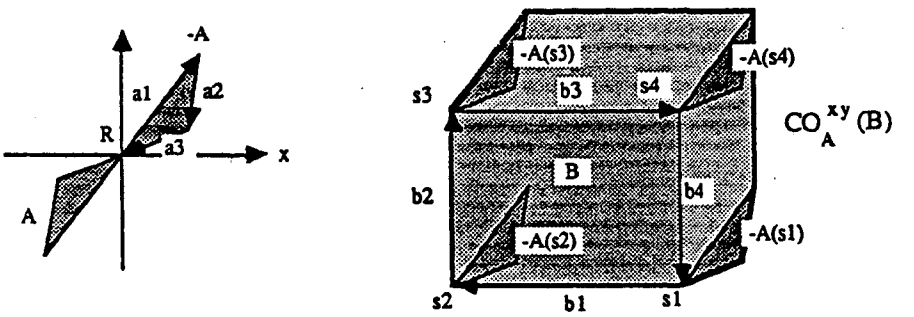


Figure 3: Construction of a C-obstacle of the type $CO_A^{xy}(B)$.

where (p, α) is a configuration of A in $C_A^{D_q}$. The function $F_{A,B}(p)$ can be evaluated using the algorithms described in the previous section, but the function $G_{A,B}(D)$ cannot be directly computed since it represents the intersection of an infinite number of sets. A practical method for computing an approximation of $G_{A,B}(D)$, consists in reducing D to a point while expanding obstacles inversely to the shape of D:

$$G_{A,B}(D) = F_{A,B'}(r)$$

with:

$B' = CO_D^{xyz}(B)$.

r = reference point of D.

This method is illustrated by the figure 4. Its algorithmic complexity is in time $O(p+q)$, if p and q represent the terms associated to the growing transform and to the computation of the valid ranges of values.

4.4 Dealing with articulated mechanisms:

4.4.1 A method for constructing C-obstacles:

Our approach for dealing with an articulated mechanism A of the type "open kinematic chain", consists in successively analysing the constraints imposed by the obstacles on the different componants A_i of the mechanism. For that purpose, the joints of A are ordered from the fixed extremity of A towards the free one. Then it becomes possible to apply a recursive algorithm, in which each step leads to study the behavior of a componant A_i $(i = 1, 2 \cdots n)$, after having fixed the positions of the anterior d.o.f q_j $(j = 1, 2 \cdots i-1)$ to a set of chosen values $(v_1, v_2, \cdots v_{i-1})$. At this step, the posterior componants A_k

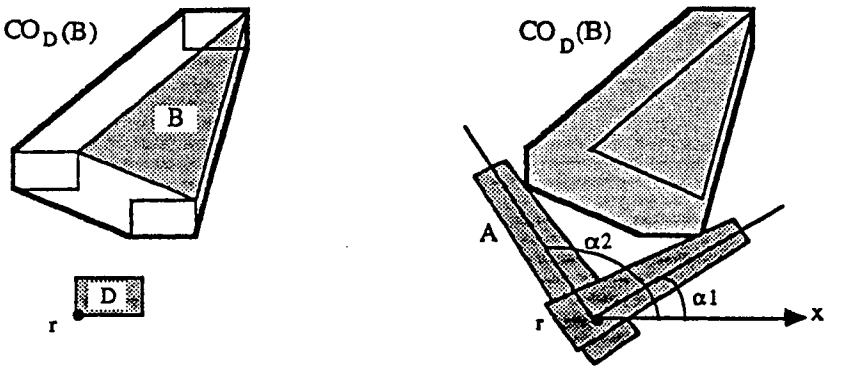


Figure 4: Computing the valid ranges of orientation associated to a translating and rotating polygonal object. (a) Determination of the set $CO_D(B)$. (b) Determination of the valid ranges of orientation of A relatively to $CO_D(B)$: $[0\ \alpha_1] \cup [\alpha_2\ 2\pi]$.

$(k = i + 1 \cdots n)$ are ignored by the system. This computation is applied for each small interval of values obtained by sampling the domains D_j $(j = 1, 2 \cdots i - 1)$ associated with the joint variable q_j.

The trajectory planning methods described in [18], [14], [29] and [39] are based on this approach. Even if the applied computations and the constructed free-space representations are different, all these methods lead (1) to consider the C-space as a set of hyperparallelepipeds of the type $dq_1 \times dq_2 \cdots dq_n$, and (2) to compute an approximation of C-obstacles in this set using the recursive algorithm illustrated in the figure 5. In this example, $CO_A^{D_2}(B)$ represents the conjonction of the constraints imposed by B on the three joints of A, for all $q_1 \in D_2$. This set may be seen as the projection of the part of $CO_A(B)$ located between the planes P_1 and P_2 of C_A (a slice). Finally, the interval Q represents the values of Q_3 which may generate a collision with B, when $q_1 \in D_2$ and $q_2 \in D_4'$. Using this computation, $CO_A(B)$ can be approximate by the union of all the sets of the type: $D_i \times D_j' \times Q_k$.

In our system, the recursive algorithm which compute the C-obstacles is based on two techniques allowing to respectively *compute the position constraints* generated by the B_i, and to *propagate these constraints* along the robot arm.

4.4.2 Computing the position constraints:

This computation is done using the techniques described above. For that purpose, the local behavior of each link A_j $(j = 2, 3 \cdots n)$ is analyzed after having fixed the position of the anterior links $A_1, A_2 \cdots A_{j-1}$, and after having associated a small variational domain

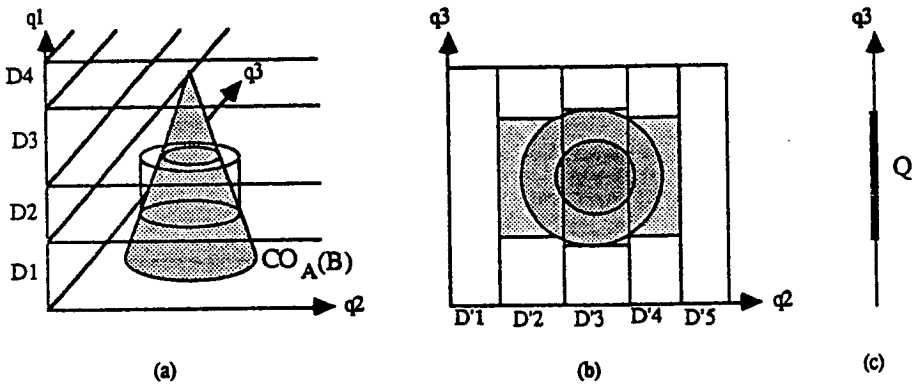


Figure 5: Computing an approximation of $CO_A(B)$ for a three joints robot A. (a) Constructing the slices associated to q_1. (b) Representing the approximation of $CO_A^{D_2}(B)$ associated with the slice D_2. (c) Representing the approximation of $CO_A^{D_2 \times D'_4}(B)$ associated with the slices D_2 and D'_4.

dq_{j-1} to the joint variable q_{j-1}. Let R_j be the reference point of A_j (located on the joint axis for symplifying the computations), and D_j the locus of the positions of R_j when q_{j-1} takes all the values in dq_{j-1}. D_j is either an arc of circle or a straight line, depending on the type of the joint A_{j-1}.

