Chapter 4

Motion Synthesis

This Chapter presents the synthesis of the fine-motion plan. The plan is a sequence of compliant and guarded motions which define a non-ambiguous path from an initial configuration to a goal configuration. The mechanism to find a non-ambiguous path is presented, which is based on an algorithm that evaluates the $\mathcal{C}$-arcs of the path. Then, free-space motion commands and contact-space motion commands to traverse the path are synthesized. Finally, task execution and implementation issues are discussed.

4.1 Path synthesis overview

The aim of this thesis is to synthesize a fine-motion plan, i.e. a sequence of compliant and guarded motions, that allows the successful execution of an assembly task despite the uncertainty affecting it. The plan is synthesized from the geometry of the task and takes explicitly into account the modelling and sensing uncertainties that affect the task. It has two phases.

The first phase plans a nominal path in free-space, i.e. without considering uncertainty, that connects an initial configuration with a goal configuration. It uses the motion planning algorithm developed in Chapter 2, which is based on an exact cell decomposition method. This method decomposes the free-space and the contact-space, and represents them as two graphs, $\mathcal{C}_f$-graph and $\mathcal{C}_c$-graph, respectively.

The initial configuration is connected to an initial goal node of $\mathcal{C}_f$-graph which belongs to the cell where the initial configuration lies. The path connecting the initial configuration with the initial node is defined to be inside the cell, thus being a free path. The same is done for the goal configuration. The $\mathcal{C}_f$-graph is searched in order to find the path of minimum cost composed of $\mathcal{C}_f$-arcs connecting the initial node with the goal node. In the absence of uncertainty, the obtained path would allow the execution of the assembly from the initial configuration to the goal configuration. Nevertheless, in the presence of uncertainty, the clearance of some of the $\mathcal{C}_f$-arcs of the solution path may not be enough to traverse it collisions-free, and the execution may fail to reach the goal.

Therefore, the second phase of the plan tries to guarantee that the synthesized plan be non-ambiguous, i.e. that the possible contacts occurring during the traversing of the path do not provoke a failure of the task execution. A recurrent cycle of evaluation/synthesis is done in order to find a path in $\mathcal{C}_{free}$:

1- The path is evaluated by searching the $\mathcal{C}_f$-arcs where the uncertainty may cause the assembly to fail.

2- These $\mathcal{C}_f$-arcs are removed from the $\mathcal{C}_f$-graph and a new path is searched again.

When all the $\mathcal{C}_f$-arcs of a path are non-ambiguous, then the path is returned and the motion commands to traverse it are synthesized. If this is not possible, the path which have less ambiguous $\mathcal{C}_f$-arcs is selected, and a patch path to bypass each ambiguous $\mathcal{C}_f$-arc (or sequence of ambiguous $\mathcal{C}_f$-arcs) is synthesized in contact-space, using the same recurrent evaluation/synthesis cycle. If non-ambiguous patch plans are found then the success of the task is not guaranteed.

4.2 Compliant motion

In order to be able to automatically execute assembly tasks with robots in the presence of uncertainty, fine-motion planners make use of active compliance strategies based on sensorial feedback: when contact exists between the mobile object and the fixed objects, the robot complies with the reaction force, while moving along the projection of the commanded direction into the subspace orthogonal to the reaction force. In this work, the force-compliant control based on the generalized damping model is assumed, which is represented by the following equation:

$$\vec{f}(t) = B(\vec{v}_c(t) - \vec{v}_o(t)) \tag{4.1}$$

where $\vec{f}(t)$ is the actual reaction force exerted by the fixed objects on the mobile object at time t, B is the damping matrix (which is the inverse of the accommodation matrix), $\vec{v}_o(t)$ the actual velocity of the mobile object, and $\vec{v}_c(t)$ the commanded velocity. The damping matrix B is assumed to be diagonal.

4.3 Path synthesis

This Section deals with the synthesis of the solution path. First the algorithms to evaluate the $\mathcal{C}$-arcs of a path are presented, and then the path synthesis algorithm is introduced, which is an iterative procedure based on the evaluation of the $\mathcal{C}$-arcs, in order to guarantee that the goal can be reached.

4.3.1 Path evaluation

As defined in Chapter 3, the observed configuration c_o is compatible with the occurrence of a contact situation C_S if C_S can occur at c_o for some values of the deviations.

Let C_{S_i} be the contact situation corresponding to the set of basic contacts S_i, and let $N(c)$ be the number of all the possible contact situations of the assembly task[1] without taking into account the contact situations where c is involved, provided that c is a nominal contact situation. Then, the following algorithm determines the set $C(c)$ of contact situations that a given configuration c is compatible with.

```
Compatible-Configuration($c$)
    $C(c) = \emptyset$
    FOR $i = 1$ TO $N(c)$
        IF Contact-Identification($c_o$, $C_{S_i}$) = TRUE THEN $C(c) = C(c) \cup C_{S_i}$
    RETURN $C(c)$
END
```

Definition: A configuration c is *distinguishable* if when $c_o = c$, then the current contact situation can be identified by the use of the sensory information of configuration and force. Otherwise c is *non-distinguishable.*

The current contact situation can be identified by the use of the sensory information of configuration if:

- c_o is either not compatible with any contact situation or it is compatible with only one.

When c_o is compatible with a set of contact situations $C_{S_i} \ldots C_{S_p}$, then the sensory information of configuration is not enough to identify the current contact situation. Nevertheless, it may be identified by the use of the sensory information of force if:

- the Generalized Force Domains corresponding to the contact situations $C_{S_i} \ldots C_{S_p}$ do not intersect when the set of contacts involved in a contact situation is not a subset of another:

$$\mathbf{G}'_{S_i} \cap \mathbf{G}'_{S_j} = \emptyset \quad \forall S_i, S_j \; | S_i \not\subset S_j \text{ AND } S_j \not\subset S_i \tag{4.2}$$

If this is the case the observed reaction force will be classified to a only one contact situation, since it will belong either to only one Generalized Force Domain, or to more than one Generalized Force Domain such that $S_i \subset S_j$. In this latter case, the current contact situation is assumed to be the one involving less basic contacts.

[1]Complementary contact situations are also considered.

The algorithm Distinguishable-Configuration($c, C(c)$), described below, determines if a configuration is classified as distinguishable. It uses the algorithm Compatible-Configuration(c) and the algorithm Contact-Identification(c_o, C_S) of Chapter 3.

```
Distinguishable-Configuration($c$, $C(c)$)
    d = cardinality of $C(c)$
    IF $d < 2$ RETURN TRUE
    ELSE
        IF $\mathbf{G}'_{S_i} \cap \mathbf{G}'_{S_j} = \emptyset \quad \forall S_i \in C(c), S_j \in C(c) \mid S_i \not\subset S_j$ AND $S_j \not\subset S_i$ RETURN TRUE
        ELSE RETURN FALSE
END
```

Definition: A configuration c is *compliant* at contact situation C_S and for a given commanded velocity $\vec{v}_t$ if when $c_o = c$:

- C_S can occur for some values of the deviations, and
- if C_S occurs then, when $\vec{v}_t$ is applied, the object complies at C_S moving in a direction that reduces the errors introduced by the deviations.

Otherwise c is *non-compliant*, i.e. the manipulated object either sticks at C_S or complies at it moving towards a direction that do not reduce the errors introduced by the deviations.

As an example Figure 4.1a shows an assembly task with two degrees of freedom of translation. The manipulated object is at a configuration c_o, and a velocity $\vec{v}_t$ is commanded to move the object towards the hole. Due to the presence of uncertainty, c_o may be a contact configuration of contact i or contact j, as shown in Figure 4.1b. Then, c_o is compliant at contact j for the commanded velocity $\vec{v}_t$, since if the object complies at this contact, it moves towards the hole. However, c_o is non-compliant at contact i, since if the object is permitted to comply at this contact, it would move towards a wrong direction and the hole will not be reached.

The algorithm to compute if a given configuration c is compliant at contact situation C_i which only involes the basic contact i, uses the partition of the dual plane (Sections 2.8.3 and 3.6.1). In Section 2.8.3 it was shown that the dual plane associated to a configuration c and a contact i is partitioned by the lines π'_t, π'_f and π'_r representing the planes defined by the contact reference frame.

Definition: The *motion regions* are the regions obtained from the partition of the dual plane, and correspond to directions of commanded velocities that produce similar movements of the manipulated object (i.e. produce the same sense of sliding and rotation about the contact point, or produce sticking at it).

In the presence of uncertainty the motion regions get smaller, since their borders may lie in the uncertainty domains $\mathbf{D}_{\pi'_t}$, $\mathbf{D}_{\pi'_f}$ and $\mathbf{D}_{\pi'_{r0}}$, respectively, which where computed in section 3.6.1.

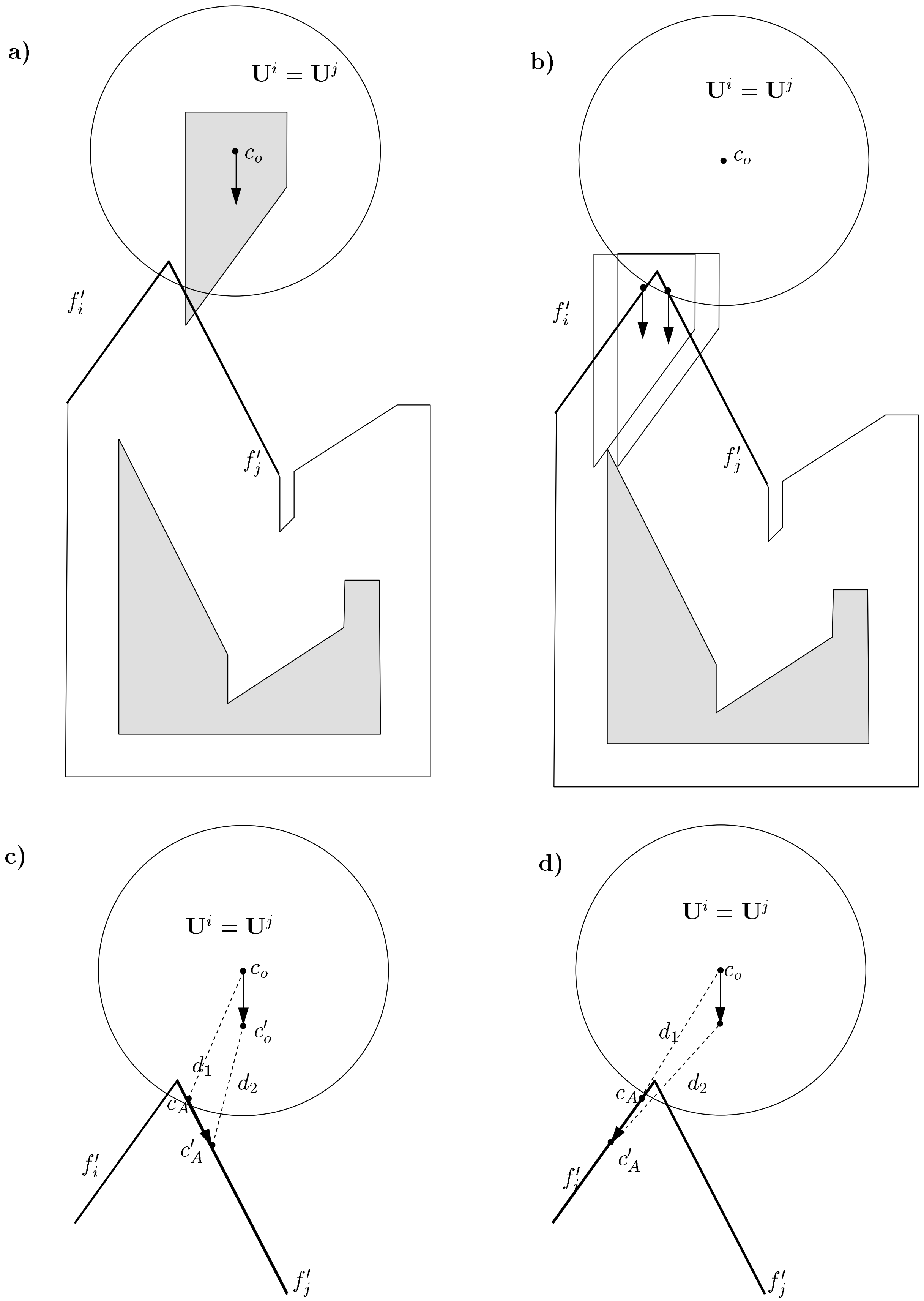


Figure 4.1: *Example of an assembly task at a given observed configuration c_o, which for the commanded velocity $\vec{v}_t$ it is compliant at contact j, and non-compliant at contact i.*

The following two conditions must be satisfied for a configuration c to be compliant at a contact situation C_i and for a given commanded velocity $\vec{v}_t$:

1- The direction $\vec{v}_t$ belongs to a motion region, i.e. it does not belong to any of the domains $\mathbf{D}_{\pi'_t}$, $\mathbf{D}_{\pi'_f}$ or $\mathbf{D}_{\pi'_{r0}}$.

2- The direction $\vec{v}_N$ of the compliant motion that results from applying $\vec{v}_t$ at an arbitrary nominal contact configuration $c_A \in \mathbf{U}^i(\Delta^i_\phi)$ is error-corrective.

When condition 1 is satisfied, then $\vec{v}_t$ is error-corrective for all the possible values of the deviations if condition 2 is also satisfied.

The following algorithm, which is illustrated in Figures 4.1c and 4.1d, computes for a given velocity command $\vec{v}_t$ if a configuration c is compliant at the contact situation C_i.

```
**Compliant-Configuration**($c_o$, $C_i$, $\vec{v}_t$)
    $\epsilon =$ an small positive real value
    $c_A =$ an arbitrary nominal contact configuration $c_A \in \mathbf{U}^i(\Delta^i_\phi)$
    Partiton the dual plane considering uncertainty
    IF $\vec{v}_t$ belongs to $D_{\pi'_f}$, $D_{\pi'_t}$ or $D_{\pi'_{r0}}$ RETURN FALSE
    Compute $d_1$, the distance from $c_o$ to $c_A$
    $c'_o = c_o + \epsilon\vec{v}_t$
    $c'_A = c_A + \epsilon\vec{v}_N$
    Compute $d_2$, the distance from $c'_o$ to $c'_A$
    IF $d_2 < d_1$ RETURN TRUE
    ELSE RETURN FALSE
END
```

When a contact situation C_S involves more than one basic contact, then a configuration c is compliant at it for a given commanded velocity $\vec{v}_t$ if the following two conditions are satisfied:

1- c is compliant at all the contacts involved in C_S.

2- The direction $\vec{v}_t$ belongs to the same motion region for all the contacts involved in C_S.

These conditions are tested in the following algorithm:

```
**Compliant-Configuration**($c$, $C_S$, $\vec{v}_t$)
    FOR ALL $i \in S$
        IF Compliant-Configuration($c$, $C_i$, $\vec{v}_t$) $=$ FALSE THEN RETURN FALSE
    IF $c$ belongs to the same motion region $\forall i \in S$ THEN RETURN TRUE
    ELSE RETURN FALSE
END
```

Definition: A contact configuration is *ambiguous* if it is both non-distinguishable and non-compliant and it is *non-ambiguous* if it is either distinguishable or compliant.

Definition: An non-ambiguous configuration is *guarded* if it is distinguishable and non-compliant.

The following algorithm evaluates if a configuration c is as ambiguous or non-ambiguous. The algorithm tests if c is compliant for all the compatible contact situations. If it is not compliant at a given contact situation, then the test to verify if it is distinguishable is performed. If this test also fails, it means that the configuration is ambiguous and the algorithm returns AMBIGUOUS, otherwise it returns GUARDED. If c is evaluated as compliant for all the possible contact situations then the the algorithm returns COMPLIANT.

```
Configuration-Evaluation($c$, $\vec{v}_t$)
    $C(c)$ = Compatible-Configuration($c$)
    FOR ALL $C_{S_i} \in C(c)$
        IF Compliant-Configuration($c$, $C_{S_i}$, $\vec{v}_t$) = FALSE THEN
            IF Distinguishable-Configuration($c$, $C_{S_i}$) = TRUE THEN RETURN GUARDED
            ELSE RETURN AMBIGUOUS
    RETURN COMPLIANT
END
```

As an example, Figure 4.2 column A, shows three different assembly tasks at a given configuration c_o. It is assumed:

- Uncertainty ϵ_s in the position of the vertices of the static object.
- Uncertainty $(\epsilon_m + \epsilon_{p_r})$ in the position of the vertices of the manipulated object.
- The presence of friction.
- A commanded velocity pointing towards the hole.

Columns B and C in Figure 4.2 show different possible contact configurations for the assemblies of column A. Configuration c_o is evaluated as:

1- GUARDED, since it is:

- *distinguishable* from force information, since the possible force directions do not overlap despite the presence of friction and the deviations in the orientation of the contact edges.
- *non-compliant*, since c is non-compliant at the contact situation of column B because if the object is permitted to comply, then the task would evolve in a wrong direction.

2- COMPLIANT, since for both possible contact situations the task evolves in the right direction if compliance is permitted. In this example, c is *non-distinguishable*, since both contact situations B and C are possible and the possible force directions overlap due to the presence of friction and the deviations in the orientation of the edges.