These hypotheses lead to locally associate two combined motions to the link A_j: the translation along D_j, and the movement (translation or rotation) associated to the joint A_j. Consequently, it becomes possible to "locally" approximate the involved subset of the C-space by a less dimensionnal space of the type $C_{A_j}^{D_j \times q}$, where q is the variational domain associated to q_j. Then, this representation can be transformed in a more useful representation of the type $C_{A_j}^{q}$, by applying the "reduction operator" leading to expand obstacles B_i inversely to the shape of D_j (see section 3.2.3). Finally, the computation of the valid ranges of values for q_j can be executed using the G function [32]:

$G_{A,B}(D_j) = F_{A,B'}(r_j)$
with:
$B' = CO_{D_j}^{xyz}(B)$
r_j = reference point of D_j

This approach gives good results as long as the dq_i are small enough. That means that too large computed domains have to be split into sets of small intervals, before beeing considered for the next joint of the arm. But in practice, we reduce the amount of geometric computations by applying two different steps (see next section):

1. Each link A_j $(j = 1, 2 \cdots n-1)$ is processed using a simplified growing transfor-

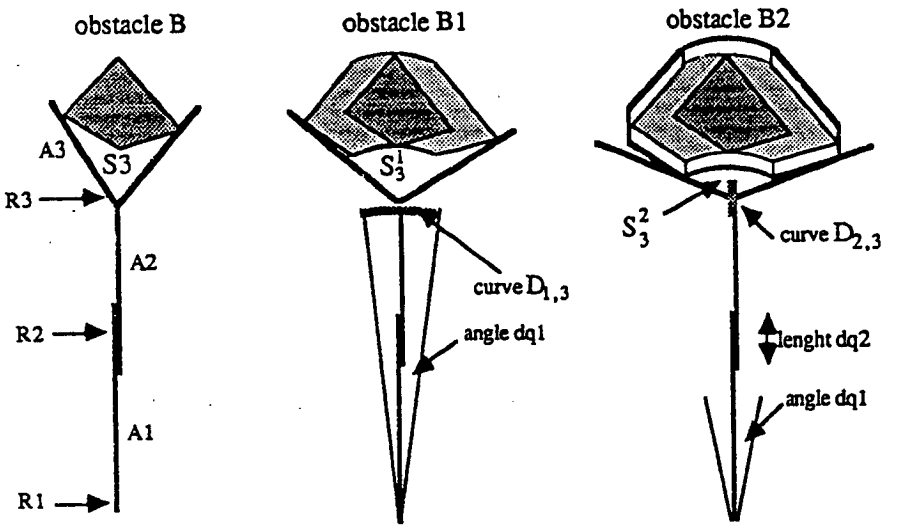


Figure 6: Propagating the position constraints along the arm.

mation and an interference checking function, both applied on a set small intervals dq_i (there exists one set for each considered slice of the type $dq_1 \times dq_2 \cdots dq_{j-1}$).

2. Link A_n is processed using the complete algorithm which leads to compute all the valid ranges of values associated to dq_n (i.e for each slice of the type $dq_1 \times dq_2 \cdots dq_n$).

4.4.3 Propagating the position constraints:

Since joints interact to each other, each set of position constraints associated to a joint A_j have to be propagated towards the next joints $A_{j+1}, A_{j+2} \cdots A_n$. That means that one must take into consideration all the previous growing operations, when computing the grown obstacles associated to the joint A_j. In order to simplify this computation, our system applies a recursive algorithm for expanding obstacles [32] (see figure 6):

$B^0 = B$
For $k = 1, \cdots j-1$: Compute $B^k = CO^{xyz}_{D^k_j}(B^{k-1})$

where D^k_j is the locus of R_j for $q_k \in dq_k$.

Each step of the algorithm leads to apply very simple growing transformations, since the related domain D^k_j is either a small straight line or a small arc of circle (which can be approximated by a straight line or a polygonal line).

Theoritically, the algorithm should be applied for each link A_j and for each slice of the type $dq_1 \times dq_2 \cdots dq_{j-1}$. But in practice, we reduce the amount of computation by determining an "upper bound" of the domains D_j, and by applying only one simple growing transformation for each slice of the type $dq_1 \times dq_2 \cdots dq_{n-1}$:

$$B' = Gros_r(B)$$
$$r = MAX_{i=1\cdots n}\ \{length(locus\ R_i)\}$$

where $Gros_r(B)$ represents an uniform expansion of B by the distance r. This approach may be extended to the whole space, when the splitting algorithm generates slices having an uniform size. Lozano-Perez [39] makes use of a similar approach by expanding the robot links with a distance equal to the maximum displacement computed using the jacobian matrix:

$$\delta = MAX_{Q \in I_1 \times I_2 \cdots I_n}\ \|J(Q) \cdot dQ\|$$

The complexity of the algorithm is in $O(N^{p-1})$, where N is the medium number of slices associated to a joint, and p is the number of d.o.f of the robot.

4.5 Constructing a free-space representation:

The free-space EL_A is composed of "cells" of the type $dq_1 \times dq_2 \cdots dq_n$. These cells are constructed using the following recursive function:

$$FREE\ (j,\ D_{i_1} \times D_{i_2} \cdots D_{i_j})$$
$$\Longleftrightarrow$$
$$FREE(j-1,\ D_{i_1} \times D_{i_2} \cdots D_{i_{j-1}})$$
$$and$$
$$A_j(q_i) \cap B = \emptyset,\ \forall q_i \in D_{i_j}$$

with:
$FREE(1,\ D_{i_1}) \Longleftrightarrow A_1(q_1) \cap B = \emptyset,\ \ \forall q_1 \in D_{i_1}$

In this algorithm, the domains D_{i_j} associated to the joint variables q_j ($j = 1, 2 \cdots n-1$) are obtained by splitting the variational domains I_j into small intervals; the sets D_{i_n} are computed using the G function. The next step consists in constructing a graph structure allowing to connect the "adjacent" free cells:

$$ADJACENT\ (j,\ D_{i_1} \times D_{i_2} \cdots D_{i_j},\ D'_{i_1} \times D'_{i_2} \cdots D'_{i_j})$$
$$\Longleftrightarrow$$
$$ADJACENT(j-1,\ D_{i_1} \times D_{i_2} \cdots D_{i_{j-1}},\ D'_{i_1} \times D'_{i_2} \cdots D'_{i_{j-1}})$$
$$et$$
$$D_{i_j} \cap D'_{i_j} \neq \emptyset$$

with:
$ADJACENT(1,\ D_{i_1},\ D'_{i_1}) \Longleftrightarrow D_{i_1} \cap D'_{i_1} \neq \emptyset$

4.6 Searching for a safe trajectory:

Each path in the free-space graph defines *one class* of safe trajectories for the robot and its payload. This class is represented by a sequence $(C_1, C_2 \cdots C_m)$ of connected free cells. Then, searching for a safe trajectory requires to apply two computational steps respectively aimed at finding an "optimal" path in the graph, and at choosing a "good" trajectory among those represented by the generated path.

The graph search step is executed using a A^* algorithm [46]. Several types of solutions may be obtained when modifying the characteristics of the cost function (see [32]). In SHARP, this cost function is based on an Euclidian distance allowing to minimize the end effector displacements. Let c_{init} be the intial configuration in the cell C_1, c_{end} the goal configuration in the cell C_n, c_i the configuration corresponding to the middle of the cell C_i, and $(C_1, C_2 \cdots C_j)$ the path computed at the step j. The costs associated to this intermediate solution are the followings:

length of the path: $d(c_{init}, c_2) + d(c_2, c_3) + \cdots d(c_{j-1}, c_j)$
distance to the goal: $d(c_j, C_{end})$

The determination of a trajectory in a path $(C_1, C_2 \cdots C_n)$ consists in choosing a "good" set of configurations (i.e as "short" and as "smooth" as possible), allowing to connect c_{init} to c_{end} within the subset of C_A represented by $(C_1, C_2 \cdots C_n)$. Since any cell C_i is an hyperparallelepiped in C_A, any couple of points located on the boundary of C_i may be connected by a *straight line in* C_A. Consequently, the problem to solve may be converted into a more simple problem, consisting in determining a set of suitable configurations on the boundaries shared by the consecutive couples of cells of the computed path. A straightforward solution is to choose at each step j the configuration located at the middle of the shared boundary $C_{j-1} \cap C_j$. However, this approach leads to generate inappropriate movements, especially when crossing through large cells. In order to partly avoid this problem, we have developped a method allowing to determine at each step j the configuration of $C_{j-1} \cap C_j$ which is the closest to the last computed configuration [32].

5 Morphological reasoning techniques:

5.1 Principle of the reasoning:

The purpose of this section is to describe the basic techniques which have been developped in SHARP, for reasoning on the contacts involved in the grasping and the part-mating operations. Three main characteristics of these contacts have been considered:

- *The local accessibility* of objects features. The problem to solve is to verify that the analyzed contacts are locally feasible according to some matter distribution criteria. The related computations evaluate a set of simple topological and geometrical properties.
- *The mechanical constraints* associated with contacts. The problem to solve is to verify that the analyzed solutions are valid according to a set of stability criteria. The related computations are mainly based on heuristics allowing to qualitatively evaluate some static properties.
- *The motion constraints* associated with contacts. The problem to solve is to determine which motions are locally possible in order to achieve (or to leave) the considered contact situation. The involved computations make use of an analytical representation of the potential motions that can be locally executed by the moving object.

The accessibility and the mechanical properties are mainly used for determining the potential grasping configurations, whereas the computation of the motion constraints is mainly executed in the context of fine motion planning.

5.2 Representing contacts:

The contacts between two objects O_1 and O_2 are represented as follows:

$$CONTACT(O_1, O_2) = \{((e_1, e_1')(e_2, e_2') \cdots (e_m, e_m')), T, P\}$$

where each set (e_i, e_i') is a couple of intersecting geometric entities, T is the type of the contact (point, line or surface), and P represents the geometric parameters which characterize the contact.

The geometric entities (GE) are those belonging to the B.R componant of the model: vertices, straight and curved edges, planar and curved faces (cylindrical, conical and spherical surfaces). The retained combinations for these entities are only those representing robust physical situations, i.e situations which can be potentially achieved despite the control and sensing errors. Since the couples of non colinear edges are not relevant

for the fine motion strategies, we will only take into consideration the contact relations of the type *face-X*, where X = face, edge or vertex. These couples are called *contact elements*. The other combinations of entities are considered as unsteady situations, corresponding to degenerated contacts called *adjacent contacts* (see figure 7). This type of contact is not physically feasible because of the control and sensing errors, but its symbolic determination allows to characterize the discontinuities generated by the geometry of the objects.

The type of the contact is defined by the topoly of the set of contact points (point, curve or surface). The characteristics of this set depend on the type, the number and the relative positions of the involved GE. For example, a planar contact may be obtained by combining two contact elements generated by two non linear straight edges and a planar face (see figure 7). This multiple contact may be assimilated to a planar contact, because it has similar static properties. In particular, the convex hull of its contact points determines a "sustentation polygon", which have the same stability properties as a contact having a similar convex polygonal surface [23]. Since there exits a small number of different cases, it is not necessary to compute the convex hull of the contact points for determining the type and the geometric characteristics of the contact. It is sufficient in our context to analyse the sets $e_i \cap e'_i$ along with their possible combinations (see [26] and [21]).

The geometric parameters characterize the contact by a couple $(S, \delta S)$, where S is the surface defined by the set $E_1 \cap E_2$ of contact points, and δS is the boundary of the set $conv(E_1 \cap E_2)$. S is called the *contact support*, and δS is called the *contact surface*. Intuitively, we will say that the mobile object can slide on S, without "breaking" the contact (i.e without modifying the topology of δS). This property is very useful for planning fine motions.

In some cases, S cannot be unambiguously defined by the set $E_1 \cap E_2$ (for example when the contact is generated by a planar face and a cylindrical face). In this case, we chose to consider that S is defined by the geometric entities of the fixed object (i.e the object which is not manipulated by the robot). As an example, let consider the contact (c) of the figure 7. If the cylindrical object is manipulated by the robot, then the contact will be characterized as follows:

```
(CONTACT ((FACE F1   FACE FC2))
         (TYPE droite)
         (SUPPORT plan (0 0 1))
         (BOUNDARY (P1 P2)))
```

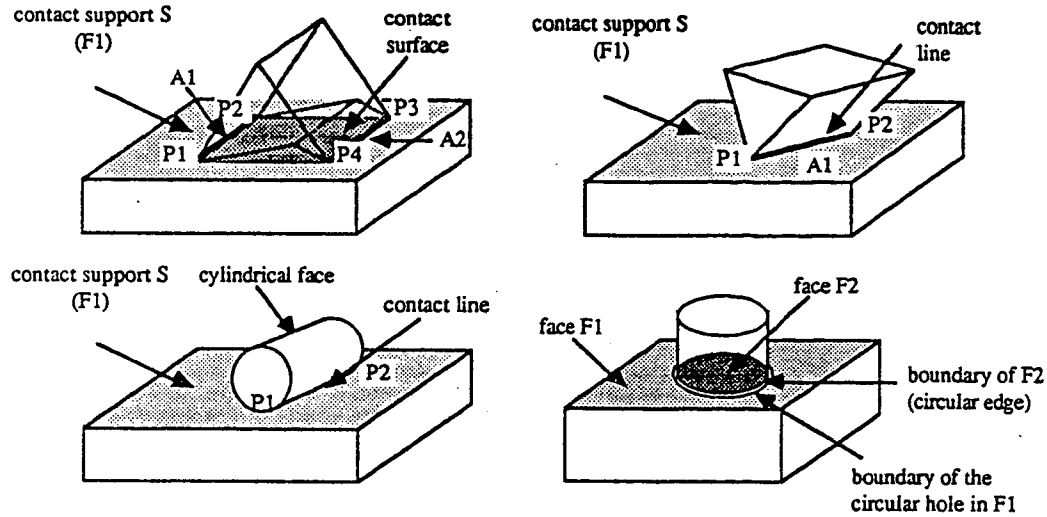


Figure 7: Example of some contacts: (a) Planar contact generated by a planar face and two edges. (b) Linear contact generated by a planar face and an edge. (c) Linear contact generated by a planar face and a cylindrical face. (d) Adjacent contact generated by two circular edges.

5.3 Reasoning on accessibility and mechanical properties:

The accessibility and the mechanical properties are evaluated in order to determine the combinations of contacts which are *locally accessible, stable and robust.* These combinations of contacts are used for constructing several sets of potential solutions for performing the grasp operations (see [25], [28], [54] and [32]). Such sets are called *potential grasps.*

Local accessibility property: One says that a combination of contacts is locally accessible, if the involved contacts are simultaneously feasible according to the local morphology of the two objects. Evaluating this property leads to verify that some simple topological and geometric constraints are verified in the vicinity of the features in contact: local convexity and P-convexity, parallelism and angular constraints, distance and dimensionnal constraints. All these properties are characterized by very simple algebraic formulas. For example, the local convexity of an edge A_1 belonging to two faces F_1 and F_2 is characterized by the formula $V_{A_1} \cdot (N_1 \wedge N_2) \leq 0$, where V_{A_1} is the edge A_1 oriented counterclockwise on the face F_1, and N_1 and N_2 are the external normal vectors of F_1 and F_2. In the same way, we will say that the edge A_1 is locally P-convex (i.e it is possible to place a plane P onto A_1, such as P do not intersect the object in the vicinity of A_1) if A_1 verify the following property:

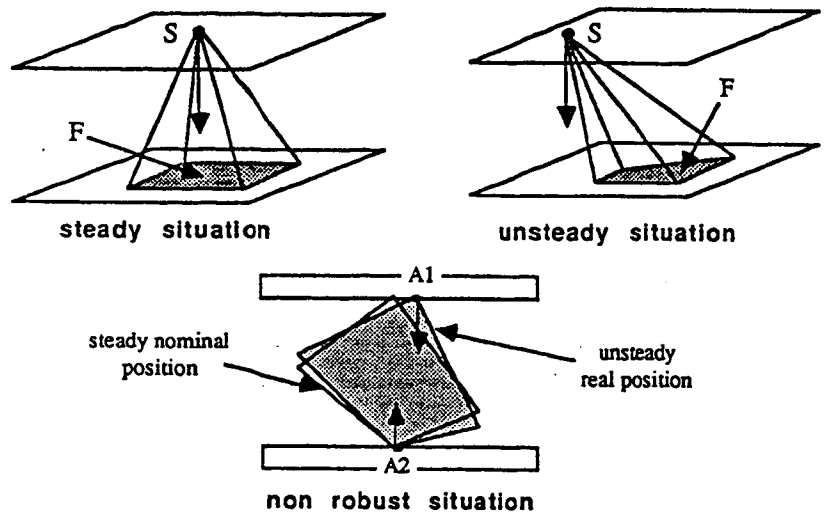


Figure 8: Stability and robustness properties. (a) Stable couple (S, F): $proj(S) \in F$. (b) Unstable couple (S, F): $proj(S) \notin F$. (c) Non robust couple (A_1, A_2): $proj(A_1) = proj(A_2) = straightline$.

$$V_{A_1} \cdot (N_1 \wedge N_2) \leq 0 \quad and \quad \exists a, b \in \Re : \; a < 0, \; b < 0, \; N_p = aN1 + bN2$$

where N_p is the external normal vector to P (P represents in this case the internal surface of one jaw of the gripper).