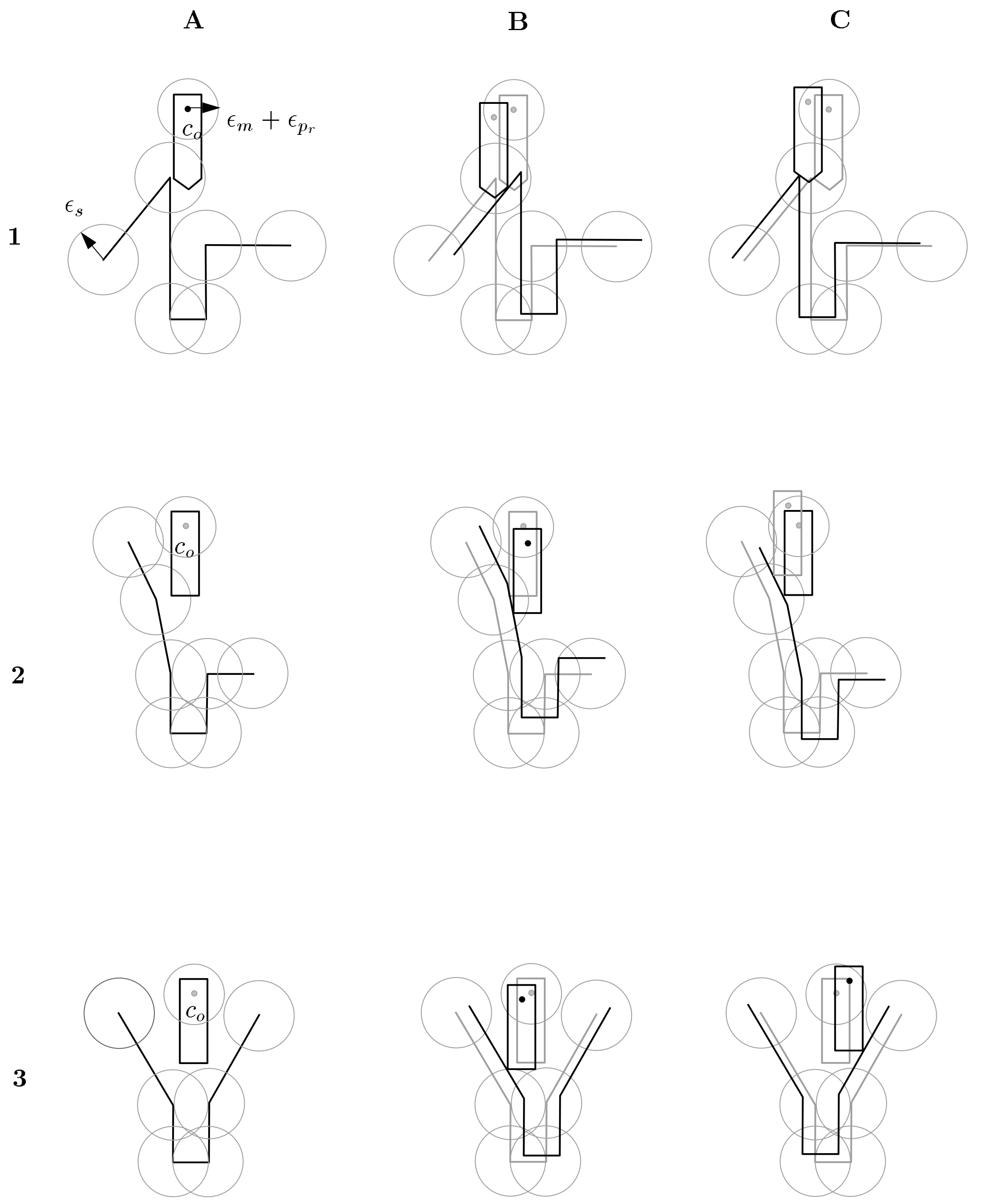


Figure 4.2: *Examples of non-ambiguous configurations.*

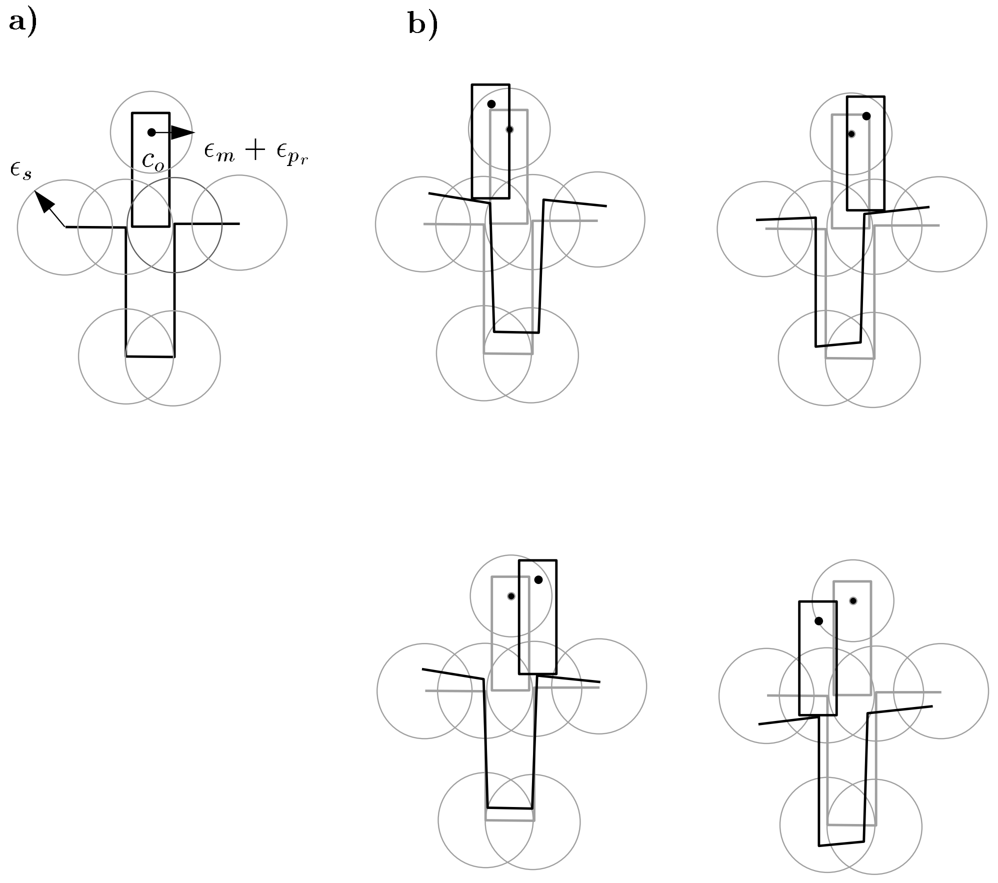


Figure 4.3: *Example of an ambiguous configuration.*

3- COMPLIANT, since for both possible contact situations the task evolves in the right direction if compliance is permitted. In this example, c is *distinguishable* from force information, since the possible force directions do not overlap despite the presence of friction and the deviations in the orientation of the contact edges.

Figure 4.3 shows another assembly task at a given configuration c_o. Figure 4.3a shows the nominal assembly task and the uncertainties considered. Figures 4.3b show four different possible contact situations that can occur when the observed configuration of the manipulated object is c_o. This configuration c_o is evaluated as AMBIGUOUS, since it is:

- *non-distinguishable*, since the possible force directions at the four possible contact situations overlap due to the presence of friction and the deviations in the orientation of the contact edges.
- *non-compliant*, since the manipulated object sticks at the possible contact situations.

Definition: A $\mathcal{C}$-arc of the path is *ambiguous* if it contains an *ambiguous* configuration, and it is *non-ambiguous* otherwise.

Definition: An non-ambiguous $\mathcal{C}$-arc of the path is *compliant* if all of its configurations are compliant, and it is *guarded* otherwise.

A compliant $\mathcal{C}$-arc must be traversed by commanding a compliant motion, i.e. a motion which follows the nominal trajectory of the $\mathcal{C}$-arc while it complies at the possible deviations in the task geometry. A guarded $\mathcal{C}$-arc must be traversed by commanding a guarded motion, i.e. a motion which follows the nominal trajectory of the $\mathcal{C}$-arc until the motion is stopped by the occurrence of a new contact situation identified by the on-line monitoring of the sensory information.

The following algorithm Arc-Evaluation($\mathcal{A}$) evaluates a $\mathcal{C}$-arc $\mathcal{A}$ of a path as either ambiguous, compliant or guarded. A finite set $F_{\mathcal{A}}$ of configurations of the $\mathcal{C}$-arc is considered, such that the distance between the configurations of the set is less than or equal to the positioning uncertainty $(\epsilon_D + \epsilon_I)$. It uses the algorithm Configuration-Evaluation(c) to verify if any of the configurations of $F_{\mathcal{A}}$ are ambiguous. If this is the case the $\mathcal{C}$-arc is evaluated as ambiguous. Otherwise, the algorithm returns COMPLIANT when all of the configurations evaluated are compliant, or GUARDED if any of them is evaluated as guarded.

```
Arc-Evaluation($\mathcal{A}$)
    $g = 0$
    FOR ALL $c \in F_{\mathcal{A}}$
        $r$ = Configuration-Evaluation($c$)
        IF $r$ = AMBIGUOUS THEN RETURN AMBIGUOUS
        ELSE IF $r$ = GUARDED THEN $g = 1$
    IF $g = 1$ THEN RETURN GUARDED
    ELSE RETURN COMPLIANT
END
```

4.3.2 Path synthesis algorithm

The algorithm Path-Synthesis(c_i, c_g) to synthesize the fine-motion plan from an initial configuration c_i to a goal configuration c_g uses the following algorithms:

- Find-Path($\mathcal{C}$-graph,n_i, n_g):
  Finds a non-ambiguous path in $\mathcal{C}$-graph (either $\mathcal{C}_f$-graph or $\mathcal{C}_c$-graph) from an initial node n_i to a goal node n_g, and if it does not exist it finds the path with less ambiguous $\mathcal{C}$-arcs.
- Find-Patches(P, N):
  Analyses the N $\mathcal{C}$-arcs of the path P, removes the ambiguous ones and substitutes them by a patch path in $\mathcal{C}_{contact}$. It returns the patched path and a flag indicating if the patches found are non-ambiguous, i.e. if the reachability of their goal is guaranteed.
- Patch-Path-Synthesis(c_{pi}, c_{pg}):
  Finds a path in $\mathcal{C}_{contact}$ between two contact configurations. It returns the patch path and a flag indicating if it is a non-ambiguous path.

The algorithm Path-Synthesis(c_i, c_g) finds a path between two configurations c_i and c_g in the following way:

(1) Find the paths from c_i and c_g to a $\mathcal{C}_f$-node of the $\mathcal{C}_f$-graph:

  (1.1) Choose $n_i \in \mathcal{C}_f$-graph as the $\mathcal{C}_f$-node that belongs to the same $\mathcal{C}$-prism as c_i and which is closest to c_i. Choose $n_g \in \mathcal{C}_f$-graph in a similar way.

  (1.2) Find a path p_{ini} from c_i to n_i, and the path p_{goal} from c_g to n_g. These paths are contained in a $\mathcal{C}$-prism and, therefore, are described as any $\mathcal{C}_f$-arc connecting two $\mathcal{C}_f$-nodes.

(2) Call the algorithm Find-Path($\mathcal{C}_f$-graph,n_i, n_g) in order to find the $\mathcal{C}$-path p between n_i and n_g.

(3) Add p_{ini} and p_{goal} to p.

(4) If the solution path is ambiguous call the algorithm Find-Patches(P, N) to bypass the ambiguous $\mathcal{C}_f$-arcs.

The algorithm returns the solution path P and a flag with the following value:

FREE: If it is a non-ambiguous path in $\mathcal{C}_{free}$.

GUARANTEED: If it is a non-ambiguous path in $\mathcal{C}_{free}$ with non-ambiguous path patches in $\mathcal{C}_{contact}$.

NON-GUARANTEED: If it is a non-ambiguous path in $\mathcal{C}_{free}$ with some ambiguous path patches in $\mathcal{C}_{contact}$.

```
**Path-Synthesis**($c_i, c_g$)
    $\mathcal{C}$= $\mathcal{C}$-space-Algorithm()
    Find $n_g$, the node of $\mathcal{C}_f$-graph closest to $c_i$
    Find $n_g$, the node of $\mathcal{C}_f$-graph closest to $c_g$
    $p_{ini}$ = path from $c_i$ to $c_{fi}$
    $p_{goal}$ = path from $c_{fg}$ to $c_g$
    $(p, N, p_{type})$ = Find-Path($n_i, n_g$,$\mathcal{C}_f$-graph)
    IF $p_{type}$ = NON-GUARANTEED THEN $(p, p_{type})$ = Find-Patches($p, N$)
    Add $p_{ini}$ and $p_{goal}$ to $p$
    RETURN ($p$, $p_{type}$)
END
```

The following algorithm Find-Path($\mathcal{C}$-graph,n_i, n_g) uses the algorithm Search-Graph($\mathcal{C}$-graph, n_{fi}, n_{fg}) which finds the path p_k with minimum cost between two nodes of a graph using the Dijkstra algorithm of Appendix A. If no path exists then the algorithm returns NON-FEASIBLE. Otherwise, it evaluates all the $\mathcal{C}$-arcs $\mathcal{A}_i^k$ of the solution path and each evaluation is recorded in a variable e_i^k:

- If all of them are non-ambiguous then the algorithm returns:

  1- The solution path p_k.

  2- The number N_k of $\mathcal{C}$-arcs of the solution path.

  3- A flag (FREE) indicating that the path can be traversed in $\mathcal{C}_{free}$, the goal being reachable despite the uncertainty affecting the task.

- Otherwise, the ambiguous $\mathcal{C}$-arcs are removed from the $\mathcal{C}$-graph and a new solution is searched. This procedure is iteratively done and, for each iteration k, the number of ambiguous $\mathcal{C}$-arcs of each path is recorded in a variable a_k. If no path is found with no ambiguous $\mathcal{C}$-arcs, then the algorithm returns:

  1- The path with less ambiguous $\mathcal{C}$-arcs.

  2- The number N_k of $\mathcal{C}$-arcs of the solution path.

  3- A flag (NON-GUARANTEED) indicating that the path has some ambiguous $\mathcal{C}$-arcs.

```
Find-Path($\mathcal{C}$-graph,$n_i, n_g$)
    $k = 1$
    $p_k$ = Search-Graph($\mathcal{C}$-graph, $n_{fi}$, $n_{fg}$)
    $N_k$ = number of $\mathcal{C}$-arcs of $p_k$
    IF $N_k = 0$ RETURN NON-FEASIBLE
    ELSE
        DO
            $a_k = 0$
            FOR $i = 1$ TO $N_k$:
                $e_i^k$ = Arc-Evaluation($\mathcal{A}_i^k$)
                IF $e_i^k$ = AMBIGUOUS THEN
                    $a_k = a_k + 1$
                    Remove $\mathcal{A}_i^k$ from $\mathcal{C}_f$-graph
            IF $a_k = 0$ RETURN ($p_k$, $N_k$, FREE)
            $k = k + 1$
            $p_k$ = Search-Graph($\mathcal{C}$-graph, $n_{fi}$, $n_{fg}$)
        WHILE $p_k \neq \emptyset$
    $K = i \mid a_i = \min\{a_j\} \; \forall j \in \{1..k\}$
    RETURN ($p_K$, $N_K$, NON-GUARANTEED)
END
```

The following algorithm Find-Patches(P, N) analyses the N $\mathcal{C}$-arcs of the solution path P, and finds the sets of ambiguous $\mathcal{C}$-arcs that are contiguous. For each set it calls the algorithm Patch-Path-Synthesis(c_{pi}, c_{pg}) to find a patch path to bypass the ambiguous $\mathcal{C}$-arcs. It returns:

- The solution path P already patched.
- A flag (GUARANTEED or NON-GUARANTEED) indicating if all the $\mathcal{C}$-arcs of the patches are non-ambiguous or not.

```
Find-Patches($P, N$)
    $g$ =TRUE
    FOR $i = 1$ TO $N$
        IF $e_i$ = NON-AMBIGUOUS THEN $i = i + 1$
        ELSE
            $z = i$
            WHILE $e_{z+1}$ = AMBIGUOUS DO $z = z + 1$
            $c_{pi}$ = initial configuration of $\mathcal{A}_i$
            $c_{pg}$ = goal configuration of $\mathcal{A}_z$
            $(p, p_{type})$ = Patch-Path-Synthesis($c_{pi}$, $c_{pg}$)
            IF $p_{type}$ = NON-GUARANTEED THEN $g$ = FALSE
            Remove $\mathcal{A}_i \ldots \mathcal{A}_z$ from $p$
            Add $p$ to $P$
            $i = z + 1$
    IF $g$ = TRUE RETURN ($P$, GUARANTEED)
    ELSE RETURN ($P$, NON-GUARANTEED)
END
```

The following algorithm Patch-Path-Synthesis(c_i, c_g) finds a patch path in $\mathcal{C}_{contact}$ between two contact configurations $c_i \in \mathcal{C}_{free}$and $c_g \in \mathcal{C}_{free}$in the following way:

(1) Find the paths from $\mathcal{C}_{free}$ to $\mathcal{C}_{contact}$:

(1.2) Choose $c_{ci} \in \mathcal{C}_{contact}$ as the middle configuration of a $\mathcal{C}$-item that belongs to the $\mathcal{C}$-prism that contains c_i and which is closest to c_i. Choose $c_{cg} \in \mathcal{C}_{contact}$ in a similar way.

(1.2) Find the path $p_{\mathcal{C}_c}$ from c_i to c_{ci}, and the path $p_{\mathcal{C}_f}$ from c_{cg} to c_g. These paths are contained in a $\mathcal{C}$-prism and therefore are described as any $\mathcal{C}_f$-arc connecting two $\mathcal{C}_f$-nodes.