Stability property: This property allows to verify that the created physical situation is stable, i.e that the resulting forces and torques associated to the involved contacts are equal to zero. Instead of evaluating this property using an explicit representation of the forces and the torques, we chose to apply very simple geometric computations based on a qualitative knowledge of the static. The basic idea consists in verifying that the resulting reaction force of each contact "goes through" the contact surface δS of an other contact of the studied set. In the case of a set of contacts created by a two parallel jaws gripper, this condition may be evaluated by projecting the contact surfaces on a plane parallel to the jaws. If the obtained domains intersect, then the combination of "frictionless" contacts is considered as stable (see figure 8). This property is called the *mutual visibility property*, since it assumes that the contacts are located "in front of each other".

Robustness property: This property allows to verify that the combination of contacts remains stable, when slightly moving the object. Our approach for planning robust grasps (i.e grasping points which avoid the "twisting" of the object inside the jaws), consists in choosing a combination of contacts which generates no d.o.f along the directions

having some possible associated position errors. Consequently, we will eliminate the solutions which keep free a rotation around an axis not normal to the internal faces of the jaws, or not parallel to the revolute axis of some objects (see figure 8).

All the computations involved in the determination of the local morphological properties described above, have been implemented in SHARP using an ordered set of *geometric filters* (see [54] and [32]).

5.4 Reasoning on motion constraints:

5.4.1 The concept of potential motion:

Each contact reduces the number of d.o.f of the moving object (i.e the gripper or the manipulated object). In order to determine the next motion to execute from a given contact situation, the system must reason on the directions of movement 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 reasoning, 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 ds greater than the maximum control error, and which generates 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. This formalism is described below.

5.4.2 Representing potential translations:

A translation can be represented by a couple (v, a), where v is an unitary vector of $\Re^3$ and a is a real number. Then, a set of translations can be represented by a subset of $S(1) \times \Re$, where $S(1)$ is the unitary sphere. Since our purpose is to reason on sets of translation directions, we will only represent the related domains on $S(1)$. Then, a point on the sphere defines a particular translation direction, and a spherical domain characterizes a set of possible translating motions. For example, a planar contact generates an half-sphere domain, and a couple of contacts determines a domain obtained by intersecting those associated to each contact (see figure 9). Such a computation may be executed by projecting the constructed domains on a plane (φ, ϑ), where φ and ϑ are the spherical coordinates.

Contacts having a planar support:

When contacts are of the type "planar face - X" (X represents any type of GE), the computation of the resulting domains on $S(1)$ may be executed using the following

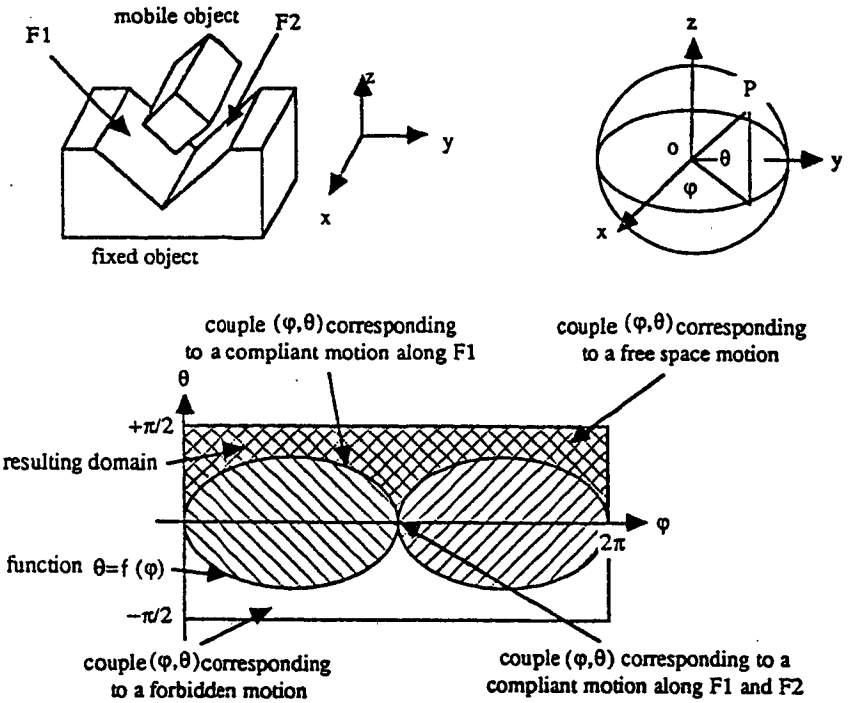


Figure 9: Representing the potential motions associated to a couple of planar contacts.

function [51]:

$$\vartheta = \arctan((N_x \cdot \cos\varphi + N_y \cdot \sin\varphi)/ - N_z) \tag{1}$$

where (N_x, N_y, N_z) is the normal external vector to the contact plane (see figure 9).

Contact having a curved support:

In this case, the applied computations depend on the type of the involved surfaces. For example, a cylindrical contact will only generate two possible translating motions along the revolute axis of the contact surface. On the other hand, a conical contact will generate a more complicated domain characterized by the function:

$$\cos\vartheta(a\cos\varphi + b\sin\varphi) + c\cos\vartheta - \cos\alpha = 0 \tag{2}$$

where (a, b, c) and α are the cone axis and the cone top angle.

Other types of contacts may also be generated by non planar entities. But it is possible to apply the same type of computation, by using a first order approximation of the potential movements. This approximation is obtained by considering the fictitious planar contacts defined by the tangent planes associated to the contact surfaces [32].

For example, a contact generated by two partial cylindrical surfaces defines a domain of the type "$D(P_1) \cap D(P_2) \cap D(P_3) \cup D(cylinder)$", where $D(P_1)$, $D(P_2)$ and $D(P_3)$ are the domains associated to the three tangent planes located on the boundaries of the contact surface, and $D(cylinder)$ represents the two directions associated to the cylinder revolute axis.

5.4.3 Properties of the representation:

Let $S = \{C1, C2, \ldots Cn\}$ be a contact situation, and C_i the related contacts.The potential motions associated to S are represented by a domain D. This domain is defined as the intersection of the domains $D_1, D_2 \cdots D_n$ associated to the contacts $C_1, C_2 \cdots C_n$. Each D_i is computed using the function (1) or (2). Let D_i^* be its inside part, and δD_i its boundary ($D_i = D_i^* \cup \delta D_i$).

An important property of this representation is to clearly differenciate the different types of motions which can be associated to S (see figure 9):

- Each couple (φ, ϑ) outside of D represents a forbidden motion.
- Each couple (φ, ϑ) in D represents a motion which leads to break all the contacts.
- Each couple (φ, ϑ) on δD represents a compliant motion which leads to slide on some contact surfaces: if $(\varphi, \vartheta) \in \delta D_{i_1} \cap \delta D_{i_2} \cap \cdots \delta D_{i_j}$ and $(\varphi, \vartheta) \notin \delta D_k (k \neq i_1, i_2 \cdots i_j)$, then (φ, ϑ) defines a motion which maintains the contacts $C_{i_1}, C_{i_2} \ldots C_{i_j}$ (the other contacts are broken).

An other property of this representation is to be *complete* for contacts having a planar support S. Then, any existing solution is contained in D, but it is impossible to guarantee that this solution will be found in a finite time. Our approach for dealing with this problem is described in the next section.

The completeness property is not preserved when the support of the contact is not planar surface. But the applied first order approximation is reasonnable, since it allows to locally represent the potential compliant motions as a set of tangential motions. This approach is consistant with the fact that such motions are always executed tangentially to the surfaces which support the contacts.

5.4.4 Dealing with rotations:

Dealing with rotations is more complicated, since the content of the potential rotating domains depends on both the orientations and the positions of the rotation axes. For example, a rotation axis belonging to the contact plane but located outside of the contact surface δS, will generate motions leading to break the contact; the same axis located tangentially to the boundary of δS will generate motions allowing to modify the type of the contact.