(2) Find the paths from c_{ci} and c_{cg} to a $\mathcal{C}_c$-node of the $\mathcal{C}_c$-graph:

(2.1) Choose $n_i \in \mathcal{C}_c$-graph as the $\mathcal{C}_c$-node that belongs to the same $\mathcal{C}$-item as c_{ci} and which is closest to c_{ci}. Choose $n_g \in \mathcal{C}_c$-graph in a similar way.

(2.2) Find a path p_{ini} from c_{ci} to n_i, and the path p_{goal} from c_{cg} to n_g. These paths are contained in a $\mathcal{C}$-item and, therefore, are described as any $\mathcal{C}_c$-arc connecting two $\mathcal{C}_c$-nodes.

(3) Call the algorithm Find-Path($\mathcal{C}_c$-graph,n_i, n_g) in order to find the path p between n_i and n_g.

(4) Add $p_{\mathcal{C}_c}$, $p_{\mathcal{C}_f}$, p_{ini} and p_{goal} to p.

The algorithm returns:

1- The patch path p.

2- A flag indicating if it is a non-ambiguous path or not, i.e. weather it is guaranteed that the goal will be reached or not.

```
Patch-Path-Synthesis($c_i$, $c_g$)
    Find $c_{ci} \in \mathcal{C}_{contact}$, the middle configuration of a $\mathcal{C}$-item closest to $c_i$
    Find $c_{cg} \in \mathcal{C}_{contact}$, the middle configuration of a $\mathcal{C}$-item closest to $c_g$
    $p_{\mathcal{C}_c}$ = path from $c_i$ to $c_{ci}$
    $p_{\mathcal{C}_f}$ = path from $c_{cg}$ to $c_g$
    Find $n_i$, the node of $\mathcal{C}_c$-graph closest to $c_{ci}$
    Find $n_g$, the node of $\mathcal{C}_c$-graph closest to $c_{cg}$
    $p_{ini}$ = path from $c_{ci}$ to $n_i$
    $p_{goal}$ = path from $n_g$ to $c_{cg}$
    $(p, N, p_{type})$ = Find-Path($\mathcal{C}_c$-graph,$n_i, n_g$)
    Add $p_{\mathcal{C}_c}$, $p_{\mathcal{C}_f}$, $p_{ini}$ and $p_{goal}$ to $p$
    IF $p_{type}$ = NON-GUARANTEED THEN RETURN ($p$,NON-GUARANTEED)
    ELSE RETURN ($p$,GUARANTEED)
END
```

4.4 Motion commands

The $\mathcal{C}$-arcs of the solution path will be evaluated either as compliant or guarded, regarding the behavior of the motion when a new contact occurs.

- A compliant $\mathcal{C}$-arc can be traversed by following the nominal configurations of the $\mathcal{C}$-arc, and by complying at any new contact produced by the deviations due to the uncertainty that affects the assembly task.

- A guarded $\mathcal{C}$-arc can be traversed by following the nominal configurations of the $\mathcal{C}$-arc, and terminates when a new contact situation occurs. Then, the new contact situation is identified and:
  - If the contact configuration is compliant at the identified contact situation, then the motion resumes along the nominal $\mathcal{C}$-arc as a compliant motion.
  - If the contact configuration is non-compliant at the identified contact situation, then the motion along the $\mathcal{C}$-arc is terminated, and a new motion command must be issued. If the $\mathcal{C}$-arc is considered to have reached its goal, then the $\mathcal{C}$-arc to be followed is the next $\mathcal{C}$-arc in the solution path. Otherwise, an error patch plan must be followed in order to recover from the current contact situation.

Therefore, motion commands will have the following components:

1- A component to follow the nominal $\mathcal{C}$-arcs of the solution path (either in $\mathcal{C}_{free}$ or $\mathcal{C}_{contact}$). This component depends on the nominal geometry, and in the absence of uncertainty it would permit the correct performance of the assembly task. This component is computed from the nominal geometry and from the current measured configuration.

2- An accommodation component necessary to guarantee the contact maintenance if the trajectory to be followed is a $\mathcal{C}_c$-arc. If the $\mathcal{C}$-arc is compliant, this component is devoted to comply to any new contact, by modifying the trajectory to be followed in the direction perpendicular to the $\mathcal{C}$-arc. This component depends on the force measurements and is computed by a force control loop used to maintain a bounded reaction force.

4.4.1 Component to follow a $\mathcal{C}$-arc

The component of the commanded motion to follow a $\mathcal{C}$-arc is obtained from the nominal geometry of the task and from the current observed configuration.

Motions in $\mathcal{C}_f$

The expression of a $\mathcal{C}_f$-arc between two configurations $n_i = (x_i, y_i, \phi_i)$ and $n_g = (x_g, y_g, \phi_g)$ was determined in Chapter 2:

> Let e_1, e_2 and e_3 be the three $\mathcal{C}$-edges of the $\mathcal{C}$-prism where n_i and n_g belong to. Let $e_1(\phi) = (x_1(\phi), y_1(\phi), \phi)$, $e_2(\phi) = (x_2(\phi), y_2(\phi), \phi)$ and $e_3(\phi) = (x_3(\phi), y_3(\phi), \phi)$ be the three configurations of the $\mathcal{C}$-edges for orientation ϕ. Then, the $\mathcal{C}_f$-arc is expressed as:

$$\begin{aligned} x(\phi) &= x_1(\phi) + \alpha(\phi)[x_2(\phi) - x_1(\phi)] + \beta(\phi)[x_3(\phi) - x_1(\phi)] \\ y(\phi) &= y_1(\phi) + \alpha(\phi)[y_2(\phi) - y_1(\phi)] + \beta(\phi)[y_3(\phi) - y_1(\phi)] \\ q(\phi) &= \rho\phi \end{aligned} \tag{4.3}$$

with:

$$\begin{aligned} \alpha(\phi) &= \alpha_i + (\alpha_g - \alpha_i)\frac{\phi - \phi_i}{\phi_g - \phi_i} \\ \beta(\phi) &= \beta_i + (\beta_g - \beta_i)\frac{\phi - \phi_i}{\phi_g - \phi_i} \end{aligned} \tag{4.4}$$

being α_i, α_g, β_i and β_g determined by equation (4.3) for the configurations n_i and n_g.

Therefore, to follow a trajectory between two configurations n_i and n_g, equation (4.3) is used by iteratively subsituting n_i by the current measured configuration c_o and moving the manipulated object along the tangent direction to the $\mathcal{C}_f$-arc.

Motions in $\mathcal{C}_c$

Let $n_i = (x_i, y_i, \phi_i)$ and $n_g = (x_g, y_g, \phi_g)$ be two contact configurations corresponding to the same contact situation involving one basic contact. The expression of a $\mathcal{C}_c$-arc between n_i and n_g was also determined in Chapter 2:

Let e_1, e_2 be the two $\mathcal{C}$-edges of the $\mathcal{C}$-item where n_i and n_g belong to. Let $e_1(\phi) = (x_1(\phi), y_1(\phi), \phi)$ and $e_2(\phi) = (x_2(\phi), y_2(\phi), \phi)$ be the two configurations of the $\mathcal{C}$-edges for orientation ϕ. Then, the $\mathcal{C}_c$-arc is expressed as:

$$\begin{aligned} x(\phi) &= x_1(\phi) + \alpha(\phi)[x_2(\phi) - x_1(\phi)] \\ y(\phi) &= y_1(\phi) + \alpha(\phi)[y_2(\phi) - y_1(\phi)] \\ q(\phi) &= \rho\phi \end{aligned} \tag{4.5}$$

with:

$$\alpha(\phi) = \alpha_i + (\alpha_g - \alpha_i)\frac{\phi - \phi_i}{\phi_g - \phi_i} \tag{4.6}$$

being α_o, α_g determined by equation (4.5) for the configurations n_i and n_g.

Therefore, to follow a trajectory between two contact configurations n_i and n_g while maintaining the contact equation (4.5) is used by iteratively subsituting n_i by the current measured configuration c_o and moving the manipulated object along the tangent direction to the $\mathcal{C}_c$-arc.

If n_i and n_g correspond to a contact situation involving two basic contacts, then the expression of the $\mathcal{C}_c$-arc is that of the $\mathcal{C}$-edge that connects the two contact configurations.

4.4.2 Accommodation components

During a compliant or a guarded motion, if the trajectory to be followed is a $\mathcal{C}_c$-arc, then an accommodation component $\vec{v}_f$ is added in order to guarantee that the contact situation is maintained, producing a bounded and constant reaction force.