This characteristic of the rotations makes the previous approach impractical for the potential rotating motions, since several representations may be associated to a single contact. On the other hand, it suggests to develop an other approach consisting in grouping together the rotation axes having an "homogeneous behavior" relatively to the contacts. For example, we will group in a single set all the rotation axes which generate motions allowing to maintain the analysed contact. In the processed contacts, these sets are easilly derived from the type of the contact surfaces. For example, the rotation axes which maintain a planar contact are those which are normal to the contact plane; those which maintain a cylindrical or a conical contact are located on the revolute axis of the surface.

This approach allows to easily deal with a large set of practical cases. However, it needs to be developped in a more complete way in the future.

5.5 Constructing the solution space:

5.5.1 The set of potential grasps:

A potential grasp represents a set of grasping positions based on the same combination of contacts. In order to guide the selecting process, a partial order based on an heuristic evaluation of the "quality" of the involved combination of contacts is established among the potential grasps. This approach allows to first analyse the potential solutions which are more likely to generate a feasible grasping position. But it does not allow to generate a sequence of different grasping positions based on several combination of contacts, when a single solution cannot be found by the system. Such a method (called regrasping) have been developped in [41].

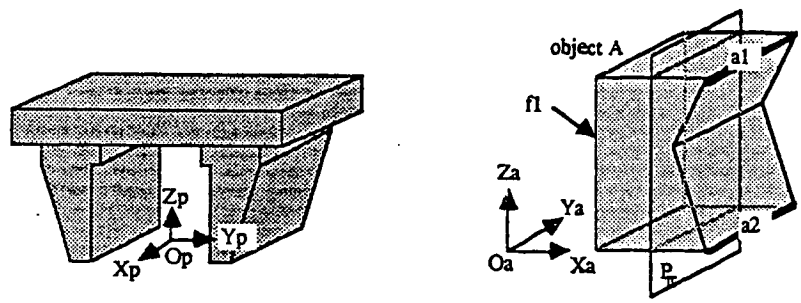
Representing potential grasps:

Let A be the object to be grasped and P the robot gripper. A potential grasp $\Pi(A/P)$ is represented as follows [28] [54] (see figure 10):

$$\Pi(A/P) = (\pi, P_c, P_m, h)$$

where $\pi = (C_1, C_2 \cdots C_m)$ is the set of the involved contacts, P_c represents the "preshaping" parameters of P (i.e the configuration of P before grasping and the jaws action to execute), P_m represents the orientation and the moving constraints associated to P, and h is an heuristic assessment of the *quality* of the solution (stability and robustness) and of its *degree of accessibility.* At this step, all the geometric parameters of the contacts C_i are not specified, since the position of P have not been already computed by the system.

As we make the assumption that the grasping operations are executed using a two parallel jaws gripper, P_m may be characterized by a plane called the *gripping plane* P_π. This plane is parallel to the two planar contact surfaces (i.e the internal faces of the jaws), and it is located at an equal distance to this two surfaces. Then, the constraints P_m can



Π (A/P) = ((((a1 a2) plane) ((f1) plane)
(close 1.5)
((0.75 0.0 0.0) (1.0 0.0 0.0)) h)

Figure 10: Example of a potential grasp.

be characterized by the expression "$O_P \in P_\pi$ and $X_P O_P Z_P \,//\, P_\pi$", where $(O_P\, X_P\, Y_P\, Z_P)$ is the reference frame associated to P as shown in the figure 10.

Searching for a solution in $\Pi(A/P)$:

Searching for a solution in $\Pi(A/P)$ consists in determining a particular configuration of P which verify the constraints represented by π and P_π, and which can be realized in the initial and the final environments of the object to be grasped. Let $P(X, \Delta)$ be the configuration of P corresponding to $O_p = X$ and $O_p Z_p = \Delta$, W the set of GE belonging to A and to the obstacles located in the initial and the final environment of A, and E the set of GE involved in π. The configuration $P(X, \Delta)$ is unambiguously defined by the couple (X, Δ), since $X_p O_p Z_p$ is parallel to P_π.

Property 5.1 *Let p be a point in P_π and Δ a vector in P_π. A couple (p, Δ) will be considered as a particular solution in $\Pi(A/P)$, iff:*

- $P(p, \Delta) \cap e \neq \emptyset, \quad \forall e \in E$
- $P(x, \Delta) \cap \{W - E\} = \emptyset, \quad \forall x \in half\text{-}line(p, \Delta)$.

The first expression verifies that all the contacts of π are simultaneously realized in the configuration $P(p, \Delta)$. Such a configuration we be considered as *valid*. The other expression verifies that the involved object features can be reached along the direction Δ. Such a configuration will be considered as *safe*.

Then, searching for a particular solution in $\Pi(A/P)$ can be done using three main steps: (1) choose a direction Δ in P_π, (2) construct the set S_π of the points $p \in P_\pi$ such as (p, Δ) verify the property 5.1, and (3) choose a point p in S_π such as $P(p, \Delta)$ minimizes the risks of sliding. This method is detailed in section 6.2.

5.5.2 The state graph associated to the fine motions:

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. This graph does not explicitly contain all the possible solutions, but we will see further that it may be locally refined in case of failure. Such a failure occurs during the search phase, and it means that no feasible fine motion strategy is represented in the current state graph.

Representing the state graph:

Let A and B be the two objects to assemble. The state graph $G(A/B)$ associated to the fine motions allowing to assemble A on B, is represented by a directed graph $(\mathcal{S}, \mathcal{A})$. Each node in $\mathcal{S}$ represents a robot state E_p (see section 3.1), and each arc in $\mathcal{A}$ defines a robot motion allowing to move from a state to an other one.

A node s_i in $\mathcal{S}$ is characterized as follows:

$$s_i = (P, C, D, I, Q)$$

where P is a set of positions for the robot, C is a set of contacts, D is a set of potential motions, I is a qualitative information on the associated physical situation (for example: a cylindrical insertion), and Q is an heuristic assessment of the "quality" of this situation (robustness relatively to uncertainties and mechanical faults). P and C characterize the state E_p. D defines the motions which can be theoritically applied from E_p (this information is used when refining the graph). I and Q are used for guiding the search function.

An arc a_{ij} in $\mathcal{A}$ is characterized as follows:

$$a_{ij} = (T, C, A, Q)$$

where T is a geometric transform, C and A are two sets of faces on which the mobile object must respectively slide and stick, and Q is an heuristic weight evaluating the "quality" of the motion (collision and sticking risks when slightly modifying the environment, effects of the gravity ...). The parameters T, C and A characterize the motion allowing to move from E_{p_i} to E_{p_j}. The parameter Q is used for guiding the search function.

Constructing the state graph:

Our analytic representation of potential motions allows to characterize the whole set of movements which are potentially feasible from a given state E_i. 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" analyzed 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 [8].

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 analyzing 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. In particular, most of fine motions can be executed along a direction normal or parallel to the surfaces located in the vicinity of the involved contacts. This is the reason why the selected directions correspond to some characteristic points in D: points located at the intersections of several curves of the type 1 and 2, points periodically distributed on the boundaries of D.

The algorithm used for constructing the state graph is described in section 6.3.

Searching for a solution in $G(A/B)$:

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. Each node in IS is considered as a possible starting point for constructing a fine motion strategy, because the related physical situation can be directly achieved by the robot.

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 "good path" SG starting from a node s in IS and ending at node GS.
2. Verify that each arc of SG generates a movement which can be executed by the robot (no collision, reachability of the goal). Refine $G(A/B)$ in case of failure, and goto (1).
3. Synthesize the fine motion program from SG.

This algorithm is detailed in section 6.3. In the general case, SG is a sub-graph which includes at each level all the states (nodes of $G(A/B)$) which can be reached after executing the selected motion. This situation comes from the fact that the control errors may sometimes make the robot stop in different contact situations. This point is discussed in [32]. It is not taken into consideration in the current search function, since the states and the motions selected when constructing the state graph avoid such situations.