One basic contact maintenance

Given a contact situation involving one basic contact, then if the commanded velocity points towards the $\mathcal{C}$-face, a reaction force arises and the mobile object moves along an instantaneous direction of motion over the tangent plane. If the object motion has a positive component along the direction $\vec{t}_p$, then the reaction force will have the direction $\vec{e}^{\,-}$, or $\vec{e}^{\,+}$ otherwise, where $\vec{e}^{\,-}$ and $\vec{e}^{\,+}$ are the directions of the generalized friction cone edges defined in (2.57). Then, the proposed $\vec{v}_f$ is in the opposite direction:

$$\vec{v}_f = \begin{cases} -v_f\vec{e}^{\,-} & if \quad \vec{v}_t \cdot \vec{t}_p > 0 \\ -v_f\vec{e}^{\,+} & \text{otherwise} \end{cases}$$

The module v_f is chosen in accordance to the desired reaction force, f_d:

$$v_f = B^{-1} f_d \tag{4.7}$$

Under some special conditions [38] (e.g. when the reference point of the manipulated object is far away from the contact point and almost above it, being the friction coefficient big enough), one of the edges of the generalized friction cone may dip below the tangent plane, possibly giving rise to motion ambiguities. Therefore, if this is the case, the component $\vec{v}_f$ is set to zero.

Figure 4.4 shows the decomposition of the commanded velocity $\vec{v}_c$ corresponding to a trajectory defined by a $\mathcal{C}_c$-arc:

$\vec{v}_t$: component devoted to the following of the nominal trajectory defined by the $\mathcal{C}_c$-arc.

$\vec{v}_f$: the accommodation component devoted to the contact maintenance.

Two basic contacts maintenance

When the object moves maintaining both contacts due to a commanded velocity pointing towards the tangent planes, a reaction force results whose direction is a linear combination of the involved edges of the corresponding generalized friction cones. Then the proposed accommodation component $\vec{v}_f$ is in the opposite direction:

$$\vec{v}_f = -\frac{\beta_i\vec{e}_i + \beta_j\vec{e}_j}{\mid \beta_i\vec{e}_i + \beta_j\vec{e}_j \mid} v_f \tag{4.8}$$

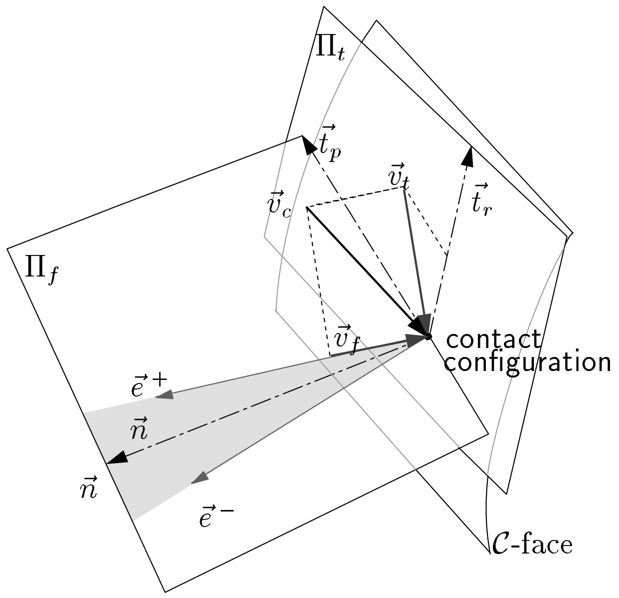


Figure 4.4: *Commanded velocity decomposition.*

where, as before, the module v_f is chosen in accordance to the desired reaction force, and:

$$\vec{e}_k = \begin{cases} \vec{e}_k^{\,-} & if \quad \vec{v}_t \cdot \vec{t}_p^{\,k} > 0 \\ \vec{e}_k^{\,+} & \text{otherwise} \end{cases}$$

with $k = i, j$, and $\beta_i, \beta_j > 0$.

4.5 Task execution

4.5.1 Task execution procedure

Given a planar assembly task described by the geometry of the objects and the values of the modelling and sensing uncertainties, the previous Sections and Chapters have presented:

- The algorithms to synthesize a plan, which is a solution path composed of a sequence of compliant and guarded motions defined either in $\mathcal{C}_{free}$ or in both $\mathcal{C}_{free}$ and $\mathcal{C}_{contact}$.
- The expressions to synthesize the motion commands to follow the trajectory specified by the solution path.

Then, the task execution procedure, shown below, executes each motion of the solution sequence until the proper termination condition terminates it, being the last motion terminated at the assembly goal. In order to improve the motion performance, the sensory information is used to estimate some of the geometric parameters subject to uncertainty, besides being used for the execution and termination of the motions.

```
Task-execution()
    FOR all the arcs of the solution sequence
        DO
            Compute-motion()
            Execute-motion()
            Get-sensory-information()
            Estimate-parameters()
        WHILE Terminate() $\neq$ TRUE
END
```

The functions Compute-motion() and Estimate-parameters() have been analyzed in Sections 4.4 and 3.5, respectively. The functions Execute-motion() and Get-sensory-information() depend on the robot and sensors used for the task execution, and are brefly commented below. The function Terminate() is used to terminate the motion along a $\mathcal{C}$-arc by returning TRUE when one of the following termination conditions holds, or FALSE otherwise.

Termination conditions

Two types of termination conditions can be associated to each $\mathcal{C}$-arc in order to properly conclude the execution of the motion:

TCC: *Termination condition based on configuration information.*
The goal configuration belongs to the Termination Configuration Domain defined as follows. For motions in free-space the Termination Configuration Domain is the volume in $\mathcal{C}$-space determined by the uncertainty in the position and orientation of the robot, i.e. the cylinder of radius ϵ_{p_r} and height ϵ_{ϕ_r}. For motion in contact-space the Termination Configuration Domain is the Contact Configuration Domain, which is determined by the uncertainty in the position and orientation of the robot, and the uncertainties affecting the topological elements involved in the contact. It is used to terminate compliant motions.

TCF: *Termination condition based on force information.*
The measured force belongs to the force generalized domain of a given contact situation. It is used to terminate either guarded motions or compliant motions when the goal node of the $\mathcal{C}$-arc corresponds to a contact situation involving one more contact than those involved in the $\mathcal{C}$-arc.

4.5.2 A case study

A case study has been designed in order to verify the performance of the task execution. The planner proposed in this thesis is not yet fully automated. The nominal solution path in $\mathcal{C}_{free}$ and the possible nominal patch paths in $\mathcal{C}_{contact}$ are automatically computed

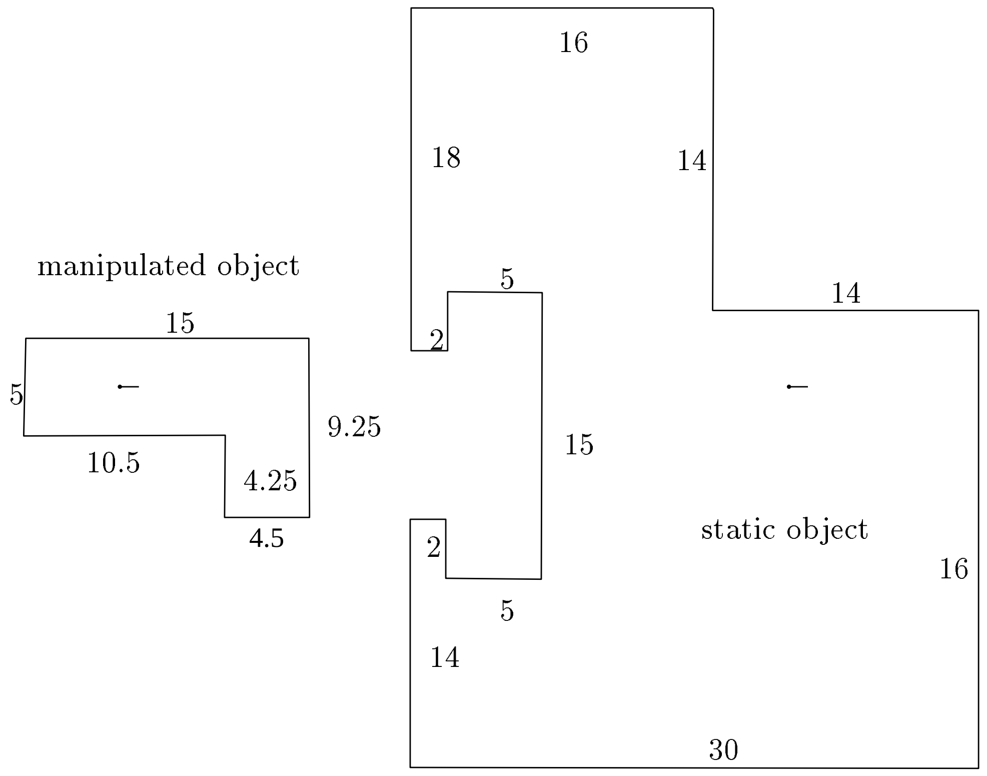


Figure 4.5: *Assembly task (distances in cm).*

from the task geometry, as well as the robot commands needed to follow the $\mathcal{C}$-arcs of the solution path. Nevertheless, the selection of the ambiguous $\mathcal{C}$-arcs to be substituted by paths in $\mathcal{C}_{contact}$ has been done manually.

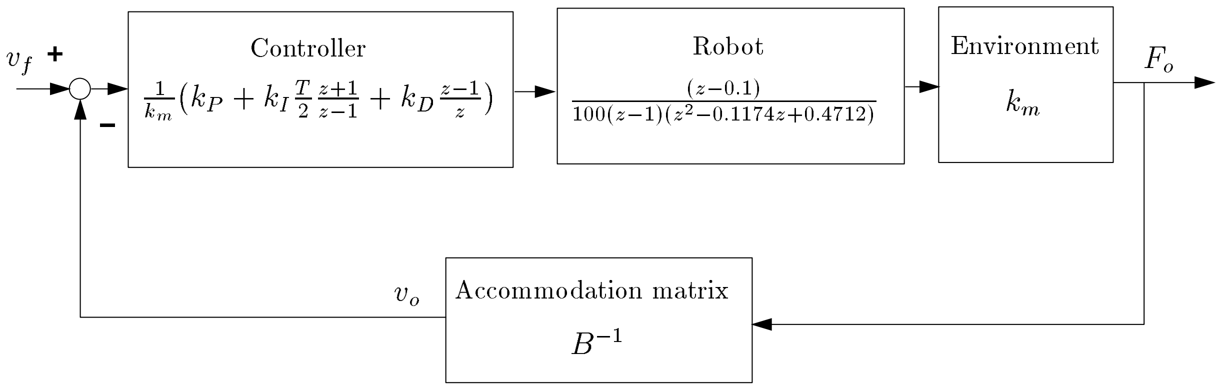
The assembly task

The assembly task shown in Figure 4.5 is used to illustrate the task execution. The uncertainties affecting the task are:

$$\begin{aligned} \epsilon_{p_r} &= 0.1mm \\ \epsilon_{\phi_r} &= 0.1^o \\ \epsilon_s &= 5mm \\ \epsilon_m &= 1mm \end{aligned} \tag{4.9}$$

Setup

The assembly task is to be executed by a Stäubli RX-90 robot equiped with a JR3 force sensor. The robot is controlled with a serial line by an external computer, a Silicon Graphics CRIMSON/Elan, using the real-time path-modification mode of the robot,



v_f
+
−
Controller
$\frac{1}{k_m}(k_P + k_I \frac{T}{2}\frac{z+1}{z-1} + k_D \frac{z-1}{z})$
Robot
$\frac{(z-0.1)}{100(z-1)(z^2-0.1174z+0.4712)}$
Environment
k_m
F_o
v_o
Accommodation matrix
B^{-1}

Figure 4.6: *Control loop.*

known as the *alter mode*. The force control loop used to maintain a bounded reaction force is shown in Figure 4.6, where:

- The sampling period is $T = 32$ ms.
- The robot is modelled as:

$$G_R(z) = \frac{(z-0.1)}{100(z-1)(z^2-0.1174z+0.4712)} \tag{4.10}$$

- The environment has been modelled as a stiffness constant k_m.
- The PID controller has the following values:

$$k_P = 1266.0 \tag{4.11}$$
$$k_I = 920.6 \tag{4.12}$$
$$k_D = 20.6 \tag{4.13}$$

  The PID controller has been synthesized in order to provide a settling time inferior to 1 s, and no overshooting. The gain is divided by an estimation of the environment stiffness k_m in order to adapt the controller to different environments.
- The accommodation matrix is diagonal.

The force sensor is controlled by the Silicon Graphics workstation with a VME bus. It has a sample rate of 8 KHz, and is programmed to use a digital filter with cutoff at 31.25 Hz, providing a new force measurement each 32 ms.

The solution path

$\mathcal{C}_{free}$ and $\mathcal{C}_{contact}$ are partitioned and the corresponding $\mathcal{C}_f$-graph and $\mathcal{C}_c$-graph are obtained, being composed of 693 and 226 nodes, respectively. Figure 4.6 shows two snapshots of the $\mathcal{C}$-space and the triangular mesh used for the partition of $\mathcal{C}_{free}$, corresponding to two different orientations.

The solution path in $\mathcal{C}_{free}$ is composed of the following 13 nodes and is shown in Figure 4.8:

$$22, 23, 154, 155, 173, 171, 170, 169, 164, 163, 132, 123, 125 \tag{4.14}$$

The $\mathcal{C}$-arc connecting nodes 173 and 171 shown in Figure 4.9 is a guarded $\mathcal{C}$-arc, since a contact may occur which cannot be accommodated. Although a patch plan can be issued in order to recover from this situation if this contact occurs, the final solution path has been chosen in order to avoid this from occurring. The final solution path shown in Figure 4.10 has a patch in $\mathcal{C}_{contact}$ and is composed of the following $\mathcal{C}$-nodes:

$$\begin{array}{rl} 22, 23, 154, 155, 173, & \text{of } \mathcal{C}_{free} \\ 82, 83, 84, 109, 106, 104, 121, 116, & \text{of } \mathcal{C}_{contact} \\ 125 & \text{of } \mathcal{C}_{free} \end{array}$$

Task execution

The assembly task has been successfully executed several times with slightly different positions of the objects in order to consider the uncertainty affecting the task. Figure 4.11 shows several snapshots of the task execution.