6 Implementation of the motion planners:

6.1 The trajectory planner:

As explained in section 4, our method for computing *safe trajectories* operates in two phases allowing to successively construct a graph representing an approximation of the free-space, and to search this graph using a A^* algorithm. The main originality of the method lies in the fact that the computation of the free-space representation is done using a constraint propagation mechanism allowing (1) to recursively apply very simple growing transformations depending at each step on the type of the analyzed joint, and (2) to parametrize the algorithm according to several constraints drawn from both the environment and the mechanical structure of the arm (see [29]). In the current implementation, the parameters which evaluate the cluttering of the workspace have been arbitrarily fixed to constant values (i.e the discretization of each d.o.f is done using an uniform predifined value). As explained in section 4.4.3, this approach allows to reduce the amount of computation by applying a single growing transformation of the type $Gros_r(B)$ for each slice of the type $dq_1 \times dq_2 \cdots dq_{n-1}$. Then the missing domains dq_n associated to these slices are computed using the algorithms described in section 4.2 and 4.4.2. The search phase is executed as described in section 4.6.

This method can theoritically be applied for the whole arm. But in practice it is only used for planning the wrist trajectories, because of its exponential complexity in terms of the number of d.o.f. Then, the required rotations of the wrist are determined by applying two complementary planning steps based on heuristics [32]:

- Determination of the wrist movements (with a fixed orientation) in the vicinity of the initial and of the goal positions.
- Determination of the rotations of the wrist along the computed trajectory, by considering the volume swept by the gripper and the payload when rotating.

This approach works fairly well when the manipulated object is small relatively to the gripper. It necessitates to execute the reorientation operations in some predifined uncluttered areas, when the size of the swept volume makes the algorithm fail.

The trajectory planner have been implemented in LUCID-LISP on a SUN 260. Most of the experimentations have been executed in simulation (see figure 11). Some of them have give rise to real executions using a six d.o.f SCEMI robot. But the CPU time required for computing the free-space model in the current version of the system is not really significant, because most of the geometric computations are performed using some external functions belonging to the modelling system (these functions have not been optimized and compiled). For instance, 28 mn of CPU time was needed for processing the example shown in figure 11. This example has give rise to a graph having 480 free cells of the type $5 \times 10 \times 15$, where 5^o, 10^o and 15^o are the angular sectors which have been used for discretizing the joint space.

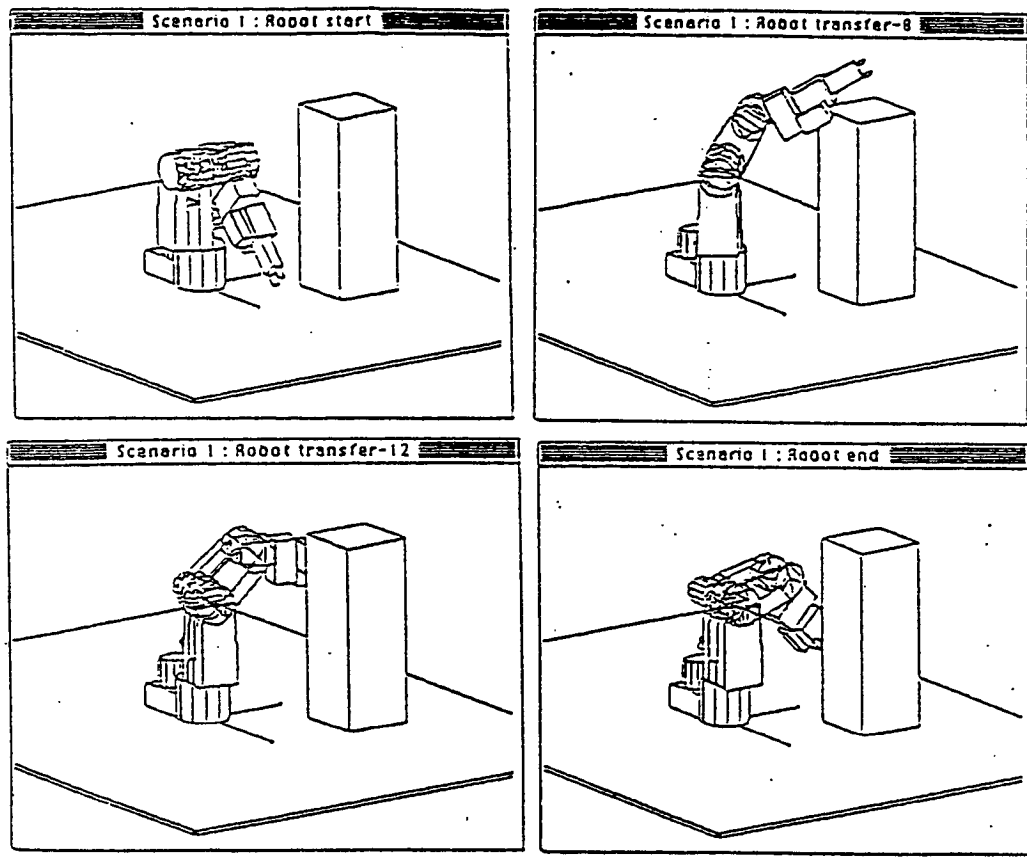


Figure 11: A simple trajectory computed by the system.

6.2 The grasp planner:

As explained in section 3.2, our method for computing *secure grasps* operates in two phases allowing to successively compute a set of potential solutions, and to determine the dynamic parameters of the chosen solution.

The purpose of the first phase is to determine the contact sets which are locally accessible, stable and robust. This computation is executed using an ordered set of geometric filters implementing the accessibility and the mechanical properties described in section 5.3. This approach has been developped in order to reduce the size of the search tree, by first applying the more discriminant filters. The resulting potential solutions are represented as shown in section 5.5.1.

The second phase verifies that the generated potential solutions are reachable in both the initial and the final environments. It determines for that purpose the gripper con-

figurations which are both *valid* and *safe*, according to the position constraints imposed by the environment (see section 5.5.1). This computation allows to reject the unfeasible solutions and to determine the missing parameters of the selected one (position and orientation of the gripper, approach and deproach directions).

The implemented algorithm is the following:

- **Phase 1**

  1. Determination of the potential contacts.
  2. Determination of the potential combination of contacts and construction of the related potential grasps of the type $\Pi(A/P)$.

- **Phase 2**

  1. Choice of a potential grasp $\Pi(A/P)$ compatible with the task constraints.
  2. Choice of a direction Δ in the gripping plane P_π.
     If all the directions have been analyzed, then goto (1).
  3. Determination of the valid configurations $P(X,\Delta)$ in $\Pi(A/P)$:

     $$V_\pi(\Delta) = \{p \in P_\pi :\ P(p,\Delta) \cap e \neq \emptyset,\ \forall e \in E\}$$

     If $V_\pi(\Delta) = \emptyset$, then goto (2).
  4. Determination of the safe configurations $P(X,\Delta)$ in $\Pi(A/P)$:

     $$S_\pi(\Delta) = \{p \in V_\pi(\Delta) :\ P(p,\Delta) \cap \{W - E\} = \emptyset\}$$

     If $S_\pi(\Delta) = \emptyset$, then goto (2).
  5. Choice of a point p in $S_\pi(\Delta)$, such as $P(p,\Delta)$ maximizes the surface of contact and minimizes the torque created by the gravity.

This algorithm is completed by a complementary step allowing to choose a gripping plane, when the grasping operation is executed on a revolute surface. The choice of the direction Δ is made using an heuristic which determines the free angular sectors associated to the center of gravity of the object to be grasped (see [27]). The computation of the sets $V_\pi(\Delta)$ and $S_\pi(\Delta)$ can be executed using the classical growing transformation CO^{xz} [55]:

$$V_\pi(\Delta) = \cap_{e_{ij} \in E}\ Proj_{P_\pi}(CO^{xz}_{F_i[\Delta]}(e_{ij}))$$

$$S_\pi(\Delta) = V_\pi(\Delta) - Proj_{P_\pi}(CO^{xz}_{P[\Delta]}(B))$$

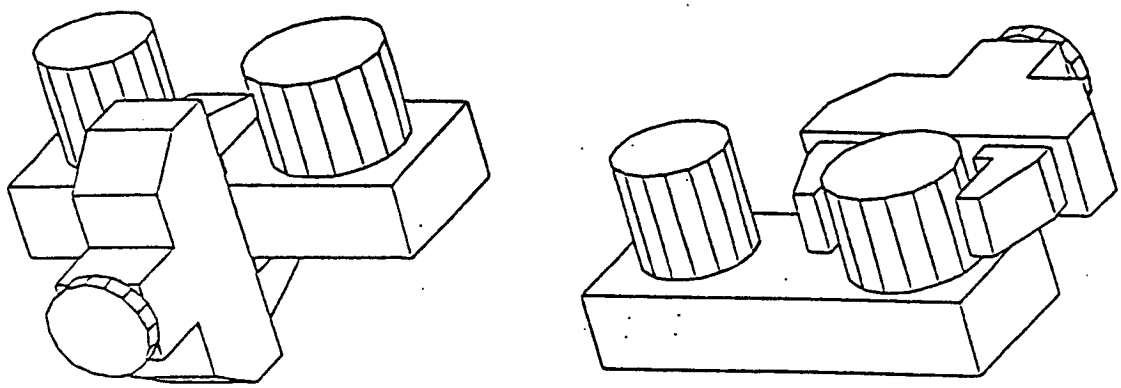


Figure 12: Two grasping positions computed by the system.