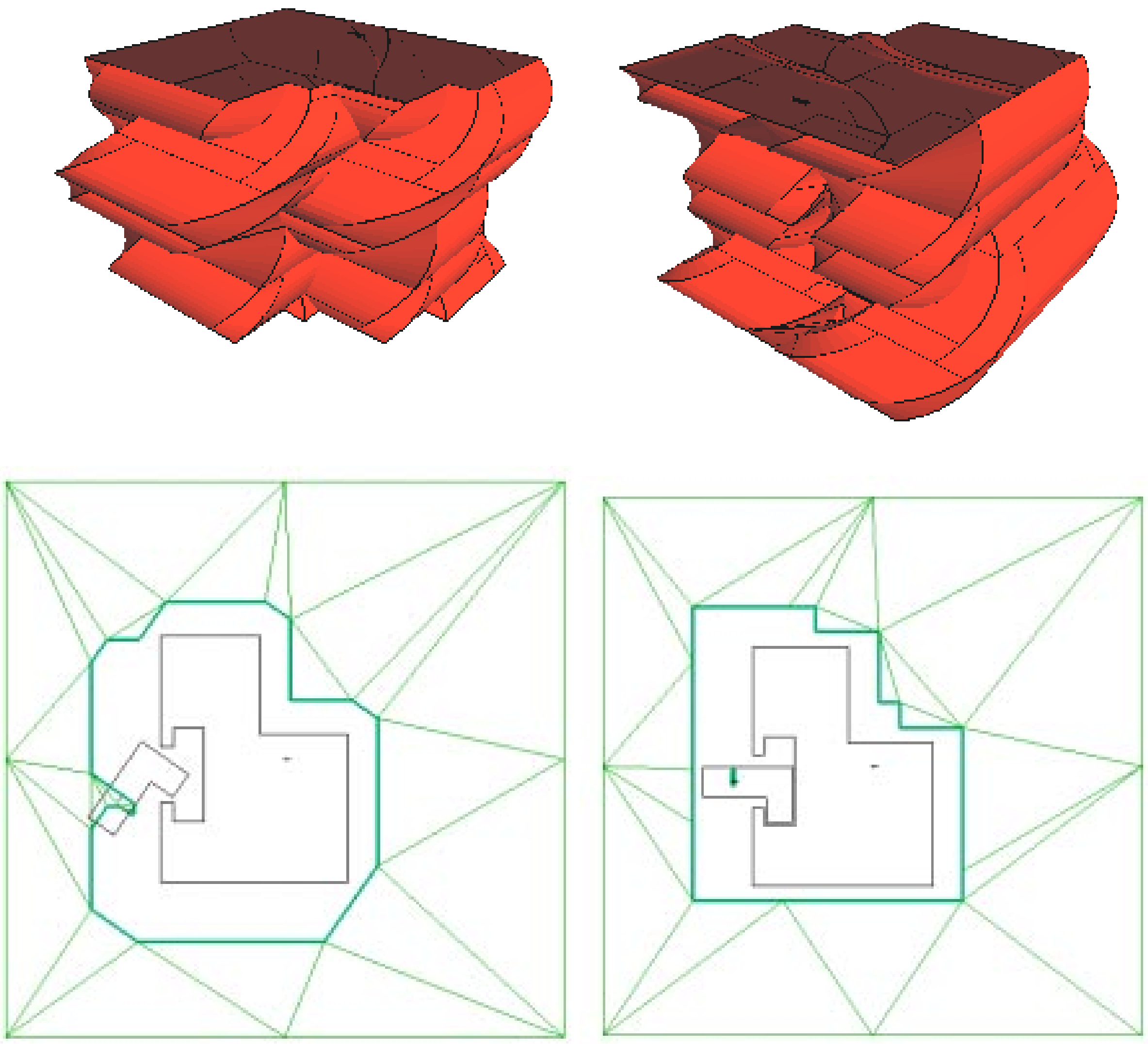


Figure 4.7: *$\mathcal{C}$-space of the assembly task.*

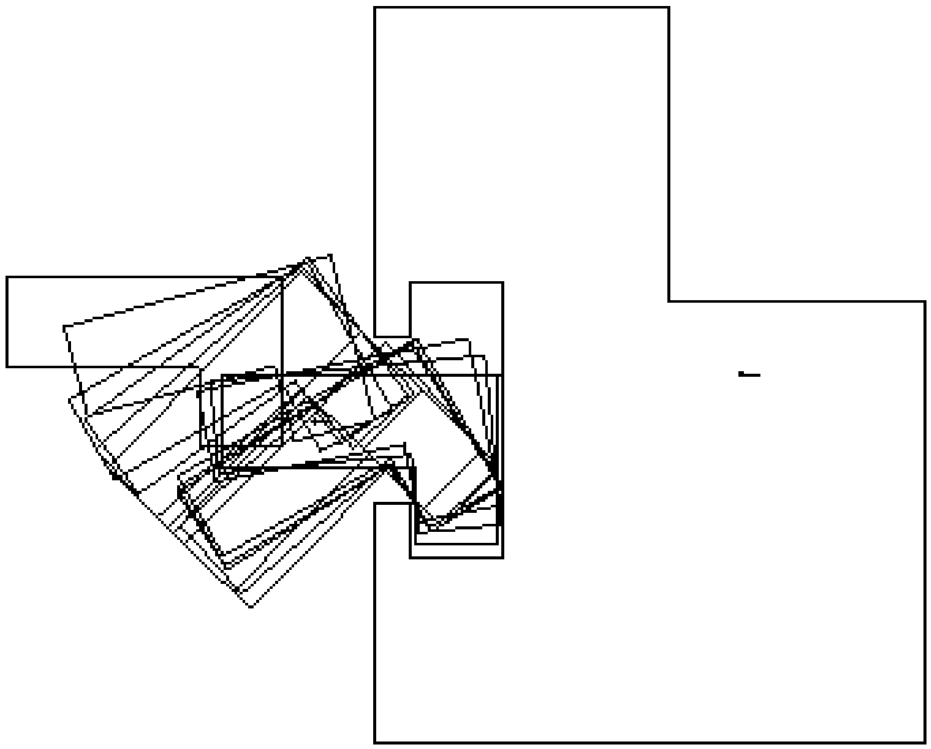


Figure 4.8: *Solution path in $\mathcal{C}_{free}$.*

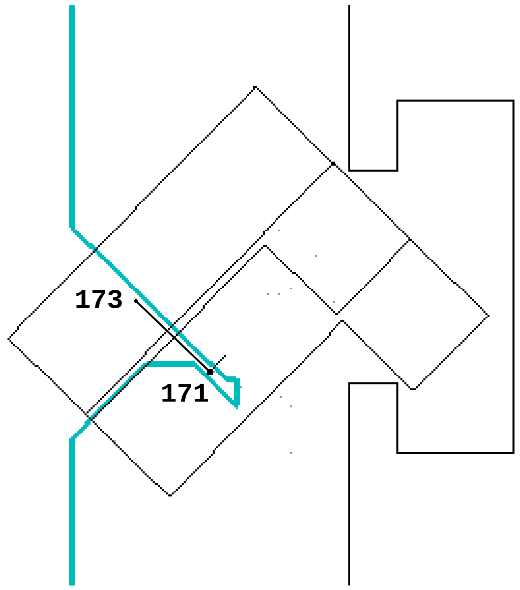


Figure 4.9: *Guarded $\mathcal{C}_c$-arc between $\mathcal{C}_c$-nodes 173 and 171.*

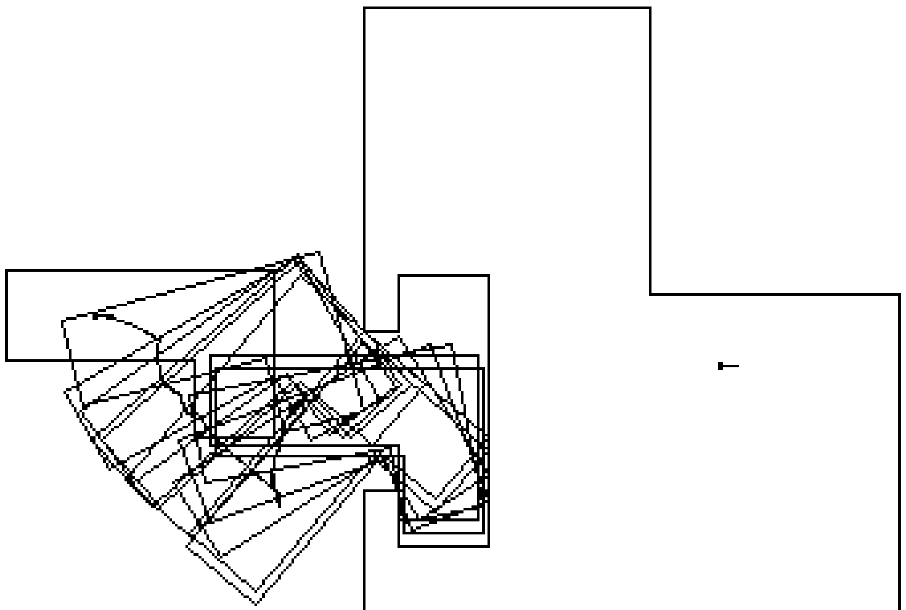


Figure 4.10: *Solution path in $\mathcal{C}_{free}$ and $\mathcal{C}_{contact}$.*

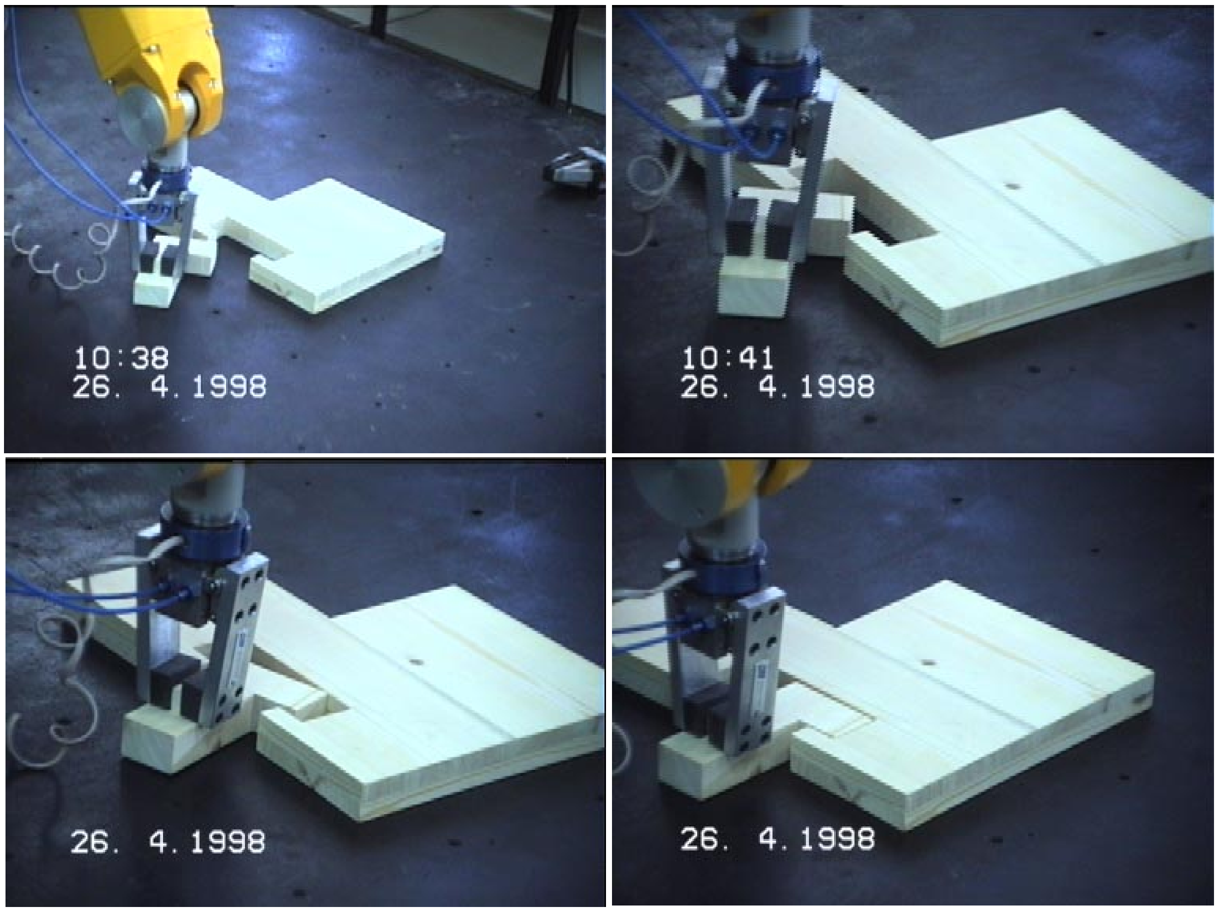


Figure 4.11: *Task execution.*