where $Proj_{P_\pi}(X)$ is the orthogonal projection of X on P_π, $F_i[\Delta]$ is the internal face of the jaw i oriented according to Δ, $P[\Delta]$ is the gripper oriented according to Δ, and B represents the obstacles. In our implementation, this computation is executed in a two dimensionnal space, after having projected on the gripping plane several slices delimited by a set of parallel planes (see [27]). We have also developped a specific growing transformation CO^* allowing to reject the points in $S_\pi(\Delta)$ which cannot be reached by the gripper because of the robot arm. This new transformation consists in expanding the set $CO^{xz}_{P[\Delta]}(B)$ in the direction $-\Delta$, in order to include the "shadow" of the object (see [27]).

The grasp planner has been first implemented in MACLISP on a CII-HB 70 computer. Its integration in the SHARP system has been partly realized, but several experimentations have been done in simulation with simple objects having less than 20 faces. For instance, less than one minute of CPU time was needed for computing the solutions shown in figure 12, and only one second was consumed by the first phase for generating 12 potential grasps among 32! possibilities. But this execution time increases very fast when the system has to take into consideration the surrounding obstacles.

6.3 The fine motion planner:

As explained in section 3.2, our method for computing the *fine motion strategies* operates in two phases allowing to successively construct a state graph representing the set of potential solutions, and to search this graph in order to find a "good path" representing a feasible fine motion program.

The analysis phase allows to construct 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 moving constraints associated to the contacts are examined at this step (the other motion constraints are not considered). This computation is executed using the method described in sections 5.4 and 5.4.2. It 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 efficiency (number of operations) and of 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 (see [31]).

The implemented algorithm is the following:

- **Analysis phase**

  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).

- **Search phase**

  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.

$OPEN$ represents the list of the next nodes to process, and D_x is the set of potential motions associated to the node x (see section 5.4). 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 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.

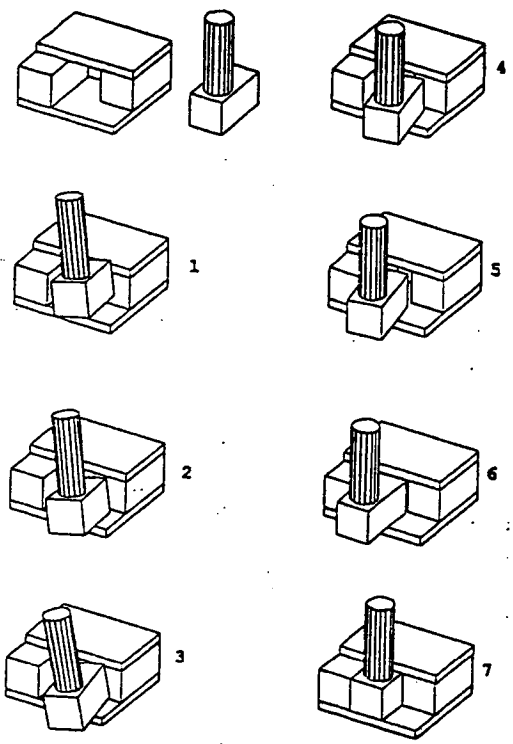
Each node in $G(A/B)$ is created after having applied a geometric computation allowing (1) to determine the involved contacts (parameter C), (2) to compute the associated set of potential motions (parameter D), and (3) to verify that the reached state E_i does not already exist in $G(A/B)$ and that it can be unambiguously distinguished from the other states represented in $G(A/B)$. Each arc in $G(A/B)$ is created after having applied a geometric computation allowing (1) to determine the valid ranges of position associated to the selected moving direction (parameter T), and (2) to determine all the contacts which are involved in the movement (parameters C and A).

The search phase is guided by a *cost function* which is combined with a set of *dynamic pieces of advices* implemented using production rules (see [31] and [51]). The cost function makes use of the heuristic weights associated rcp -r /usr2/vision/fano/thesis.dir mars: /fano/.to the nodes and to the arcs of $G(A/B)$. It leads to both minimize the number of operations and to maximize the reliability of the selected motions. The dynamic pieces of advices are activated when some particular situations are detected by the system. They mainly allow the system to deal with the physical situations which cannot be safely executed by the robot because of the position uncertainty (adjacent contact, closed obstacle ...). For example, if a contact situation cannot be directly left because of the presence of an obstacle in the vicinity of the moving part, then it is advised to first slide on the contact surface (and consequently to create new nodes and new arcs in the graph). In the same way, an adjacent contact in SG will lead the system to determine a set of intermediate contacts for guiding the robot. This approach allows to reduce the algorithmic complexity by only exploring "in detail" the branches of the graph which are really significant according to the selected solution. It also permits to integrate some well-known local strategies in the solution, when classical situations are recognized by the system (for instance: cylindrical insertion).

The collision and the reachability tests are executed using the following computations [32]:

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

where $\varepsilon = 2(\varepsilon_p + \varepsilon_i)$ and the terms ε_p and ε_i represent the control and the sensing error bounds; $Sweep(A,d)$ is the volume swept by A when moving along d, $Gros_\varepsilon(B)$ and




```
(REALISER-S
    (R-robot SUR R0)
    (VIA (R1 SUR R2)(R3 SUR R4) ..... (Rl SUR Rl)))

(REALISER-C
    (point-P1 SUR face-B1)
    (SUIVANT (TRANSLATION :VECT vecteur-V1)))

(REALISER-C
    (arete-A1 SUR face-B1)
    (SUIVANT (ROTATION  :AXE (make-axe    :PT point-P1
                                          :VECT vecteur-V2)))
    (EN-MAINTENANT (point-P1 SUR face-B1)))

(REALISER-C
    (face-F1 SUR face-B1)
    (SUIVANT (ROTATION  :AXE (make-axe    :PT point-P1
                                          :VECT arete-A1)))
    (EN-MAINTENANT (arete-A1 SUR face-B1)))

(REALISER-C
    (arete-A2 SUR face-B2)
    (SUIVANT (TRANSLATION :VECT vecteur-V3))
    (EN-MAINTENANT (face-F1 SUR face-B1)))

(REALISER-C
    (face-F2 SUR face-B2)
    (SUIVANT (ROTATION  :AXE (make-axe    :PT point-P2
                                          :VECT arete-A2)))
    (EN-MAINTENANT (face-F1 SUR face-B1)(arete-A2 SUR face-B2)))

(REALISER-C
    (point-P3 SUR face-B3)
    (SUIVANT (TRANSLATION :VECT arete-A1))
    (EN-MAINTENANT (face-F1 SUR face-B1)(face-F2 SUR face-B2)))
```

Figure 13: A fine motion strategy computed by the system.

$Gros_{\varepsilon}^{-1}(B)$ respectively represent the obstacles B grown and shrunk according to ε, and $A(p)$ is the object A in the configuration p.

The last step consists in computing the missing motion parameters of the selected movements, 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 predifined 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.

The fine motion planner has been implemented in LUCID-LISP on a SUN 260. Most of the experimentations have been executed in simulation. Some of them have give 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 13 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 50 nodes and 80 arcs.

7 Conclusion:

In this paper we have shown that two types of geometric reasoning techniques have to be combined for planning the various robot motions involved in the basic assembling operations (transfer, grasping and part-mating operations). We first described the various computational tools that we have developped for implementing these two types of reasoning (called *spatial reasoning* and *morphological reasoning*). Then we explained how the proposed methods have been implemented and integrating in a single system (the SHARP system), aimed at automatically generating assembly robot programs. The obtained results and the current limitations of our approach have also been discussed. This approach has been partly validated by several experimentations executed in simulation, but very few results have been currently obtained in connection with the real robot.

Current work deals with three major points: the unsolved geometric problems mentioned in the paper (dealing with rotations for example), the algorithmic complexity (how to combine local and global reasoning techniques in order to reduce this complexity without increasing the risk of failure), and the connection between the geometric model of the task and the real robot (how to automatically compute the numerical values which are required for executing the planned motions).

Acknowledgements:

The work presented in this paper has been done by the members of the Robotics group at the LIFIA Laboratory. It was partly supported by the French National ARA project of the CNRS, the ADI agency, and the INRIA institute.

References

[1] N.Ahuja, R.T.Chien, N.Bridwell: *"Interference detection and collision avoidance among three dimensional objects"*, 1st American Association for Artificial Intelligence conference, Stanford, August 1980.

[2] J.D.Boissonnat: *"Stable matching between a hand structure and an object silhouette"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-4, no 6, November 1982.

[3] J.W.Boyse: *"Interference detection among solids and surfaces"*, CACM, vol.22, nb.1, January 1979.

[4] J.M.Brady et al.: *"Robot Motion: Planning and Control"*, MIT Press, Cambridge, MA, 1982.

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

[6] R.A.Brooks: *"Planning collision free motions for pick and place operations"*, International Journal of Robotics Research, vol 2, no 4, 1983.

[7] R.C.Brost: *"Planning robot grasping motions in the presence of uncertainty"*, Technical Report CMU-RI-TR-85-12, Carnegie Mellon University, The Robotics Institute, Pittsburgh, July 1985.

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

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

[10] B.R.Donald: *"Motion Planning with Six Degrees of Freedom"*, AI-TR-791. Cambridge, Mass.: Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1984.

[11] B.Dufay: *"Apprentissage par induction enRobotique: Application à la synthèse de programmes de montage"*,Thèse de 3ème cycle, INPG, Grenoble, June 1983.

[12] 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.

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

[14] B.Faverjon: *"Obstacle avoidance using anoctree in the configuration space of a manipulator"*, IEEE International Conference on Robotics and Automation, Atlanta, March 1984.

[15] B.Faverjon, P.Tournassoud: *"A local based method for path planning of manipulators with a high number of degrees of freedom"*, IEEE International Conference on Robotics and Automation, Raleigh, April 1987.

[16] R.S.Fearing: *"Simplified grasping and manipulation with dextrous robot hands"*, AI-Memo-809, Artificial Intelligence Lab., M.I.T., Cambridge, November 1984.

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

[18] L.Gouzenes: *"Strategies for solving collision-free trajectories problems for mobile and manipulator robots"*, International Journal of Robotics Reserch, vol 3, no 4, 1984.

[19] H.Hanafusa, H.Asada: *"Stable prehension by a robot hand with elastic fingers"*, 7th International Symposium on Industrial Robots Tokyo, October 1977.

[20] H.Hanafusa, H.Asada: *"A robot hand with elastic fingers and its application to assembly process"*, IFAC Symposium on Information and Control Problems in Manufacturing Technology, Tokyo, 1977.

[21] A.Ijel: *"Inférence de l'équilibre d'un objet à partir d'une représentation des contacts"*, Convention IA 89, Paris, January 1989.

[22] K.Ikeuchi et al.: *"Determining grasp points using photometric stereo and the PRISM binocular stereo system"*, AI-Memo-772, Artificial Intelligence Lab., M.I.T., Cambridge, August 1984.

[23] M.Joyal, P.Provost: *"Statique"*, Masson & Cie, 1966.

[24] O.Khatib: *"Commande dynamique dans l'espace opérationnel des robots manipulateurs en présence d'obstacles"*,Thèse de Docteur-Ingénieur, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, December 1980.

[25] C.Laugier: *"A program for automatic grasping of objects with a robot arm"*, 11th International Symposium on Industrial Robots, Tokyo, October 1981.

[26] C.Laugier, B.Dufay: *"Geometrical reasoning in automatic grasping and contact analysis"*, PROLAMAT 82, Leningrad, May 1982.

[27] C.Laugier, J.Pertin: *"Automatic grasping: a case study in accessibility analysis"*, Published in "Advanced Software in Robotics", edited by A.Danthine and M.Géradin, North Holland, 1984.

[28] C.Laugier: *"Influence du raisonnement géométrique dans le choix d'une prise d'objet"*, Rapport de Recherche IMAG no. 414, December 1983.

[29] C.Laugier, F.Germain: *"An adaptative collision-free trajectory planner"*, '85 International Conference on Advanced Robotics, Tokyo, September 1985.

[30] C.Laugier, J.Troccaz: *"S.H.A.R.P.: A system for automatic programming of manipulation robots"*, 3rd International Symposium of Robotics Research, Gouvieux, October 1985.

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

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

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

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

[35] T.Lozano-Perez, M.A.Wesley: *"An algorithm for planning collision- free paths among polyhedral obstacles"*, CACM, Vol.22, nb.1, October 1979.

[36] T.Lozano-Perez: *"Automatic planning of manipulator transfer movements"*, IEEE Transactions on System, Man and Cybernetics, SMC-11, no. 10, October 1981.

[37] 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.

[38] T.Lozano-Perez: *"Spatial planning: a configuration space approach"*, IEEE Transactions on Computers, vol. C-32, no. 2, February 1983.

[39] T.Lozano-Perez: *"Motion planning for simple manipulator robots"*, 3rd International Symposium of Robotics Research Gouvieux, October 1985.

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

[41] T.Lozano-Perez et al.: *"Handey: a robot system that recognizes, plans and manipulates"*, IEEE International Conference on Robotics and Automation, Raleigh, April 1987.

[42] D.M.Lyons: *"A simple set of grasps for a dextrous hand"*, IEEE International Conference on Robotics and Automation, St Louis, March 1985.

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

[44] M.T.Mason: *"Manipulator grasping and pushing operations"*, AI-TR-690, Artificial Intelligence Lab., M.I.T., Cambridge, June 1982.

[45] E.Mazer: *"HANDEY: Un modèle de planificateur pour la programmation automatique des robots"*, Thèse d'Etat, INPG, Grenoble, December 1987.

[46] N.J.Nilsson: *"Principles of Artificial Intelligence"*, Tioga, Palo Alto, 1980.

[47] E.G.Powell: *"An efficient collision warning algorithm for robot arms"*, 2nd American Association for Artificial Intelligence Conference, Carnegie-Mellon, August 1982.

[48] C.Reboulet, A.Robert: *"Hybrid control of a manipulator with an active compliant wrist"*, 3rd International Symposium on Robotics Research, Gouvieux, October 1985.

[49] J.T.Schwartz, M.Sharir: *"On the piano movers' problem II: General properties for computing topological properties of real algebraic manifolds"*, Dep. of Computer Science, Courant Institute of Mathematical Sciences, NYU, Report 41, February 1982.

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

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

[52] D.J.Todd: *"A method for grasping randomly oriented objects using touch sensing"*, Artificial Intelligence Lab., Queen Mary College, Univ. of London, June 1981.

[53] J.Troccaz: *"S.M.G.R.: a geometric and relational modeller for robotics"*, International Conference on Advanced Robotics, Tokyo, September 1985.

[54] J.Troccaz: *"Modélisation du raisonnement géométrique pour la programmation des robots"*, Thèse de l'Institut National Polytechnique de Grenoble, April 1986.

[55] J.Troccaz: *"On-line automatic robot programming: a case study in grasping"*, IEEE Conference on Robotics and Automation, Raleigh, April 1987.

[56] S.M.Udupa: *"Collision detection and avoidance in computer controlled manipulators"*, Proceedings IJCAI 77, Cambridge, August 1977.

[57] 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.

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

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

[60] J.D.Wolter, R.A.Volz, A.C.Woo: *"Automatic generation of gripping positions"*, RSD-TR-2-84, Center for Robotics and Integrated Manufacturing, Robot Systems Division, Ann Arbor (Michigan), February 1984.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique