



UNIVERSITAT POLITÈCNICA  
DE CATALUNYA

## **Description of a robotics-oriented relational positioning methodology**

**Adolfo Rodríguez, Luis Basáñez and Enric Celaya**

*IOC- Divisió de Robòtica*

*IOC-DT-P-2007-11*

*Setembre 2007*

**Institut d'Organització i Control  
de Sistemes Industrials**



# Description of a Robotics-Oriented Relational Positioning Methodology

Adolfo Rodríguez, Luis Basañez, and Enric Celaya

## Abstract

This paper presents a relational positioning methodology for flexibly and intuitively specifying offline programmed robot tasks, as well as for assisting the execution of teleoperated tasks demanding precise movements.

In relational positioning, the movements of an object can be restricted totally or partially by specifying its allowed positions in terms of a set of geometric constraints. These allowed positions are found by means of a 3D sequential geometric constraint solver called PMF – Positioning Mobile with respect to Fixed. PMF exploits the fact that in a set of geometric constraints, the rotational component can often be separated from the translational one and solved independently.

## I. INTRODUCTION

**W**HEN asked to perform a positioning operation, humans usually think of it in terms of satisfying geometric relations. For example, placing a glass on top of a table can be accomplished by making the bottom surface of the glass coincide with the tabletop. Positioning operations can restrict not only totally, but also partially the movement of an object. So in the previous example, the glass can freely translate along directions parallel to the tabletop and can freely rotate about an axis perpendicular to it and still comply with the imposed relation. Furthermore, by using geometric relations, a positioning operation can be defined independently of the initial configurations of the involved objects, so if by some reason the initial positions of the glass and the table change, the operation definition remains meaningful and need not be restated.

In the same way, many robot tasks require the positioning of objects with respect to their surroundings. Although this is an ubiquitous problem in robotics, most existing approaches fail to fulfill all the end user's needs, are not intuitive enough, and rarely support the notions of partial movement restriction and initial configuration independence. For example, in traditional offline programming, configurations are defined in terms of non-intuitive parameters such as homogeneous transformations and joint space coordinates. In gestural programming, the burden is placed on an operator that manually moves the robot end-effector along the desired trajectories, trading a simpler interface for possible workcell downtime and imprecision issues inherent to humans. Simulation-based programming is an attractive alternative since it imposes no workcell downtime, but its usefulness depends on a task representation interface that needs to be both intuitive and adapted to the end user's requirements. Relational positioning can be used to create such an interface.

Relational positioning is a powerful means for placing objects in space, in which the problem is formulated in terms of geometric constraints. A geometric constraint is a relation (distance, angle, tangency, ...) between two or more geometric elements (points, curves, surfaces) that must be satisfied. These elements usually represent boundary or reference features of parent objects. For example, a point may represent the vertex of a cube, and a line may represent the axis of a cylinder. A geometric constraint solver is used to find the positions that each object should have to comply with these constraints. Relational positioning problems which can be solved by positioning one object at a time are called sequential.

Specifying a robot task can be done at multiple levels. Lower levels require defining all the details needed to complete the task (points, trajectories, and the like), while higher levels involve more abstract instructions leaving the details to automated processes. Relational positioning can be used at both levels: at low levels by using geometric constraints to define trajectory points, and at high levels by using the constraints as an intermediate layer between an automatic task planner and the robot controller.

On the other hand, teleoperated task execution relies on operator skills. Some tasks involve movements that require precision such as following a line or maintaining a fixed orientation. Turro et. al. [1] present a system that can generate forces on a haptic device to restrict its movement to curves and surfaces, but these must be explicitly defined by the operator. DeJong et. al. [2] use a combination of a structured light sensor system and an augmented-reality user interface to select curves and surfaces, which are then introduced to a constrained dynamic system simulation [3] for haptic rendering. In such situations, geometric constraints required for the correct execution of the task can be defined, and their effect can be fed back to the operator via visual displays and haptic devices.

There exist many methods for solving geometric constraint problems [4], most of which can be classified as graph-based, logic-based, algebraic, or a combination of these.

*Graph-based* methods construct a (hyper)graph in which the nodes represent geometric elements and the arcs, constraints. Topological features like cyclic dependencies and open chains can be easily detected. Graph analysis identifies simpler and solvable subproblems whose solutions are combined while maintaining compatibility with the initial problem. There exist algorithms with  $O(n^2)$  [5], [6] and  $O(nm)$  [7] time complexity, where  $n$  is the number of geometric elements and  $m$  is the number of constraints.

*Logic-based* methods represent the geometric elements and constraints using a set of axioms and assertions. The solution is obtained following general logic reasoning and constraint rewriting techniques [8], [9].

*Algebraic* methods translate the problem into a set of nonlinear equations, which can be solved using a variety of numeric and symbolic methods. Numeric methods range from the Newton-Raphson method [10] that is simple but does not guarantee convergence nor finding all possible solutions, to more sophisticated ones like Homotopy [11] that guarantee both. They tend to have  $O(n^2)$  -  $O(n^3)$  time complexity. Symbolic methods use elimination techniques such as Gröbner basis to find an exact generic solution to the problem, which can be evaluated with numerical values to obtain particular solutions [12]. These methods are extremely slow, since they have  $O(c^n)$  time complexity.

The application area for geometric constraint solvers is currently dominated by the CAD community, which has widely adopted them as an intuitive framework for parts and assembly design. Most CAD solvers deal with 2D sketching problems [5], [6], but there exist methods that model parts directly in 3D [9], [13]. Among other applications not so widespread figure mechanism design, kinematic modelling, molecular modelling, and robot task specification.

A method is said to be *general* if it admits the formulation of any geometric constraint problem, and *complete* if it is able to solve –or to detect the unsolvability– of all the problems whose formulation it admits. Kramer [7] proposes a geometric constraint solver for open spatial kinematic chains and certain families of closed ones that is neither complete nor general, specially in 3D problems. Porta et. al. [14] describe a complete and general numeric algebraic method based on Cayley-Menger determinants and branch-and-prune techniques. This method has the shortcoming that it is not well suited for relational positioning, because the problem cannot be directly formulated in terms of intuitive geometric constraints. Moreover, the form in which solutions are given needs to be processed before being used in a robot programming or teleoperation application.

This paper presents PMF –Positioning Mobile with respect to Fixed– [15] a sequential geometric constraint solver for the relational positioning of rigid objects in 3D environments by means of

distance and angular constraints between points, lines and planes; PMF handles under-, well-, and overconstrained (redundant and incompatible) problems. The solver has been integrated in a framework for specifying offline programmed robot tasks and assisting the execution of teleoperated ones.

PMF is based on the LMF solver [16], [17], and can be classified as *logic-based*, since it contains a set of constraint rewriting rules that transform an input constraint set into an equivalent set whose solution is known [18].

Two paramount requirements in the design of the solver have been the ability to handle under-constrained problems, since it is often desirable to restrict only partially the motion of a robot and to guide it using the available DOFs; and that solution computation should be fast enough to be included in high-frequency control loops and updated when the topology of the problem changes (e.g., moving obstacles).

Since there is a compromise between *completeness / generality* and *computational efficiency*, it has been opted for a solver that is neither complete nor general, but capable of handling most practical problems of the application domain while fulfilling the above objectives and requirements. Such problems often turn out to be those whose solutions can be pictured qualitatively -but not quantitatively- by the user, so in most cases the user is able to naturally formulate the problem in a way that can be solved by the system.

The PMF solver is described in Sections II – V; sample problems are listed in Section VI; performance and implementation issues are covered in Section VII; and conclusions and future work are finally presented in Section VIII.

## II. SOLVER OVERVIEW

The problem addressed by PMF is that of finding all possible configurations of a 3D mobile object that satisfy a set of geometric constraints defined between the elements of the object and those of its (fixed) environment. The objects are assumed to be rigid and their positions known with respect to a fixed coordinate system.

PMF accepts as input constraints distance ( $d$ ) and angle ( $\angle$ ) relations between points, lines, and planes. Also, the particular cases of coincidence/contained ( $= / \subset$ ), parallelism ( $\parallel$ ), and perpendicularity ( $\perp$ ) relations are explicitly considered for commodity reasons, since they are used very often in practice.

The solver takes advantage of two key facts, which will be described in the following, as well as illustrated with simple examples featuring the objects depicted in Figs. 1a and 1b. Mobile (fixed) elements are identified by the subscript  $m$  ( $f$ ).

Firstly, many geometric constraints restrict both rotational and translational DOFs, but often they can be expressed in terms of pure rotational and pure translational constraints without losing their original meaning<sup>1</sup>. Consider the lines  $\mathcal{L}_m$  and  $\mathcal{L}_f$ . The *line-line* coincidence constraint  $\mathcal{L}_m = \mathcal{L}_f$  restricts all but one rotational and one translational DOF (Fig. 1c), and is equivalent to the *line-line* parallelism constraint  $\mathcal{L}_m \parallel \mathcal{L}_f$ , which is purely rotational; and the *point-line* contained constraint  $\mathbf{P}_m \subset \mathcal{L}_f$ , which is purely translational (where  $\mathbf{P}_m$  is a support point of  $\mathcal{L}_m$ ).

Secondly, the simultaneous satisfaction of two or more pure translational constraints may give rise to an implicit rotational constraint. In fact, it is possible to fully restrict an object –rotations included– using exclusively translational constraints (e.g., a Stewart platform). Conversely, the simultaneous satisfaction of any number of pure rotational constraints never gives rise to implicit translational constraints. Now consider the set of two *point-point* coincidence constraints  $\{\mathbf{P}_m = \mathbf{P}_f, \mathbf{Q}_m = \mathbf{Q}_f\}$ . While the individual satisfaction of either constraint restricts all of the

<sup>1</sup>A constraint is considered purely translational if it can be satisfied regardless of the orientation of the constrained object, and analogously, it is considered purely rotational if it can be satisfied regardless of the object's translation.

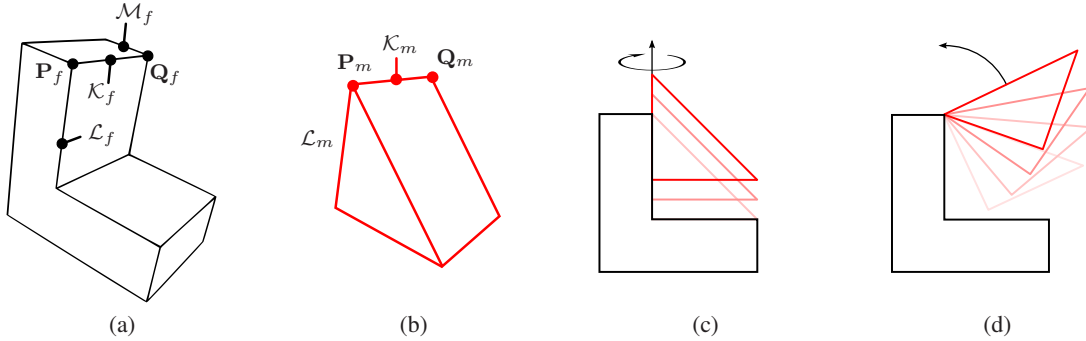


Fig. 1. Sample constraint sets between a fixed (a) and a mobile object (b):  
(c) The constraint  $\mathcal{L}_m = \mathcal{L}_f$  can be decomposed into the pure rotational and translational constraints  $\mathcal{L}_m \parallel \mathcal{L}_f$  and  $\mathbf{P}_m \subset \mathcal{L}_f$ , respectively.  
(d) The constraint set  $\{\mathbf{P}_m = \mathbf{P}_f, \mathbf{Q}_m = \mathbf{Q}_f\}$  implies the rotational constraint  $\mathcal{K}_m \parallel \mathcal{K}_f$ .

mobile object's translational DOFs, the simultaneous satisfaction of both implies the *line-line* parallelism constraint  $\mathcal{K}_m \parallel \mathcal{K}_f$ , which additionally restricts two rotational DOFs, yielding a solution with only one rotational DOF (Fig. 1d). Notice that the two constraints are compatible only if the distance between  $\mathbf{P}_m$  and  $\mathbf{Q}_m$  is the same as the distance between  $\mathbf{P}_f$  and  $\mathbf{Q}_f$ , otherwise the problem will have no solution.

An important observation is that the map between sets of geometric constraints and solutions is *not* injective, so there may exist multiple constraint sets associated to the same solution. For instance, the constraint set  $\{\mathbf{P}_m \subset \mathcal{L}_f, \mathbf{Q}_m \subset \mathcal{M}_f\}$  yields the same solution as the above example (Fig. 1d).

The core idea behind the solver consists in formulating a relational positioning problem in terms of a compact set of pure rotational and translational constraints –which will be referred to as fundamental constraints– and making all rotational constraints explicit. This permits separating the rotational component of the problem so that it can be solved first using only the rotational constraints. Then, the translational component corresponding to each allowed rotation can be easily found using the translational constraints.

This approach may fail for problems that cannot be expressed in terms of fundamental constraints. However, this is not the usual case in relational positioning and, when it appears, it is often easy for the user to provide additional constraints to make the problem solvable.

The solution process starts with the specification of the input constraints by the user, and consists on three main steps: input constraint decomposition, constraint combination, and solution synthesis. A scheme of the process is shown in Fig. 2.

### III. INPUT CONSTRAINT DECOMPOSITION

Input constraints have been selected with the aim of providing an easy way to define the problem. Through input constraint decomposition, an input constraint set  $C_I$  is transformed into an equivalent set of pure rotational and translational fundamental constraints  $C = C_R \cup C_T$  which contains fewer constraint types and is easier to work with.

There are three translational fundamental constraints, which express the distance between a point and another geometric element (point, line, or plane); and one rotational fundamental constraint, which expresses the angle between two vectors:

$$C_T \begin{cases} d(\mathbf{P}_a, \mathbf{P}_b) = p : \textit{point-point} \textit{ distance,} \\ d(\mathbf{P}_a, \mathcal{L}_b) = p : \textit{point-line} \textit{ distance,} \\ d(\mathbf{P}_a, \Pi_b) = 0 : \textit{point-plane} \textit{ coincidence,} \end{cases}$$

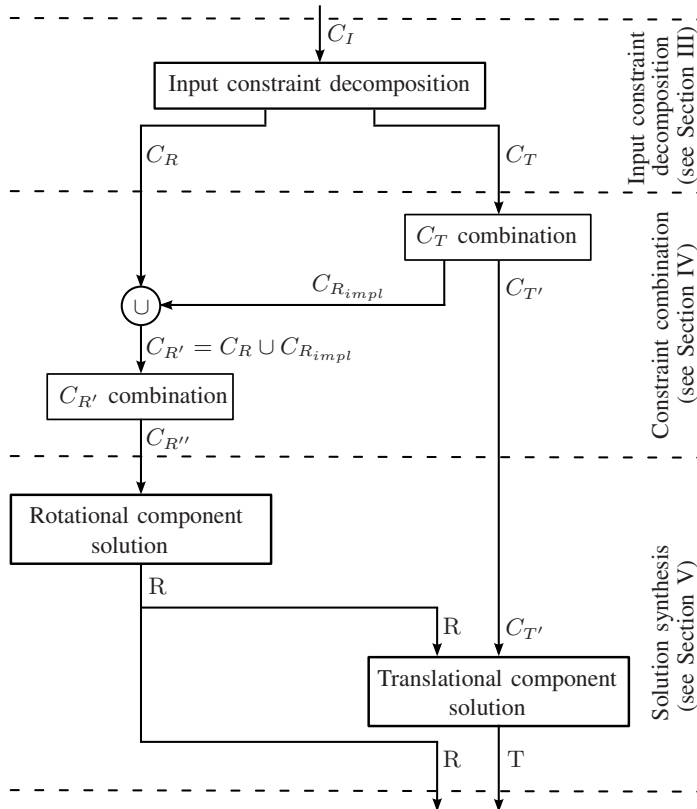


Fig. 2. PMF solver solution process.

$C_I$  represents the input constraint set;  $C_R$  and  $C_T$  represent respectively the rotational and translational fundamental constraint sets ( $C_{R_{impl}}$  contains the implicit rotational constraints); and R, T represent respectively the rotational and translational components of solution. Prime symbols indicate that the elements of a constraint set may have changed.

$$C_R \left\{ \angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = \alpha : \text{vector-vector angle.} \right.$$

Subindices “a” and “b” represent the object to which a geometric element belongs. One object is always fixed, while the other is mobile (e.g.,  $d(\mathbf{P}_m, \mathbf{P}_f) = p$ ,  $d(\mathbf{P}_f, \mathcal{L}_m) = p$ ).

The details of input constraint decomposition are listed in Table I. Input constraints which are already fundamental constraints do not need to be decomposed. The particular cases of coincidence/contained, parallelism, and perpendicularity are not explicitly shown in Table I, but can be obtained in a straightforward manner by setting the appropriate constraint parameter value. For *line-line* distance constraints, only the case that also enforces the lines to be parallel is considered, since the nonparallel case cannot be represented in terms of the adopted fundamental constraints.

#### IV. CONSTRAINT COMBINATION

In constraint combination a set of rules define a constraint rewriting engine that recursively tests constraints in pairs with the purpose of rewriting a set of fundamental constraints in a compact and explicit form with known solution. The tests verify constraint compatibility, so ill-defined cases can be labeled as unsolvable; remove redundancies; check if the pair of constraints can be substituted with a single and equally restrictive constraint (hence the compactness); and identify rotational constraints that are implicitly defined by pairs of translational ones (hence the explicitness). Constraint combination is applied separately to  $C_T$  and  $C_{R'}$ , and in that order, so that implicit rotational constraints are incorporated to  $C_R$  before the combination tests are performed on it (Fig. 2).

TABLE I  
INPUT CONSTRAINT DECOMPOSITION

Input constraint	Fundamental constraints	
	Translational	Rotational
$d(\mathbf{P}_a, \Pi_b) = p$	$d(\mathbf{P}_a, \Upsilon_b) = 0$	-
$d(\mathcal{L}_a, \mathcal{L}_b) = p^*$	$d(\mathbf{R}_a, \mathcal{L}_b) = p$	$\angle(\hat{\mathbf{d}}_{\mathcal{L}_a}, \hat{\mathbf{d}}_{\mathcal{L}_b}) = 0$
$d(\mathcal{L}_a, \Pi_b) = p$	$d(\mathbf{R}_a, \Upsilon_b) = 0$	$\angle(\hat{\mathbf{d}}_{\mathcal{L}_a}, \hat{\mathbf{n}}_{\Pi_b}) = \pi/2$
$d(\Pi_a, \Pi_b) = p$	$d(\mathbf{S}_a, \Upsilon_b) = 0$	$\angle(\hat{\mathbf{n}}_{\Pi_a}, \hat{\mathbf{n}}_{\Pi_b}) = 0$
$\angle(\mathcal{L}_a, \mathcal{L}_b) = \alpha$	-	$\angle(\hat{\mathbf{d}}_{\mathcal{L}_a}, \hat{\mathbf{d}}_{\mathcal{L}_b}) = \alpha$
$\angle(\mathcal{L}_a, \Pi_b) = \alpha$	-	$\angle(\hat{\mathbf{d}}_{\mathcal{L}_a}, \hat{\mathbf{n}}_{\Pi_b}) = \pi/2 - \alpha$
$\angle(\Pi_a, \Pi_b) = \alpha$	-	$\angle(\hat{\mathbf{n}}_{\Pi_a}, \hat{\mathbf{n}}_{\Pi_b}) = \alpha$

with  $d(\Upsilon_a, \Pi_a) = p$      $\mathbf{R}_a \subset \mathcal{L}_a$      $\mathbf{S}_a \subset \Pi_a$

\* Only the case that also enforces  $\mathcal{L}_a \parallel \mathcal{L}_b$  is considered.

The constraint rewriting rules are obtained by applying the following method to each pair of constraint types we are interested in testing:

- Find the compatibility conditions that enable the two constraints to be satisfied simultaneously. They will depend on up to four distance or angle parameters.
- Evaluate the compatibility conditions in its general form, as well as at the limit cases (e.g., parallel elements, equality of a greater-or-equal-than condition) for all possible combinations obtained by making the parameters equal to zero.
- If any of the above configurations can be represented in terms of a single fundamental constraint, create a rule that substitutes the original pair with this single constraint.
- If the constraints being tested are translational and the configuration reveals an implicit rotation, create a rule that explicitly adds this constraint to  $C_R$ .

If two constraints  $c_1$  and  $c_2$  can be substituted with the single and equally restrictive constraint  $c_3$ , this new constraint must also be combined with the remaining constraints in the set, hence the recursivity of this step. Sometimes there may be a set of alternatives or multiple solutions for  $c_3$ . For example, the intersection of two parallel cylindrical subspaces may yield two unconnected lines. When this occurs, the current problem is branched so that there is a different problem instance associated to every solution. These new problem instances are solved independently of each other.

Sometimes it may be impossible to explicitly extract a rotational constraint even if implicitly exists. This occurs when the result of a combination test cannot be expressed in terms of a *vector-vector* angle, or when more than two constraints need to be simultaneously considered to extract it.

Table II lists the compatibility conditions for all combination scenarios handled by the solver, and Fig. 3 depicts the particular example of two *point-plane* coincidence constraints. The details concerning all the constraint combination rules used by PMF are listed in Appendix IV.

## V. SOLUTION SYNTHESIS

Solution synthesis (bottom part of Fig. 2) computes transformations that position the mobile object in a configuration that simultaneously satisfies *all* the imposed geometric constraints. This is a two-step process that takes advantage of the separation of fundamental constraints into pure rotational and pure translational ones.

First, the rotational component of the solution  $\mathbf{R}$  is solved using only the constraints in  $C_{R'}$ , where  $\mathbf{R}$  maps the initial orientation of the mobile object to a subspace of the three-dimensional space of rotations that satisfies all the rotational constraints. Then, from a configuration that already satisfies  $\mathbf{R}$ , the translational component  $\mathbf{T}$  is solved using the constraints in  $C_{T'}$ , where  $\mathbf{T}$  maps

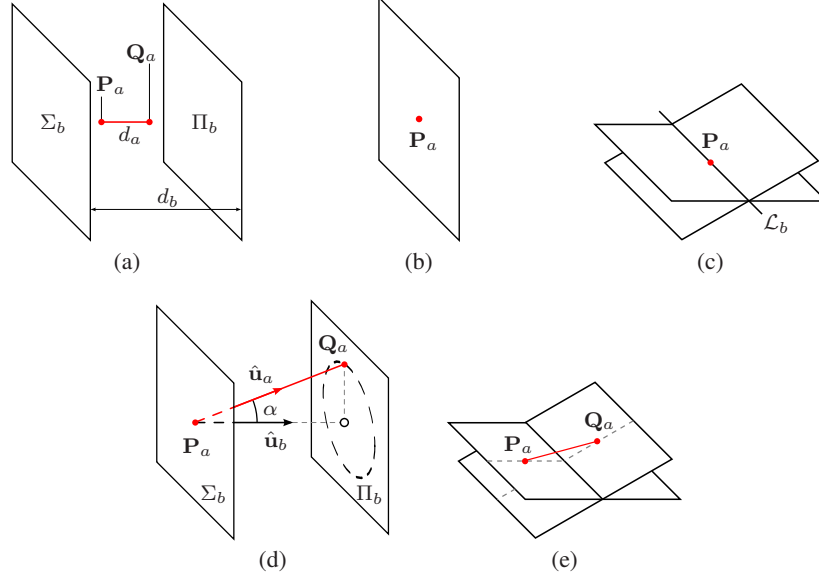


Fig. 3. Combination rules for two *point-plane* coincidence constraints  $d(\mathbf{P}_a, \Sigma_b) = 0$ ,  $d(\mathbf{Q}_a, \Pi_b) = 0$  (compatibility condition  $d_b - d_a \leq 0$ ):

- (a) ( $d_a < d_b$ ): Incompatible constraints. Label problem as unsolvable.
- (b) ( $\Sigma_b \parallel \Pi_b$ ) and ( $d_a = d_b = 0$ ): Redundant constraints – remove one.
- (c) ( $\Sigma_b \not\parallel \Pi_b$ ) and ( $d_a = 0$ ): Substitute both constraints with  $d(\mathbf{P}_a, \mathcal{L}_b) = 0$ .
- (d) ( $\Sigma_b \parallel \Pi_b$ ) and ( $d_a \geq d_b \geq 0$ ): Add the implicit rotation  $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = \alpha$  to  $C_R$ , where  $\alpha = \cos^{-1}(d_b/d_a)$ .
- (e) ( $\Sigma_b \not\parallel \Pi_b$ ) and ( $d_a \geq d_b > 0$ ): No rules are obtained for this case.

TABLE II  
COMPATIBILITY CONDITIONS FOR CONSTRAINT COMBINATION

Constraint pair	Compatibility condition
$d(\mathbf{P}_a, \mathbf{P}_b) = p$ , $d(\mathbf{Q}_a, \mathbf{Q}_b) = q$	$(d_a + d_b \geq  q - p ) \vee ( d_b - d_a  \leq p + q)$
$d(\mathbf{P}_a, \mathbf{P}_b) = p$ , $d(\mathbf{Q}_a, \mathcal{L}_b) = q$	$(d_a + d_b \geq q - p) \vee (d_b - d_a \leq p + q)$
$d(\mathbf{P}_a, \mathcal{K}_b) = p$ , $d(\mathbf{Q}_a, \mathcal{L}_b) = q$	$[(\mathcal{K}_b \parallel \mathcal{L}_b) \wedge (d_a + d_b \geq  q - p )] \vee (d_b - d_a \leq p + q)$
$d(\mathbf{P}_a, \mathcal{K}_b) = p$ , $d(\mathbf{Q}_a, \Pi_b) = 0$	$d_b - d_a \leq p$
$d(\mathbf{P}_a, \mathbf{P}_b) = p$ , $d(\mathbf{Q}_a, \Pi_b) = 0$	$d_b - d_a \leq p$
$d(\mathbf{P}_a, \Sigma_b) = 0$ , $d(\mathbf{Q}_a, \Pi_b) = 0$	$d_b - d_a \leq 0$
$d(\mathbf{P}_a, \mathcal{K}_b) = p$ , $d(\mathbf{Q}_b, \mathcal{L}_a) = q$	-
$d(\mathbf{P}_a, \mathcal{K}_b) = p$ , $d(\mathbf{Q}_b, \Pi_a) = 0$	-
$\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = \alpha$ , $\angle(\hat{\mathbf{v}}_a, \hat{\mathbf{v}}_b) = \beta$	$(\sigma_a + \sigma_b \geq  \alpha - \beta ) \vee ( \sigma_a - \sigma_b  \leq \alpha + \beta)$

where  $d_a$  ( $d_b$ ) represents the distance between the two elements belonging to object “a” (“b”).  
The same applies to  $\sigma_a$  ( $\sigma_b$ ) but with angles instead of distances.

the translation associated to an R-satisfying configuration of the mobile object to a subspace of the three-dimensional space of translations that satisfies all the translational constraints. The dimension of the above subspaces correspond to the number of available DOFs each solution component has. Notice that since T depends on the current value of R, it must be recomputed every time R changes. Algorithm 1 summarizes the process.

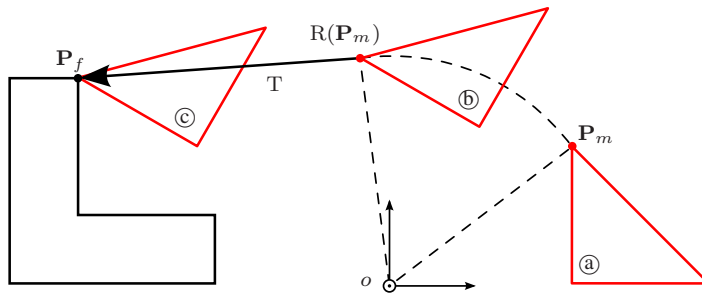


Fig. 4. Solution synthesis process: evaluation of R and T.

- Ⓐ Initial configuration of the mobile object.
- Ⓑ R-satisfying configuration.
- Ⓒ Final configuration. It satisfies both R and T.

---

### Algorithm 1 solution synthesis

---

**Require:** Fundamental constraint set  $C = C_{R'} \cup C_{T'}$

```

if problem is labelled as ill-defined then
  return problem has no solution
end if
if  $C_{R'}$  can be handled (Table III) then
  solve rotational component R (may have multiple solutions)
else
  return problem is unhandled
end if
if  $C_{T'}$  can be handled (Table IV) then
  solve translational component T for the computed rotation R
else
  return problem is unhandled
end if
if R or T have DOFs then
  initialize free parameters
end if
return R, T

```

---

Solutions are represented by a rigid transformation parameterized by as many parameters as available DOFs, and particular instances are built on demand by setting the values of these parameters. A common representation for rigid transformations are 4x4 matrices, where R and T take the form of a 3x3 matrix and a 3x1 vector, respectively. A particular solution  $P$  can be then written as

$$P = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}. \quad (1)$$

Other rigid transformation representations may also be used. For example, the current implementation of the solver uses unit quaternions to represent rotations for numerical stability reasons, but for simplicity the present discussion will adhere to the 4x4 matrix notation.

To exemplify the solution synthesis process, consider the fixed and mobile objects depicted in Figs. 1a and 1b, respectively; and let the mobile object be restricted with a constraint set that yields a solution such as the one depicted in Fig. 1d. The process is illustrated in Fig. 4, where all geometric elements are represented in the fixed coordinate system  $o$ .

TABLE III  
POSSIBLE SETS OF ROTATIONAL CONSTRAINTS

Constraints	DOF	# Solutions
$\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = 0, \angle(\hat{\mathbf{v}}_m, \hat{\mathbf{v}}_f) = 0$	0	1
$\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = \alpha, \angle(\hat{\mathbf{v}}_m, \hat{\mathbf{v}}_f) = \beta, \angle(\hat{\mathbf{w}}_m, \hat{\mathbf{w}}_f) = \gamma$ *	0	up to 8
$\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = 0$	1	1
$\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = \alpha, \angle(\hat{\mathbf{v}}_m, \hat{\mathbf{v}}_f) = \beta$	1	2
$\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = \alpha$	2	1
none	3	1

\* Only certain cases are considered.

Solving R and applying it to the initial configuration of the mobile object ① results in the R-satisfying configuration ②. Then, T is solved for the current value of R (so that points  $\mathbf{P}_m$  and  $\mathbf{P}_f$  coincide), that when applied to ② results in ③, a configuration that satisfies both R and T.

In this case,  $T = \mathbf{P}_f - R(\mathbf{P}_m)$ , and a matrix representation of the solution would have the form

$$P = \begin{bmatrix} R & \mathbf{P}_f - R(\mathbf{P}_m) \\ 0 & 1 \end{bmatrix}. \quad (2)$$

In Section V-B it will be shown that the general form of T is very similar to the one presented above.

The rotational component of the solution is solved before the translational one because rotations are *origin-preserving* transformations, this is, when the orientation of an object changes, all points except the coordinate system origin translate. On the other hand, translations are *orientation-preserving* transformations, so when the position of an object changes, its orientation remains unchanged. If the solution order is inverted so that the mobile object is first translated to satisfy the translational constraints and then rotated to satisfy the rotational ones, it can be observed that in the general case, the final configuration will no longer satisfy the translational constraints.

#### A. Rotational component

Table III lists the possible sets of rotational constraints with their corresponding number of DOFs and solutions. After performing  $C_{R'}$  reduction *any* set of rotational constraints can be matched to one of the these entries, so unmentioned cases have not been included in the list not because they are unhandled, but because they can be rewritten in a form that matches one of the table entries. The rotation R that must be applied to the mobile object in each case will be now discussed in detail. In order to facilitate the reading, solutions to the more complicated cases have been deferred to the appendices. Also, the trigonometric functions  $\sin \alpha$  and  $\cos \alpha$  will be abbreviated as  $s_\alpha$  and  $c_\alpha$ , respectively.

1) *Two parallelism constraints*  $\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = 0, \angle(\hat{\mathbf{v}}_m, \hat{\mathbf{v}}_f) = 0$  : This is a particular case of the rotation associated to two pairs of independent unit vectors,  $R(\hat{\mathbf{u}}_m, \hat{\mathbf{v}}_m \rightarrow \hat{\mathbf{u}}_f, \hat{\mathbf{v}}_f)$ , as explained in Appendix III-A.

2) *Three angle constraints*  $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = \alpha, \angle(\hat{\mathbf{v}}_a, \hat{\mathbf{v}}_b) = \beta, \angle(\hat{\mathbf{w}}_a, \hat{\mathbf{w}}_b) = \gamma$  : A “simple” analytical solution can be found when both the mobile and fixed vectors form right trihedra, cases that fall outside this criteria will not be considered. Appendix III-C details how to compute such rotation, which can have up to eight distinct solutions.

3) *One parallelism constraint*  $\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = 0$  : Let  $\sigma$  be the angle between  $\hat{\mathbf{u}}_m$  and  $\hat{\mathbf{u}}_f$ , and  $\hat{\mathbf{w}} = \hat{\mathbf{u}}_m \times \hat{\mathbf{u}}_f$ ; the rotation  $R(\phi)$  is given by

$$R(\phi) = R(\hat{\mathbf{u}}_f, \phi) R(\hat{\mathbf{w}}, \sigma) \quad (3)$$

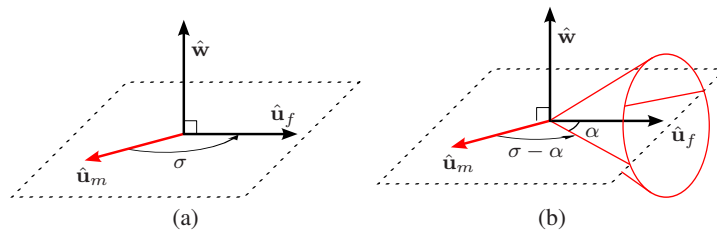


Fig. 5. Enforcement of two particular solutions of the rotational component:  
(a) One *vector-vector* parallelism constraint  $\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = 0$ .  
(b) One *vector-vector* angle constraint  $\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = \alpha$ .

where  $R(\hat{\mathbf{u}}_f, \phi)$  accounts for the free rotation about the direction of  $\hat{\mathbf{u}}_f$  and  $R(\hat{\mathbf{w}}, \sigma)$  is the fixed rotation that places  $\hat{\mathbf{u}}_m$  parallel to  $\hat{\mathbf{u}}_f$  (see Fig. 5a). Appendix III-B describes how to compute rotations defined by an axis-angle pair.

4) *Two angle constraints*  $\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = \alpha$ ,  $\angle(\hat{\mathbf{v}}_m, \hat{\mathbf{v}}_f) = \beta$  : This case gives rise to two solutions, each with one DOF that permits limited rotations along one direction. Its computation is explained in Appendix III-D.

5) *One angle constraint*  $\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = \alpha$  : Let  $\sigma$  be the angle between  $\hat{\mathbf{u}}_m$  and  $\hat{\mathbf{u}}_f$ , and  $\hat{\mathbf{w}} = \hat{\mathbf{u}}_m \times \hat{\mathbf{u}}_f$ ; the rotation  $R(\phi, \theta)$  is given by

$$R(\phi, \theta) = R(\hat{\mathbf{u}}_f, \phi) R(\hat{\mathbf{w}}, \sigma - \alpha) R(\hat{\mathbf{u}}_m, \theta). \quad (4)$$

$R(\hat{\mathbf{u}}_m, \theta)$  and  $R(\hat{\mathbf{u}}_f, \phi)$  account for the free rotations about the directions of  $\hat{\mathbf{u}}_m$  and  $\hat{\mathbf{u}}_f$ , respectively; and  $R(\hat{\mathbf{w}}, \sigma - \alpha)$  is the fixed rotation that places  $\hat{\mathbf{u}}_m$  at an angle of  $\alpha$  with  $\hat{\mathbf{u}}_f$ . Fig. 5b depicts the enforcement of this constraint.

6) *No rotational constraints*: The mobile object is able to freely rotate in any direction. However, the rotation has been parameterized according to the Yaw, Pitch, and Roll convention, in which the rotation axis correspond to those of the fixed coordinate system.

$$R(\phi, \theta, \psi) = R(\hat{\mathbf{k}}, \psi) R(\hat{\mathbf{j}}, \theta) R(\hat{\mathbf{i}}, \phi). \quad (5)$$

## B. Translational component

Table IV lists the handled sets of translational constraints with their associated number of DOFs. Basically, the handled cases correspond to the three translational fundamental constraints –considering separately the particular case when the distance parameter equals zero– and some configurations considered meaningful which cannot be expressed in terms of a single fundamental constraint, but rather two, such as the one shown in Fig. 3e. Note that contrary to the rotational component solution, not all possible combinations of translational constraints can be reduced to a form that matches an entry of Table IV. All constraint sets that are not mentioned in the table but can be rewritten through  $C_T$  reduction in a form that matches one of its entries will have a valid solution, otherwise they will be considered unhandled.

The translation  $\mathbf{T}$  that must be applied to an R-satisfying mobile object has the general form

$$\mathbf{T} = \mathbf{S}_f - R(\mathbf{S}_m) \quad (6)$$

where  $\mathbf{S}_m$  and  $\mathbf{S}_f$  belong to one of the subspaces listed in Table V. Five particular solutions of (6) will be now explained in detail: the first three solutions are associated to single constraints, and the last two correspond to solutions defined by pairs of constraints. In these examples, subindices “a” and “b” have been substituted by “m” or “f” to explicitly identify the mobile and fixed elements.

TABLE IV  
HANDLED SETS OF TRANSLATIONAL CONSTRAINTS

Constraints	DOF	Condition
$d(\mathbf{P}_a, \mathbf{P}_b) = 0$	0	
$d(\mathbf{P}_a, \mathcal{K}_b) = 0, d(\mathbf{Q}_a, \mathcal{L}_b) = 0$	0	$\mathbf{R}_b = \mathcal{K}_b \cap \mathcal{L}_b$
$d(\mathbf{P}_a, \mathcal{K}_b) = 0, d(\mathbf{Q}_a, \Pi_b) = 0$	0	$\mathbf{R}_b = \mathcal{K}_b \cap \Pi_b$
$d(\mathbf{P}_a, \mathcal{L}_b) = 0$	1	
$d(\mathbf{P}_a, \mathcal{K}_b) = p, d(\mathbf{P}_a, \Pi_b) = 0$	1	$\mathbf{R}_b = \mathcal{K}_b \cap \Pi_b$
$d(\mathbf{P}_a, \Sigma_b) = 0, d(\mathbf{Q}_a, \Pi_b) = 0$	1	$\mathcal{L}_b = \Sigma_b \cap \Pi_b$
$d(\mathbf{P}_a, \mathbf{P}_b) = p$	2	
$d(\mathbf{P}_a, \mathcal{L}_b) = p$	2	
$d(\mathbf{P}_a, \Pi_b) = 0$	2	
none	3	

TABLE V  
PARAMETRIC REPRESENTATION OF TRANSLATIONAL SUBSPACES

Subspace	Parametric representation ( $\hat{\mathbf{d}}_1, \hat{\mathbf{d}}_2,$ and $\hat{\mathbf{d}}_3$ are mutually perpendicular)
Point	$\mathbf{S} = \mathbf{P}$
Line	$\mathbf{S}(\lambda) = \mathbf{P} + \lambda \hat{\mathbf{d}}$
Plane	$\mathbf{S}(\lambda_1, \lambda_2) = \mathbf{P} + \lambda_1 \hat{\mathbf{d}}_1 + \lambda_2 \hat{\mathbf{d}}_2$
Ellipse/circle	$\mathbf{S}(\alpha) = \mathbf{P} + a c_\alpha \hat{\mathbf{d}}_1 + b s_\alpha \hat{\mathbf{d}}_2$
Sphere	$\mathbf{S}(\alpha, \beta) = \mathbf{P} + a(c_\alpha s_\beta \hat{\mathbf{d}}_1 + s_\alpha s_\beta \hat{\mathbf{d}}_2 + c_\beta \hat{\mathbf{d}}_3)$
Cylinder	$\mathbf{S}(\lambda, \alpha) = \mathbf{P} + \lambda \hat{\mathbf{d}}_1 + a(c_\alpha \hat{\mathbf{d}}_2 + s_\alpha \hat{\mathbf{d}}_3)$
$\mathbb{R}^3$	$\mathbf{S}(\lambda_1, \lambda_2, \lambda_3) = \mathbf{P} + \lambda_1 \hat{\mathbf{d}}_1 + \lambda_2 \hat{\mathbf{d}}_2 + \lambda_3 \hat{\mathbf{d}}_3$

In order to obtain the expressions associated to the remaining cases, the same line of thought –with minor differences– can be applied.

1) *One point-point coincidence constraint*  $d(\mathbf{P}_m, \mathbf{P}_f) = 0$ : Both  $\mathbf{S}_m$  and  $\mathbf{S}_f$  belong to point subspaces, so (6) becomes

$$\mathbf{T} = \mathbf{P}_f - \mathbf{R}(\mathbf{P}_m). \quad (7)$$

2) *One point-point distance constraint*  $d(\mathbf{P}_m, \mathbf{P}_f) = p$ : The mobile point  $\mathbf{P}_m$  is constrained to the surface of a sphere with center  $\mathbf{P}_f$  and radius  $p$ , as shown in Fig. 6a, so  $\mathbf{S}_m$  belongs to a point subspace and  $\mathbf{S}_f$  belongs to a spherical subspace:

$$\mathbf{T}(\alpha, \beta) = \underbrace{\mathbf{P}_f + p(c_\alpha s_\beta \hat{\mathbf{i}} + s_\alpha s_\beta \hat{\mathbf{j}} + c_\beta \hat{\mathbf{k}})}_{\mathbf{S}_f} - \underbrace{\mathbf{R}(\mathbf{P}_m)}_{\mathbf{S}_m}. \quad (8)$$

3) *One point-line coincidence constraint*  $d(\mathbf{P}_f, \mathcal{L}_m) = 0$ : The mobile line  $\mathcal{L}_m$  is constrained to contain the fixed point  $\mathbf{P}_f$  at all times, so  $\mathbf{S}_m$  belongs to a line subspace and  $\mathbf{S}_f$  belongs to a point subspace:

$$\mathbf{T}(\lambda) = \underbrace{\mathbf{P}_f}_{\mathbf{S}_f} - \underbrace{\mathbf{R}(\mathbf{P}_m + \lambda \hat{\mathbf{d}}_{\mathcal{L}_m})}_{\mathbf{S}_m} \quad (9)$$

where  $\hat{\mathbf{d}}_{\mathcal{L}_m}$  and  $\mathbf{P}_m$  are the direction vector and a support point of  $\mathcal{L}_m$ , respectively.

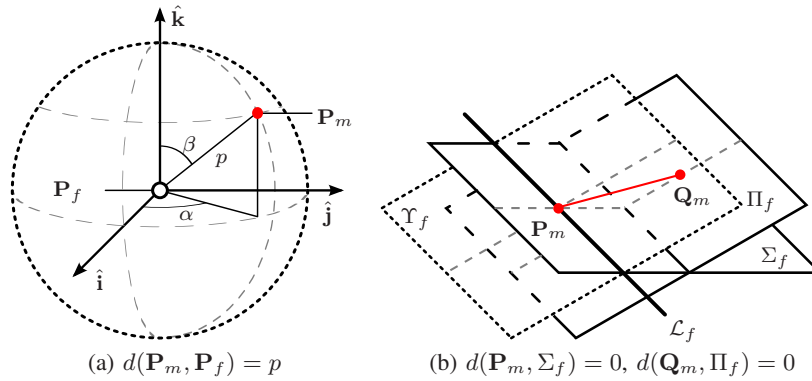


Fig. 6. Two particular solutions of the translational component.

(a) One *point-point* distance constraint.

(b) Two *point-plane* coincidence constraints.

4) *One point-line distance and one point-plane coincidence constraints*  $d(\mathbf{P}_m, \mathcal{L}_f) = p$ ,  $d(\mathbf{P}_m, \Pi_f) = 0$  : The mobile point  $\mathbf{P}_m$  is constrained to an ellipse (or circle), so  $\mathbf{S}_m$  belongs to a point subspace and  $\mathbf{S}_f$  belongs to an elliptical subspace:

$$\mathbf{T}(\alpha) = \underbrace{\mathbf{R}_f + p \left[ c_\alpha \hat{\mathbf{d}}_1 + (s_\alpha / s_{\sigma_f}) \hat{\mathbf{d}}_2 \right]}_{\mathbf{S}_f} - \underbrace{\mathbf{R}(\mathbf{P}_m)}_{\mathbf{S}_m} \quad (10)$$

where

$$\begin{aligned} \mathbf{R}_f &= \mathcal{L}_f \cap \Pi_f \\ \hat{\mathbf{d}}_1 &= \hat{\mathbf{n}}_{\Pi_f} \times \hat{\mathbf{d}}_{\mathcal{L}_f} \\ \hat{\mathbf{d}}_2 &= \hat{\mathbf{d}}_1 \times \hat{\mathbf{n}}_{\Pi_f}. \end{aligned}$$

5) *Two point-plane coincidence constraints*  $d(\mathbf{P}_m, \Sigma_f) = 0$ ,  $d(\mathbf{Q}_m, \Pi_f) = 0$  : To solve this case the two constraints are transformed into an equivalent pair of *point-plane* constraints that share the same mobile point.

Stating (6) for the second constraint yields

$$\mathbf{T} = \mathbf{Q}_f - \mathbf{R}(\mathbf{Q}_m) \quad (11)$$

where  $\mathbf{Q}_f$  is a point contained in the planar subapace associated to  $\Pi_f$ . Adding and subtracting  $\mathbf{P}_m$  and rearranging

$$\mathbf{T} = \mathbf{Q}_f - \mathbf{R}(\mathbf{Q}_m + \mathbf{P}_m - \mathbf{P}_m) \quad (12)$$

$$\mathbf{T} = [\mathbf{Q}_f - \mathbf{R}(\mathbf{Q}_m - \mathbf{P}_m)] - \mathbf{R}(\mathbf{P}_m). \quad (13)$$

Defining  $\mathbf{S}'_f = \mathbf{Q}_f - \mathbf{R}(\mathbf{Q}_m - \mathbf{P}_m)$ , (13) becomes

$$\mathbf{T} = \mathbf{S}'_f - \mathbf{R}(\mathbf{P}_m). \quad (14)$$

Equation (14) is equivalent to the *point-plane* coincidence constraint  $d(\mathbf{P}_m, \Upsilon_f) = 0$ , where  $\Upsilon_f$  is the plane parallel to  $\Pi_f$  that passes through  $\mathbf{S}'_f$ .

The original constraint pair can be now expressed through the equivalent  $d(\mathbf{P}_m, \Sigma_f) = 0$ ,  $d(\mathbf{P}_m, \Upsilon_f) = 0$ , that shares the same mobile point, which in turn can be further simplified into the case of one *point-line* coincidence constraint  $d(\mathbf{P}_m, \mathcal{L}_f) = 0$ , where  $\mathcal{L}_f = \Sigma_f \cap \Upsilon_f$ . Notice that  $\mathcal{L}_f$  depends on  $\mathbf{R}$  and must be recomputed every time its value changes. The solution is depicted in Fig. 6b.

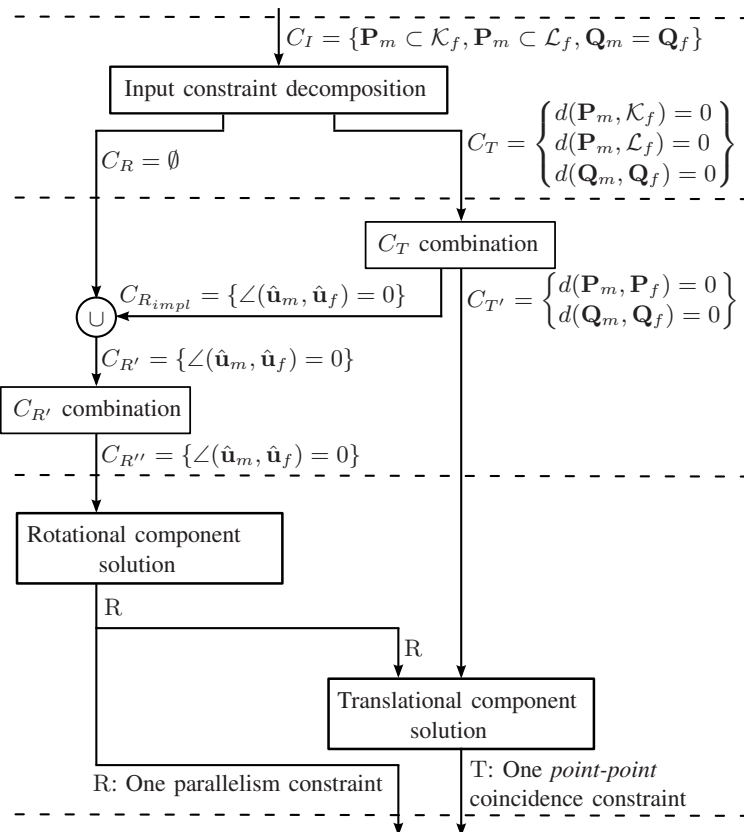


Fig. 7. Solution process for a problem involving the objects from Figs. 1a/1b.

## VI. SAMPLE PROBLEMS

This section presents three examples that illustrate how PMF works and how it can be applied to its target applications. The first example details the solution process for a simple problem and emphasizes the contribution of each step. The second example shows how to specify a simple offline programmed assembly task using sets of geometric constraints, and emphasizes on how to define trajectory reference points with well-constrained configurations. The third example consists of a teleoperated painting task, and it shows how task execution can be improved by partially restricting the motion of the operator to a subspace of interest.

### A. A simple example

Consider one more time the fixed and mobile objects depicted in Figs. 1a and 1b, respectively. The mobile object is to be positioned according to the input constraint set

$$C_I = \{\mathbf{P}_m \subset \mathcal{K}_f, \mathbf{P}_m \subset \mathcal{L}_f, \mathbf{Q}_m = \mathbf{Q}_f\}$$

where  $\mathbf{P}_m = [0 \ 5 \ 3]^T$ ,  $\mathbf{Q}_m = [0 \ 7 \ 3]^T$ ,  $\mathbf{Q}_f = [-2 \ 0 \ 3]^T$ ; and  $\mathcal{K}_f, \mathcal{L}_f$  are the lines with direction vectors  $\hat{\mathbf{d}}_{\mathcal{K}_f} = [0 \ 1 \ 0]^T$ ,  $\hat{\mathbf{d}}_{\mathcal{L}_f} = [0 \ 0 \ 1]^T$ , and share the support point  $\mathbf{P}_f = [0 \ 0 \ 3]^T$ .

Fig. 7 outlines the solution process for this problem, and shows how constraint sets are affected by each step. The solution, depicted in Fig. 1d, has one rotational DOF.

1) *Input constraint decomposition:* According to Table I, the constraints in  $C_I$  become

$$\begin{aligned} C_T &= \{d(\mathbf{P}_m, \mathcal{K}_f) = 0, d(\mathbf{P}_m, \mathcal{L}_f) = 0, d(\mathbf{Q}_m, \mathbf{Q}_f) = 0\} \\ C_R &= \emptyset. \end{aligned}$$

2) *Constraint combination:*

- The first two constraints share the same mobile point, and since  $\mathcal{K}_f \cap \mathcal{L}_f = \mathbf{P}_f$ , they can be rewritten as  $d(\mathbf{P}_m, \mathbf{P}_f) = 0$ , yielding

$$C_{T'} = \{d(\mathbf{P}_m, \mathbf{P}_f) = 0, d(\mathbf{Q}_m, \mathbf{Q}_f) = 0\}.$$

- Since  $d(\mathbf{P}_m, \mathbf{Q}_m) = d(\mathbf{P}_f, \mathbf{Q}_f)$ , the two constraints in  $C_{T'}$  imply the rotation  $\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = 0$ , where  $\hat{\mathbf{u}}_m = [0 \ 1 \ 0]^T$  points in the direction of  $\overrightarrow{\mathbf{P}_m \mathbf{Q}_m}$  and  $\hat{\mathbf{u}}_f = [-1 \ 0 \ 0]^T$  points in the direction of  $\overrightarrow{\mathbf{P}_f \mathbf{Q}_f}$ ,

$$C_{R'} = \{\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = 0\}.$$

3) *Solution synthesis:*

- *Rotational component:* R is computed for one parallelism constraint (Section V-A.3) and has one DOF.

Let  $\hat{\mathbf{w}} = \hat{\mathbf{u}}_m \times \hat{\mathbf{u}}_f = [0 \ 0 \ 1]^T$  and  $\sigma = \pi/2$  be the angle between  $\hat{\mathbf{u}}_m$  and  $\hat{\mathbf{u}}_f$ , then

$$\begin{aligned} \mathbf{R}(\phi) &= \mathbf{R}(\hat{\mathbf{u}}_f, \phi) \mathbf{R}(\hat{\mathbf{w}}, \sigma) \\ \mathbf{R}(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ \mathbf{R}(\phi) &= \begin{bmatrix} 0 & -1 & 0 \\ c_\phi & 0 & s_\phi \\ -s_\phi & 0 & c_\phi \end{bmatrix}. \end{aligned} \tag{15}$$

- *Translational component:*  $C_{T'}$  is matched to the most restrictive entry of Table IV, hence T is computed for one *point-point* coincidence constraint (Section V-B.1), and has no DOFs.

$$\begin{aligned} \mathbf{T} &= \mathbf{P}_f - \mathbf{R}(\mathbf{P}_m) \\ \mathbf{T} &= [0 \ 0 \ 3]^T - \mathbf{R}[0 \ 5 \ 3]^T. \end{aligned} \tag{16}$$

Particular instances of the solution can be found by evaluating the DOF parameter. For example, setting  $\phi = 0$  yields the rigid transformation  $P$ :

$$P = \begin{bmatrix} 0 & -1 & 0 & 5 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{17}$$

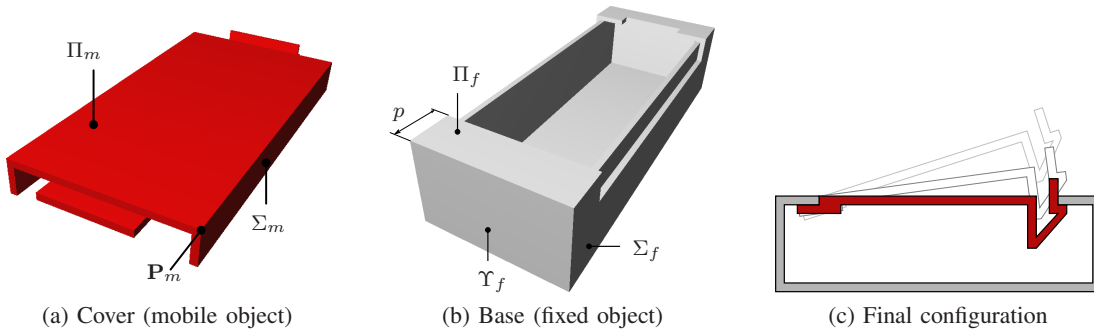


Fig. 8. Assembly task consisting on positioning *cover* (a) with respect to *base* (b) as shown in (c).

### B. An assembly task

Consider the objects *cover* and *base*, depicted in Figs. 8a and 8b, respectively. The task consists on securing *cover* (which is being grasped by a robot arm) to *base* as shown in Fig. 8c. Notice that since *cover* features a spring-activated locking mechanism, it must be positioned following the right sequence of configurations rather than directly heading to its final state. Consider now the following sequence of configurations for successfully performing the task:

- (a) Position *cover* at an angle with respect to *base* ensuring the objects do not collide (Fig. 9a).
- (b) Align *cover* with the rails on *base*'s sides (Fig. 9b).
- (c) Translate *cover* along the rails (Fig. 9c).
- (d) Rotate *cover* until the spring on its front locks (Figs. 9d and 8c).

Configuration (a) (Fig. 9a) is represented with a constraint set that fully restricts *cover*'s motion:

$$C_I = \left\{ \begin{array}{l} \Sigma_m = \Sigma_f, \\ d(\mathbf{P}_m, \Pi_f) = d_v, \\ d(\mathbf{P}_m, \Upsilon_f) = p + d_h, \\ \angle(\Pi_m, \Pi_f) = \alpha \end{array} \right\}.$$

Configurations (b), (c), and (d) are then reached by sequentially setting to zero  $d_v$ ,  $d_h$ , and  $\alpha$ , respectively (Figs. 9b-9d).

One important remark is that if the constraints are defined symbolically (i.e., using references to the geometric entities instead of their particular values at some instant in time), the task need not be redefined if the initial configurations of *cover* or *base* change, the solver will automatically recomputes the new solutions when necessary. This enables changing the distribution of the robot workcell with minimal downtime and no task reprogramming.

### C. A teleoperated spray-painting task

Consider a spray-painting teleoperated task in which an operator must paint a free curve on a plane surface. Suppose now that to ensure optimal paint covering the nozzle of the spray painter must remain at a fixed constant distance from the painted surface as well as normal to it. If the operator is to perform successfully the task, he/her must simultaneously trace the desired curve while satisfying the above relations.

Geometric constraints can be used to enforce the painting distance and orientation conditions, hence lowering the burden on the operator and permitting him/her to concentrate on tracing the curves. A scheme depicting the task is shown in Fig. 10.

By defining the constraint set  $C_I = \{d(\Pi_m, \Pi_f) = p\}$ , the plane  $\Pi_m$ , that contains the nozzle and is normal to it, will remain parallel and at a distance  $p$  from the painting plane  $\Pi_f$ . The

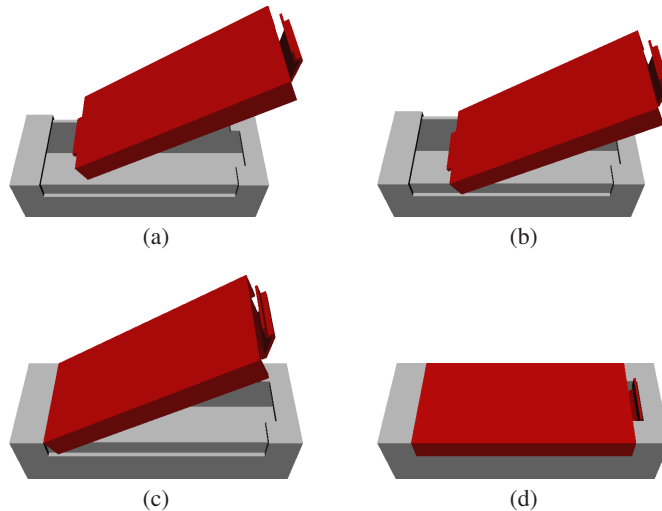


Fig. 9. Sequence followed to accomplish the assembly. Robot gripper is handling *cover*, but is not shown for clarity.

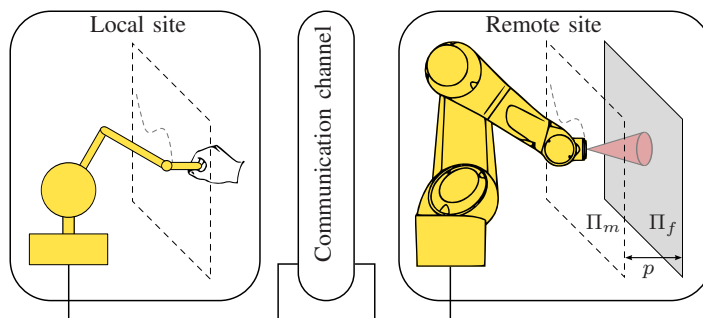


Fig. 10. A teleoperated spray-painting task.

solution subspace, which permits translations along a plane and rotations about its normal, is sent to a haptic guidance module that generates virtual forces on the operator to maintain it inside the subspace. Details on how this module works as well as further examples like peg-in-hole insertions and following a guide can be found in [18], [19].

## VII. PERFORMANCE AND IMPLEMENTATION ISSUES

### A. Time complexity

In the following, the time complexity of the methodology is proven to be quadratic  $O(n^2)$  – where  $n$  represents the number of input constraints– by showing that the individual steps that comprise the solution process are at most  $O(n^2)$ :

*Input constraint decomposition* has  $O(n)$  complexity because the decomposition of one input constraint into fundamental ones is a constant time operation that must be performed once for each input constraint. Since the decomposition of an input constraint produces at most one translational and one rotational fundamental constraint (Table I), the number of fundamental constraints is upper bounded to  $n$  translational and  $n$  rotational ones.

*Constraint combination* has  $O(n^2)$  complexity. Given the combinatorial nature of this step, the maximum number of combination tests –which are constant time operations– that are performed on a problem instance is upper bounded by  $\binom{n}{2} = n(n-1)/2$ . This bound is rarely reached because the number of fundamental constraints usually decreases during the constraint reduction steps. Additionally, since each geometric constraint restricts at least one DOF, the number of input

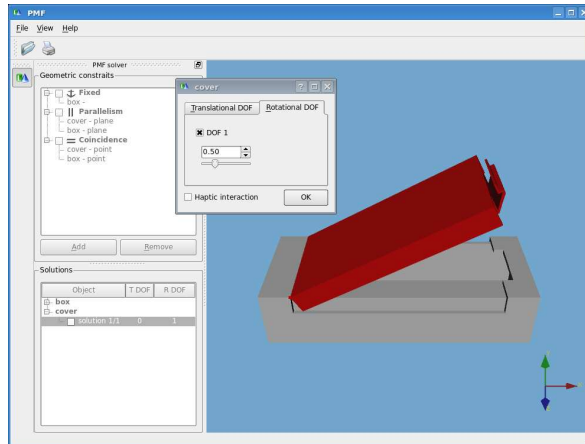


Fig. 11. Screenshot of the solver user interface.

constraints for most practical problems –including those with multiple redundant constraints– is usually small.

*Solution synthesis* is a constant time operation  $O(1)$ .

### B. Implementation

The current implementation of the PMF solver is written in C++. Its performance has been measured on a desktop PC with an Intel Pentium 4 processor running at 3.4GHz. The solution times range from 0.05ms for a simple problem with one constraint and one solution, to 1.5ms for a fully constrained problem with five constraints and thirty two distinct solutions. In contrast, methods containing iterative processes are more sensitive to singularities and degeneracies, and tend to have much greater variations in their solution times. Comparatively, the examples of [14] featuring one mobile object have solution time variations of up to five orders of magnitude, as opposed to two for PMF.

Fig. 11 shows a screenshot of the solver user interface that includes a 3D model of the task (right) and a control panel (left) that permits the graphical interactive definition of input constraints, displays information regarding the solutions of the current problem, and permits moving the mobile object along its current solution subspace with a mouse/keyboard or a haptic device.

## VIII. CONCLUSIONS AND FUTURE WORK

A relational positioning methodology for flexibly and intuitively specifying offline programmed robot tasks, as well as for assisting the execution of teleoperated tasks demanding precise movements has been presented.

It was shown how an object positioning operation can be formulated in terms of geometric constraints for restricting totally or partially the movements of an object independently of its initial configuration. As a means to find the solutions to such a problem, PMF, a 3D sequential geometric constraint solver has been proposed. The solver exploits the fact that in sets of geometric constraints, the rotational component can often be separated from the translational one and solved independently. PMF can handle under-, well-, and overconstrained problems with multiple solutions; and although it is not complete, the solvable subset handles most of the problems a user would be interested in. It has been demonstrated that the solution process has quadratic time complexity  $O(n^2)$  for the number of input constraints, and experimental data shows that the solution times of the current implementation allow real-time interaction with a human operator and the inclusion

of the solver in high-frequency loops that require response times within the millisecond order of magnitude.

Future lines of work comprise extending the solver's capabilities for handling multiple mobile objects and interfacing it with kinematics and path planning modules to enable the specification and execution of more complex (multi)robot tasks.

#### APPENDIX I ORTHONORMAL BASIS ASSOCIATED TO TWO VECTORS

Given two independent unit vectors  $\hat{\mathbf{u}}$  and  $\hat{\mathbf{v}}$ , the orthonormal basis associated to them is  $\{\hat{\mathbf{u}}, \hat{\mathbf{v}}_{\mathbf{u}}, \hat{\mathbf{w}}\}$ , where

$$\hat{\mathbf{v}}_{\mathbf{u}} = \frac{\hat{\mathbf{v}} - (\hat{\mathbf{u}} \cdot \hat{\mathbf{v}})\hat{\mathbf{u}}}{|\hat{\mathbf{v}} - (\hat{\mathbf{u}} \cdot \hat{\mathbf{v}})\hat{\mathbf{u}}|} \quad (\text{A-1})$$

$$\hat{\mathbf{w}} = \hat{\mathbf{u}} \times \hat{\mathbf{v}}_{\mathbf{u}}. \quad (\text{A-2})$$

#### APPENDIX II COMPUTATION OF $\hat{\mathbf{s}}_{\hat{\mathbf{u}}, \hat{\mathbf{v}}, \alpha, \beta}^{\pm}$

Given two independent unit vectors  $\hat{\mathbf{u}}$  and  $\hat{\mathbf{v}}$ , and two nonzero angles  $\alpha$  and  $\beta$ , there exist up to two unit vectors  $\hat{\mathbf{s}}_{\hat{\mathbf{u}}, \hat{\mathbf{v}}, \alpha, \beta}^+$  and  $\hat{\mathbf{s}}_{\hat{\mathbf{u}}, \hat{\mathbf{v}}, \alpha, \beta}^-$  that form an angle of  $\alpha$  with  $\hat{\mathbf{u}}$  and an angle of  $\beta$  with  $\hat{\mathbf{v}}$ . These vectors can be expressed in the orthonormal basis  $\{\hat{\mathbf{u}}, \hat{\mathbf{v}}_{\mathbf{u}}, \hat{\mathbf{w}}\}$  (see Appendix I) associated to  $\hat{\mathbf{u}}$  and  $\hat{\mathbf{v}}$  as:

$$\hat{\mathbf{s}}_{\hat{\mathbf{u}}, \hat{\mathbf{v}}, \alpha, \beta}^{\pm} = c_{\alpha}\hat{\mathbf{u}} + c_{\phi}\hat{\mathbf{v}}_{\mathbf{u}} \pm c_{\theta}\hat{\mathbf{w}}, \quad (\text{A-3})$$

where

$$\begin{aligned} c_{\phi} &= (c_{\beta} - c_{\alpha}c_{\sigma})/s_{\sigma} \\ c_{\theta} &= (s_{\alpha}^2 - c_{\phi}^2)^{1/2} \\ \sigma &= \cos^{-1}(\hat{\mathbf{u}} \cdot \hat{\mathbf{v}}). \end{aligned}$$

#### APPENDIX III ROTATION MATRICES

##### A. Rotation defined by two pairs of vectors

Given two pairs of independent unit vectors  $\{\hat{\mathbf{u}}, \hat{\mathbf{v}}\}$  and  $\{\hat{\mathbf{r}}, \hat{\mathbf{s}}\}$ , the rotation  $R(\hat{\mathbf{u}}, \hat{\mathbf{v}} \rightarrow \hat{\mathbf{r}}, \hat{\mathbf{s}})$  places  $\hat{\mathbf{u}}$  parallel to  $\hat{\mathbf{r}}$  and  $\hat{\mathbf{v}}$  in the plane defined by  $\hat{\mathbf{r}}$  and  $\hat{\mathbf{s}}$ . If  $\hat{\mathbf{u}} \cdot \hat{\mathbf{v}} = \hat{\mathbf{r}} \cdot \hat{\mathbf{s}}$ , then  $\hat{\mathbf{v}}$  will additionally be parallel to  $\hat{\mathbf{s}}$ .

$$R(\hat{\mathbf{u}}, \hat{\mathbf{v}} \rightarrow \hat{\mathbf{r}}, \hat{\mathbf{s}}) = \begin{bmatrix} \hat{\mathbf{r}} & \hat{\mathbf{s}}_{\mathbf{r}} & \hat{\mathbf{t}} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}} & \hat{\mathbf{v}}_{\mathbf{u}} & \hat{\mathbf{w}} \end{bmatrix}^T \quad (\text{A-4})$$

where  $\{\hat{\mathbf{u}}, \hat{\mathbf{v}}_{\mathbf{u}}, \hat{\mathbf{w}}\}$  and  $\{\hat{\mathbf{r}}, \hat{\mathbf{s}}_{\mathbf{r}}, \hat{\mathbf{t}}\}$  are the orthonormal basis associated to  $\{\hat{\mathbf{u}}, \hat{\mathbf{v}}\}$  and  $\{\hat{\mathbf{r}}, \hat{\mathbf{s}}\}$ , respectively.

##### B. Rotation defined by an axis-angle pair

Given a unit vector  $\hat{\mathbf{u}} = (u_x, u_y, u_z)$  and a scalar angular value  $\phi$ , the rotation by  $\phi$  about the direction of  $\hat{\mathbf{u}}$  is given by

$$R(\hat{\mathbf{u}}, \phi) = \begin{bmatrix} au_x^2 + c_{\phi} & au_xu_y - u_zs_{\phi} & au_xu_z + u_ys_{\phi} \\ au_xu_y + u_zs_{\phi} & au_y^2 + c_{\phi} & au_yu_z - u_xs_{\phi} \\ au_xu_z - u_ys_{\phi} & au_yu_z + u_xs_{\phi} & au_z^2 + c_{\phi} \end{bmatrix} \quad (\text{A-5})$$

with  $a = 1 - c_{\phi}$ .

### C. Rotation defined by three angle constraints

Consider the three rotational constraints  $\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = \alpha$ ,  $\angle(\hat{\mathbf{v}}_m, \hat{\mathbf{v}}_f) = \beta$ , and  $\angle(\hat{\mathbf{w}}_m, \hat{\mathbf{w}}_f) = \gamma$ . Since the solution process is valid only for *positive* right trihedra, it may be necessary to manipulate the third constraint so that it satisfies  $\hat{\mathbf{u}}_m \times \hat{\mathbf{v}}_m = \hat{\mathbf{w}}_m$  and  $\hat{\mathbf{u}}_f \times \hat{\mathbf{v}}_f = \hat{\mathbf{w}}_f$ . The rotation  $\mathbf{R}$  can be calculated as the following composition

$$\mathbf{R} = \mathbf{R}_1 \mathbf{R}_2 \mathbf{R}_3. \quad (\text{A-6})$$

Let  $(\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}})$  be the axis of the fixed coordinate system. The computation of  $\mathbf{R}_1$  and  $\mathbf{R}_3$  is straightforward

$$\mathbf{R}_1(\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}} \rightarrow \hat{\mathbf{u}}_f, \hat{\mathbf{v}}_f, \hat{\mathbf{w}}_f) = \begin{bmatrix} \hat{\mathbf{u}}_f & \hat{\mathbf{v}}_f & \hat{\mathbf{w}}_f \end{bmatrix} \quad (\text{A-7})$$

$$\mathbf{R}_3(\hat{\mathbf{u}}_m, \hat{\mathbf{v}}_m, \hat{\mathbf{w}}_m \rightarrow \hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}) = \begin{bmatrix} \hat{\mathbf{u}}_m & \hat{\mathbf{v}}_m & \hat{\mathbf{w}}_m \end{bmatrix}^T. \quad (\text{A-8})$$

$\mathbf{R}_2$  has the form

$$\mathbf{R}_2(\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}} \rightarrow \hat{\mathbf{a}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}) = \begin{bmatrix} \hat{\mathbf{a}} & \hat{\mathbf{b}} & \hat{\mathbf{c}} \end{bmatrix} \quad (\text{A-9})$$

where  $\hat{\mathbf{a}} \cdot \hat{\mathbf{i}} = c_\alpha$ ,  $\hat{\mathbf{b}} \cdot \hat{\mathbf{j}} = c_\beta$ , and  $\hat{\mathbf{c}} \cdot \hat{\mathbf{k}} = c_\gamma$ .

Expressing the rotation in (A-9) in terms of the Euler<sub>XZX</sub> angles  $(\theta_1, \phi, \theta_2)$  and solving yields  $\phi = \alpha$  and up to eight distinct combinations for  $\theta_1$  and  $\theta_2$ . The first four –listed as  $\{\theta_1, \theta_2\}$  pairs– are  $\{k_1, k_2\}$ ,  $\{-k_1, -k_2\}$ ,  $\{\pi + k_1, \pi + k_2\}$ , and  $\{\pi - k_1, \pi - k_2\}$  with

$$\begin{aligned} k_1 &= (\psi_1 + \psi_2)/2 \\ k_2 &= (\psi_1 - \psi_2)/2 \\ \psi_1 &= \cos^{-1} [(c_\gamma + c_\beta)(1 - c_\alpha)/s_\alpha^2] \\ \psi_2 &= \cos^{-1} [(c_\gamma - c_\beta)(1 + c_\alpha)/s_\alpha^2]. \end{aligned}$$

The last four pairs are obtained by interchanging the values of  $\theta_1$  and  $\theta_2$ .

### D. Rotation defined by two angle constraints

Consider the rotational constraints  $\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}_f) = \alpha$  and  $\angle(\hat{\mathbf{v}}_m, \hat{\mathbf{v}}_f) = \beta$ , and let  $\sigma_m$  and  $\sigma_f$  be angles between the mobile and fixed vectors, respectively. The rotation  $\mathbf{R}(\phi)$  permits limited rotations along one direction and can be computed by the composition of three rotations

$$\mathbf{R}(\phi) = \mathbf{R}(\hat{\mathbf{u}}_f, \phi) \mathbf{R}_B \mathbf{R}(\hat{\mathbf{u}}_m, \theta(\phi)). \quad (\text{A-10})$$

$\mathbf{R}(\hat{\mathbf{u}}_f, \phi)$  is the free rotation around the direction of  $\hat{\mathbf{u}}_f$ .

$\mathbf{R}_B$  is the constant rotation that enforces the first angular constraint and is computed according to Appendix III-A:

$$\mathbf{R}_B = \mathbf{R}(\hat{\mathbf{u}}_m, \hat{\mathbf{u}}'_f \rightarrow \hat{\mathbf{v}}_m, \hat{\mathbf{v}}'_f) \quad (\text{A-11})$$

where

$$\begin{aligned} \hat{\mathbf{u}}'_f &= \mathbf{R}(\hat{\mathbf{w}}, \alpha) \hat{\mathbf{u}}_f \\ \hat{\mathbf{v}}'_f &= \mathbf{R}(\hat{\mathbf{w}}, \alpha + \sigma_m) \hat{\mathbf{u}}_f \\ \hat{\mathbf{w}} &= \hat{\mathbf{v}}_f \times \hat{\mathbf{u}}_f. \end{aligned}$$

$\mathbf{R}(\hat{\mathbf{u}}_m, \theta(\phi))$  enforces the second angular constraint.

Since angles  $\sigma_f$ ,  $\alpha$ ,  $\sigma_m$  and  $\beta$  form the sides of a spherical quadrilateral,  $\phi$  is the exterior angle between  $\sigma_f$  and  $\alpha$ , and  $\theta$  is the exterior angle between  $\alpha$  and  $\sigma_m$ , the value of  $\theta$  can be computed using the expression that relates contiguous exterior angles of a spherical quadrilateral [20]

$$\theta(\phi) = 2 \arctan(a/b) \quad (\text{A-12})$$

$$a = -xs_{\sigma_m} \pm [(s_{\sigma_m}s_{\beta})^2 - (z - c_{\sigma_m}c_{\beta})^2]^{1/2} \quad (\text{A-13})$$

$$b = zc_{\sigma_m} - ys_{\sigma_m} - c_{\beta}$$

$$x = s_{\sigma_f}s_{\phi}$$

$$y = -s_{\alpha}c_{\sigma_f} - c_{\alpha}s_{\sigma_f}c_{\phi}$$

$$z = c_{\alpha}c_{\sigma_f} - s_{\alpha}s_{\sigma_f}c_{\phi}.$$

When  $z = \cos(\sigma_m \pm \beta)$ , the square root in (A-13) becomes zero, so the extrema of the validity intervals for  $\phi$  are given by

$$c_{\phi_1} = [c_{\alpha}c_{\sigma_f} - c_{(\sigma_m+\beta)}] / s_{\alpha}s_{\sigma_f} \quad (\text{A-14})$$

$$c_{\phi_2} = [c_{\alpha}c_{\sigma_f} - c_{(\sigma_m-\beta)}] / s_{\alpha}s_{\sigma_f}. \quad (\text{A-15})$$

Depending on the values of  $c_{\phi_1}$  and  $c_{\phi_2}$ ,  $\phi$  may take any possible value, belong to an interval, belong to two symmetrical intervals, or the degenerate cases of one and two discrete values:

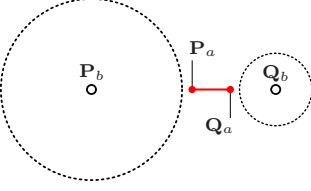
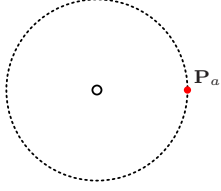
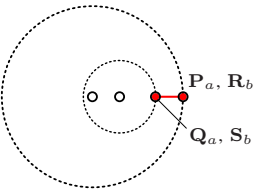
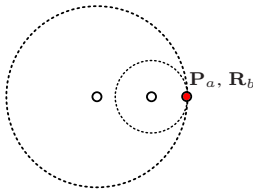
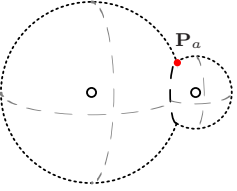
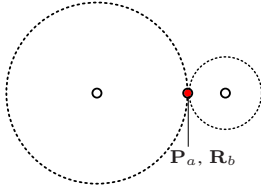
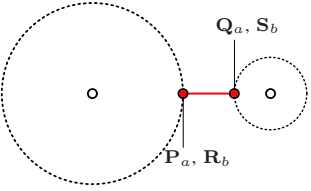
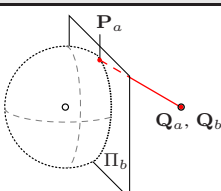
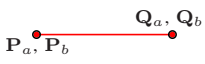
$$\phi \in \begin{cases} [0, 2\pi] & \text{if } (c_{\phi_1} \notin A) \wedge (c_{\phi_2} \notin A) \\ [\phi_1, 2\pi - \phi_1] & \text{if } (c_{\phi_1} \in A) \wedge (c_{\phi_2} \notin A) \\ [-\phi_2, \phi_2] & \text{if } (c_{\phi_1} \notin A) \wedge (c_{\phi_2} \in A) \\ B & \text{if } (c_{\phi_1} \in A) \wedge (c_{\phi_2} \in A) \end{cases} \quad (\text{A-16})$$

where  $A = [-1, 1]$  and  $B$  is a set of closed intervals defined as:

$$B = \begin{cases} [\phi_1, \phi_2] \cup [-\phi_2, -\phi_1] & \text{if } \phi_1 \leq \phi_2 \\ [\phi_2, \phi_1] \cup [-\phi_1, -\phi_2] & \text{if } \phi_1 > \phi_2. \end{cases} \quad (\text{A-17})$$

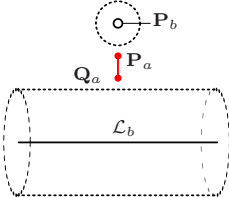

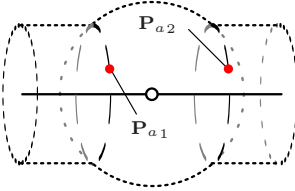
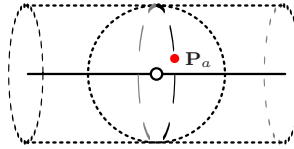
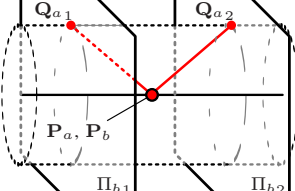
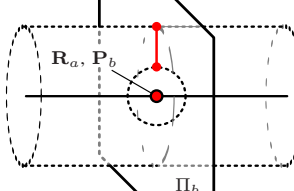
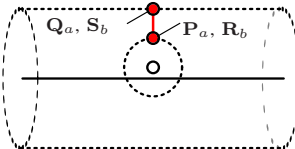
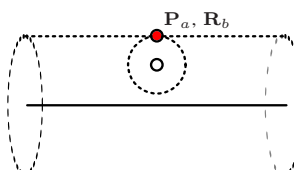
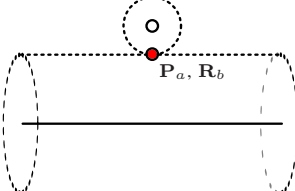
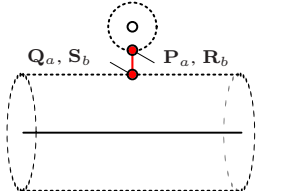
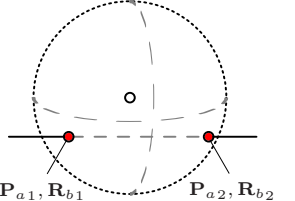
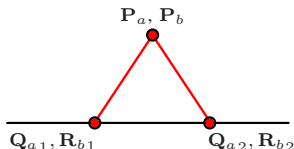
APPENDIX IV  
CONSTRAINT COMBINATION TABLES

TABLE I  
TWO POINT-POINT DISTANCE CONSTRAINTS  $d(\mathbf{P}_a, \mathbf{P}_b) = p$ ,  $d(\mathbf{Q}_a, \mathbf{Q}_b) = q$

$(d_a + d_b <  p - q ) \vee ( d_b - d_a  > p + q)$ <b>Solutions:</b> 0 (Incompatible) 	$(p = q) \wedge (d_a = 0) \wedge (d_b = 0)$ <b>Solutions:</b> 1 $d(\mathbf{P}_a, \mathbf{P}_b) = p$ 
$(d_a + d_b =  p - q ) \wedge (d_a \neq 0) \wedge (d_b \neq 0)$ <b>Solutions:</b> 1 $d(\mathbf{P}_a, \mathbf{R}_b) = 0$ $d(\mathbf{Q}_a, \mathbf{S}_b) = 0$ with $\mathbf{R}_b = \mathbf{P}_b + \text{sgn}(p - q)p\hat{\mathbf{u}}_b$ $\mathbf{S}_b = \mathbf{Q}_b + \text{sgn}(p - q)q\hat{\mathbf{u}}_b$ 	$(d_b =  p - q ) \wedge (d_a = 0) \wedge (d_b \neq 0)$ <b>Solutions:</b> 1 $d(\mathbf{P}_a, \mathbf{R}_b) = 0$ with $\mathbf{R}_b = \mathbf{P}_b + \text{sgn}(p - q)p\hat{\mathbf{u}}_b$ 
$(d_b >  p - q ) \wedge (d_b < p + q) \wedge (d_a = 0) \wedge (d_b \neq 0)$ <b>Solutions:</b> 1 $d(\mathbf{P}_a, \mathcal{L}_b) = r$ $d(\mathbf{P}_a, \Pi_b) = 0$ with $\mathbf{R}_b = \mathbf{P}_b + k\hat{\mathbf{u}}_b$ $k = (d_b^2 + p^2 - q^2)/(2d_b)$ $r = (p^2 - k^2)^{1/2}$ 	$(d_b = p + q) \wedge (d_a = 0) \wedge (d_b \neq 0)$ <b>Solutions:</b> 1 $d(\mathbf{P}_a, \mathbf{R}_b) = 0$ with $\mathbf{R}_b = \mathbf{P}_b + p\hat{\mathbf{u}}_b$ 
$(d_b - d_a = p + q \neq 0) \wedge (d_a \neq 0) \wedge (d_b \neq 0)$ <b>Solutions:</b> 1 $d(\mathbf{P}_a, \mathbf{R}_b) = 0$ $d(\mathbf{Q}_a, \mathbf{S}_b) = 0$ with $\mathbf{R}_b = \mathbf{P}_b + p\hat{\mathbf{u}}_b$ $\mathbf{S}_b = \mathbf{Q}_b - q\hat{\mathbf{u}}_b$ 	$[(d_a + d_b > p) \vee (d_b - d_a < p)] \wedge (d_a - d_b < p) \wedge (p \neq 0) \wedge (q = 0)$ <b>Solutions:</b> 1 $d(\mathbf{P}_a, \Pi_b) = 0$ $d(\mathbf{Q}_a, \mathbf{Q}_b) = 0$ with $\mathbf{R}_b = \mathbf{Q}_b - k\hat{\mathbf{u}}_b$ $k = (d_a^2 + d_b^2 - p^2)/(2d_b)$ 
$(d_a = d_b \neq 0) \wedge (p = 0) \wedge (q = 0)$ <b>Solutions:</b> 1 $d(\mathbf{P}_a, \mathbf{P}_b) = 0$ $d(\mathbf{Q}_a, \mathbf{Q}_b) = 0$ $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = 0$ 	

with  $\mathcal{L}_b(\hat{\mathbf{u}}_b, \mathbf{P}_b)$      $\Pi_b(\hat{\mathbf{u}}_b, \mathbf{R}_b)$      $\hat{\mathbf{u}}_a = \overrightarrow{\mathbf{P}_a\mathbf{Q}_a}/|\mathbf{P}_a\mathbf{Q}_a|$      $\hat{\mathbf{u}}_b = \overrightarrow{\mathbf{P}_b\mathbf{Q}_b}/|\mathbf{P}_b\mathbf{Q}_b|$

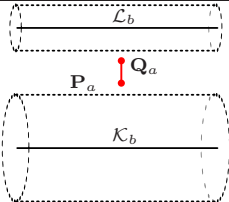
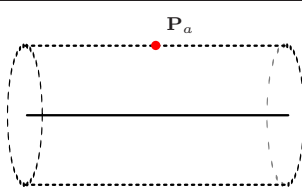
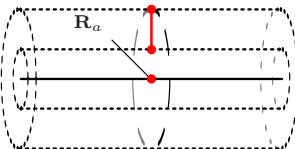
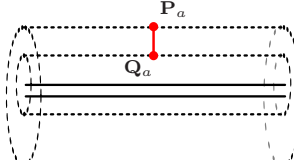
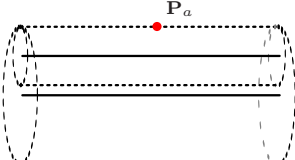
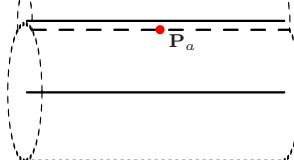
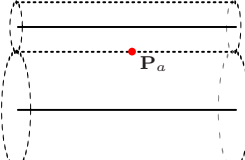
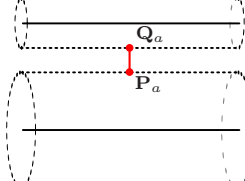
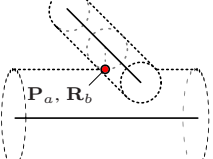
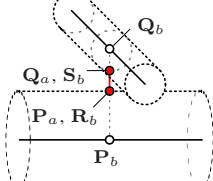
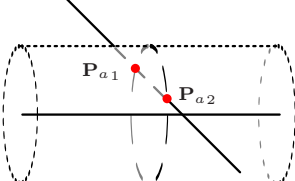
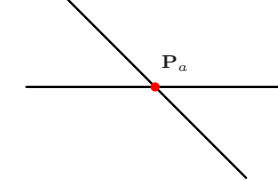
TABLE II  
POINT-POINT AND POINT-LINE DISTANCE CONSTRAINTS  $d(\mathbf{P}_a, \mathbf{P}_b) = p, d(\mathbf{Q}_a, \mathcal{L}_b) = q$

<p><math>(d_b - d_a &gt; p + q) \vee (d_a + d_b &lt; q - p)</math></p>  <p><b>Solutions:</b> 0 (Incompatible)</p>	<p><math>(p = 0) \wedge (q = 0) \wedge (d_a = 0) \wedge (d_b = 0)</math></p>  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathbf{P}_b) = 0</math></p>
<p><math>(p &gt; q) \wedge (q \neq 0) \wedge (d_a = 0) \wedge (d_b = 0)</math></p>  <p><b>Solutions:</b> 2 <math>d(\mathbf{P}_a, \mathcal{L}_b) = q</math> <math>d(\mathbf{P}_a, \mathbf{P}_b) = 0</math></p> <p>with <math>\Pi_b(\hat{\mathbf{d}}_{\mathcal{L}_b}, \mathbf{R}_b)</math> <math>\mathbf{R}_b = \mathbf{P}_b \pm k \hat{\mathbf{d}}_{\mathcal{L}_b}</math> <math>k = (p^2 - q^2)^{1/2}</math></p>	<p><math>(p = q) \wedge (d_a = 0) \wedge (d_b = 0)</math></p>  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathcal{L}_b) = q</math> <math>d(\mathbf{P}_a, \mathbf{P}_b) = 0</math></p> <p>with <math>\Pi_b(\hat{\mathbf{d}}_{\mathcal{L}_b}, \mathbf{P}_b)</math></p>
<p><math>(d_a &gt; q) \wedge (d_a \neq 0) \wedge (d_b = 0) \wedge (p = 0)</math></p>  <p><b>Solutions:</b> 2 <math>d(\mathbf{P}_a, \mathbf{P}_b) = 0</math> <math>d(\mathbf{Q}_a, \Pi_b) = 0</math></p> <p>with <math>\Pi_b(\hat{\mathbf{d}}_{\mathcal{L}_b}, \mathbf{R}_b)</math> <math>\mathbf{R}_b = \mathbf{P}_b \pm \hat{\mathbf{d}}_{\mathcal{L}_b} k</math> <math>k = (d_a^2 - q^2)^{1/2}</math></p>	<p><math>(d_a = q - p) \wedge (d_a \neq 0) \wedge (d_b = 0)</math></p>  <p><b>Solutions:</b> 1 <math>d(\mathbf{R}_a, \mathbf{P}_b) = 0</math> <math>d(\mathbf{R}_a, \Pi_b) = 0</math></p> <p>with <math>\mathbf{R}_a = \mathbf{P}_a - p \hat{\mathbf{u}}_a</math> <math>\Pi_b(\hat{\mathbf{d}}_{\mathcal{L}_b}, \mathbf{P}_b)</math></p>
<p><math>(d_a + d_b = q - p) \wedge (d_a \neq 0) \wedge (d_b \neq 0)</math></p>  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathbf{R}_b) = 0</math> <math>d(\mathbf{Q}_a, \mathbf{S}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathbf{P}_b - p \hat{\mathbf{u}}_b</math> <math>\mathbf{S}_b = \mathbf{Q}_b - q \hat{\mathbf{u}}_b</math></p>	<p><math>(d_b = q - p) \wedge (d_a = 0) \wedge (d_b \neq 0)</math></p>  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathbf{R}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathbf{P}_b - p \hat{\mathbf{u}}_b</math></p>
<p><math>(d_b = p + q) \wedge (d_a = 0) \wedge (d_b \neq 0)</math></p>  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathbf{R}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathbf{P}_b + p \hat{\mathbf{u}}_b</math></p>	<p><math>(d_b - d_a = p + q) \wedge (d_a \neq 0) \wedge (d_b \neq 0)</math></p>  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathbf{R}_b) = 0</math> <math>d(\mathbf{Q}_a, \mathbf{S}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathbf{P}_b + p \hat{\mathbf{u}}_b</math> <math>\mathbf{S}_b = \mathbf{Q}_b - q \hat{\mathbf{u}}_b</math></p>
<p><math>(d_b &lt; p) \wedge (d_a = 0) \wedge (q = 0)</math></p>  <p><b>Solutions:</b> 2 <math>d(\mathbf{P}_a, \mathbf{R}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathbf{Q}_b \pm k \hat{\mathbf{d}}_{\mathcal{L}_b}</math> <math>k = (p^2 - d_b^2)^{1/2}</math></p>	<p><math>(d_b &lt; d_a) \wedge (d_b \neq 0) \wedge (p = 0) \wedge (q = 0)</math></p>  <p><b>Solutions:</b> 2 <math>d(\mathbf{P}_a, \mathbf{P}_b) = 0</math> <math>d(\mathbf{Q}_a, \mathbf{R}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \text{proj}_{\mathcal{L}_b} \mathbf{P}_b \pm k \hat{\mathbf{d}}_{\mathcal{L}_b}</math> <math>k = (d_a^2 - d_b^2)^{1/2}</math></p>

with  $\mathbf{Q}_b = \text{proj}_{\mathcal{L}_b} \mathbf{P}_b$      $\hat{\mathbf{u}}_a = \overline{\mathbf{P}_a \mathbf{Q}_a} / |\mathbf{P}_a \mathbf{Q}_a|$      $\hat{\mathbf{u}}_b = \overline{\mathbf{P}_b \mathbf{Q}_b} / |\mathbf{P}_b \mathbf{Q}_b|$

TABLE III

TWO POINT-LINE DISTANCE CONSTRAINTS  $d(\mathbf{P}_a, \mathcal{K}_b) = p$ ,  $d(\mathbf{Q}_a, \mathcal{L}_b) = q$ 

$[(\mathcal{K}_b \parallel \mathcal{L}_b) \wedge (d_a + d_b <  p - q )] \vee (d_b - d_a > p + q)$  <p><b>Solutions:</b> 0 (Incompatible)</p>	$(\mathcal{K}_b \parallel \mathcal{L}_b) \wedge (p = q) \wedge (d_a = 0) \wedge (d_b = 0)$  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathcal{K}_f) = p</math></p>
$(\mathcal{K}_b \parallel \mathcal{L}_b) \wedge (d_a =  p - q ) \wedge (d_a \neq 0) \wedge (d_b = 0)$  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathcal{K}_b) = p</math> <math>d(\mathbf{R}_a, \mathcal{K}_b) = 0</math></p> <p>with <math>\mathbf{R}_a = \mathbf{P}_a + \text{sgn}(p - q)p\hat{\mathbf{u}}_a</math></p>	$(\mathcal{K}_b \parallel \mathcal{L}_b) \wedge (d_a + d_b =  p - q ) \wedge (d_a \neq 0) \wedge (d_b \neq 0)$  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathcal{M}_b) = 0</math> <math>d(\mathbf{Q}_a, \mathcal{N}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathbf{P}_b + \text{sgn}(p - q)p\hat{\mathbf{u}}_b</math> <math>\mathbf{S}_b = \mathbf{Q}_b + \text{sgn}(p - q)q\hat{\mathbf{u}}_b</math></p>
$(\mathcal{K}_b \parallel \mathcal{L}_b) \wedge (d_b =  p - q ) \wedge (d_a = 0) \wedge (d_b \neq 0)$  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathcal{M}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathbf{P}_b + \text{sgn}(p - q)p\hat{\mathbf{u}}_b</math></p>	$(\mathcal{K}_b \parallel \mathcal{L}_b) \wedge (d_b >  p - q ) \wedge (d_b < p + q) \wedge (d_a = 0) \wedge (d_b \neq 0)$  <p><b>Solutions:</b> 2 <math>d(\mathbf{P}_a, \mathcal{M}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathbf{P}_b + k_1\hat{\mathbf{u}}_b \pm k_2\hat{\mathbf{v}}_b</math> <math>k_1 = (d_b^2 + p^2 - q^2)/(2d_b)</math> <math>k_2 = (p^2 - k_1^2)^{1/2}</math> <math>\hat{\mathbf{v}}_b = \hat{\mathbf{d}}_{\mathcal{K}_b} \times \hat{\mathbf{u}}_b</math></p>
$(\mathcal{K}_b \parallel \mathcal{L}_b) \wedge (d_b = p + q) \wedge (d_a = 0) \wedge (d_b \neq 0)$  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathcal{M}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathbf{P}_b + p\hat{\mathbf{u}}_b</math></p>	$(\mathcal{K}_b \parallel \mathcal{L}_b) \wedge (d_b - d_a = p + q \neq 0) \wedge (d_a \neq 0) \wedge (d_b \neq 0)$  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathcal{M}_b) = 0</math> <math>d(\mathbf{Q}_a, \mathcal{N}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathbf{P}_b + p\hat{\mathbf{u}}_b</math> <math>\mathbf{S}_b = \mathbf{Q}_b - q\hat{\mathbf{u}}_b</math></p>
$(\mathcal{K}_b \nparallel \mathcal{L}_b) \wedge (d_b = p + q) \wedge (d_a = 0) \wedge (d_b \neq 0)$  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathbf{R}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathbf{P}_b + p\hat{\mathbf{u}}_b</math></p>	$(\mathcal{K}_b \nparallel \mathcal{L}_b) \wedge (d_b - d_a = p + q) \wedge (d_a \neq 0) \wedge (d_b \neq 0)$  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathbf{R}_b) = 0</math> <math>d(\mathbf{Q}_a, \mathbf{S}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathbf{P}_b + p\hat{\mathbf{u}}_b</math> <math>\mathbf{S}_b = \mathbf{Q}_b - q\hat{\mathbf{u}}_b</math> <math>(\mathbf{P}_b \subset \mathcal{K}_b) \wedge (d(\mathbf{P}_b, \mathcal{L}_b) = d_b)</math></p>
$(\mathcal{K}_b \nparallel \mathcal{L}_b) \wedge (d_b < q) \wedge (p = 0) \wedge (q \neq 0) \wedge (d_a = 0)$  <p><b>Solutions:</b> 2 <math>d(\mathbf{P}_a, \mathbf{R}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathbf{P}_b \pm k\hat{\mathbf{d}}_{\mathcal{K}_b}</math> <math>k = (q^2 - d_b^2)^{1/2}</math></p>	$(\mathcal{K}_b \nparallel \mathcal{L}_b) \wedge (d_a = 0) \wedge (d_b = 0) \wedge (p = 0) \wedge (q = 0)$  <p><b>Solutions:</b> 1 <math>d(\mathbf{P}_a, \mathbf{R}_b) = 0</math></p> <p>with <math>\mathbf{R}_b = \mathcal{K}_b \cap \mathcal{L}_b</math></p>

with  $\mathcal{M}_b(\hat{\mathbf{d}}_{\mathcal{K}_b}, \mathbf{R}_b)$

$\mathcal{N}_b(\hat{\mathbf{d}}_{\mathcal{K}_b}, \mathbf{S}_b)$

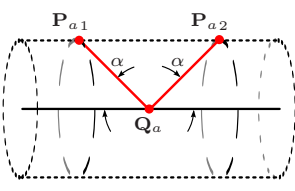
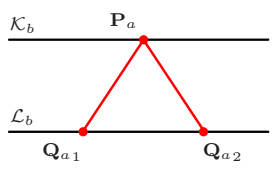
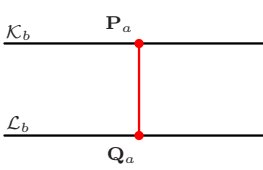
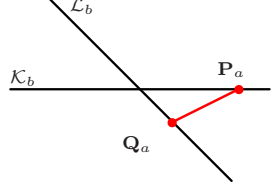
$\mathbf{P}_b \subset \mathcal{K}_b$

$\mathbf{Q}_b = \text{proj}_{\mathcal{L}_b} \mathbf{P}_b$

$\hat{\mathbf{u}}_a = \overline{\mathbf{P}_a \mathbf{Q}_a} / |\mathbf{P}_a \mathbf{Q}_a|$

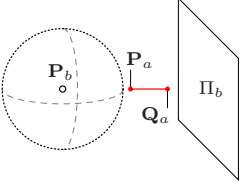
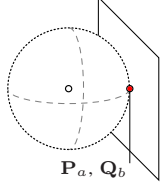
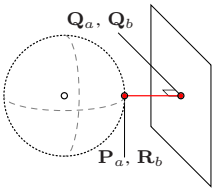
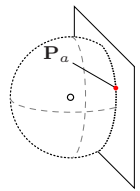
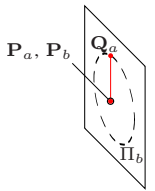
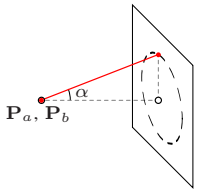
$\hat{\mathbf{u}}_b = \overline{\mathbf{P}_b \mathbf{Q}_b} / |\mathbf{P}_b \mathbf{Q}_b|$

TABLE IV  
TWO POINT-LINE DISTANCE CONSTRAINTS  $d(\mathbf{P}_a, \mathcal{K}_b) = p$ ,  $d(\mathbf{Q}_a, \mathcal{L}_b) = q$  (CONTINUED)

$(\mathcal{K}_b \parallel \mathcal{L}_b) \wedge (d_a \geq p) \wedge (d_b = 0) \wedge (q = 0)$	 <p><b>Solutions: 2</b>  <math>d(\mathbf{P}_a, \mathcal{K}_b) = p</math>  <math>d(\mathbf{Q}_a, \mathcal{L}_b) = 0</math>  <math>\angle(\hat{\mathbf{u}}_a, \pm \hat{\mathbf{d}}_{\mathcal{K}_b}) = \alpha</math></p> <p>with  <math>\alpha = \sin^{-1}(p/d_a)</math></p>	$(\mathcal{K}_b \parallel \mathcal{L}_b) \wedge (d_a > d_b) \wedge (p = 0) \wedge (q = 0)$	 <p><b>Solutions: 2</b>  <math>d(\mathbf{P}_a, \mathcal{K}_b) = 0</math>  <math>d(\mathbf{Q}_a, \mathcal{L}_b) = 0</math>  <math>\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{v}}_b) = 0</math></p> <p>with  <math>\hat{\mathbf{v}}_b = \overrightarrow{\mathbf{P}_b \mathbf{R}_b} /  \mathbf{P}_b \mathbf{R}_b </math>  <math>\mathbf{R}_b = \mathbf{Q}_b \pm k \hat{\mathbf{d}}_{\mathcal{L}_b}</math>  <math>k = (d_a^2 - d_b^2)^{1/2}</math></p>
$(\mathcal{K}_b \parallel \mathcal{L}_b) \wedge (d_a = d_b \neq 0) \wedge (p = 0) \wedge (q = 0)$	 <p><b>Solutions: 1</b>  <math>d(\mathbf{P}_a, \mathcal{K}_b) = 0</math>  <math>d(\mathbf{Q}_a, \mathcal{L}_b) = 0</math>  <math>\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = 0</math></p>	$(\mathcal{K}_b \not\parallel \mathcal{L}_b) \wedge (d_a \neq 0) \wedge (d_b = 0) \wedge (p = 0) \wedge (q = 0)$	 <p><b>Solutions: 1</b>  <math>d(\mathbf{P}_a, \mathcal{K}_b) = 0</math>  <math>d(\mathbf{Q}_a, \mathcal{L}_b) = 0</math>  <math>\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{v}}_b) = \pi/2</math></p> <p>with  <math>\hat{\mathbf{v}}_b = \hat{\mathbf{d}}_{\mathcal{K}_b} \times \hat{\mathbf{d}}_{\mathcal{L}_b}</math></p>

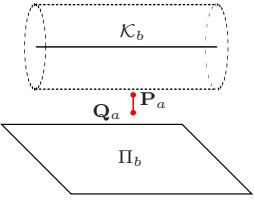
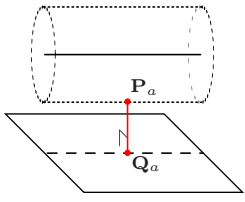
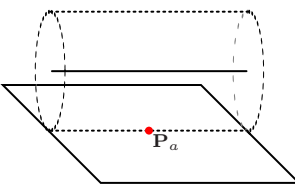
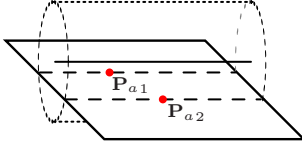
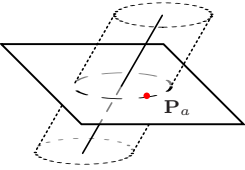
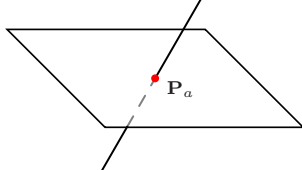
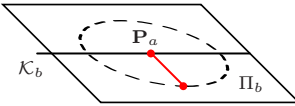
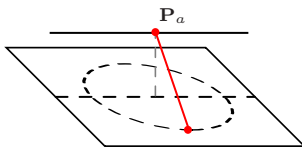
with  $\mathbf{P}_b \subset \mathcal{K}_b$      $\mathbf{Q}_b = \text{proj}_{\mathcal{L}_b} \mathbf{P}_b$      $\hat{\mathbf{u}}_a = \overrightarrow{\mathbf{P}_a \mathbf{Q}_a} / |\mathbf{P}_a \mathbf{Q}_a|$      $\hat{\mathbf{u}}_b = \overrightarrow{\mathbf{P}_b \mathbf{Q}_b} / |\mathbf{P}_b \mathbf{Q}_b|$

TABLE V  
POINT-POINT DISTANCE AND POINT-PLANE COINCIDENCE CONSTRAINTS  $d(\mathbf{P}_a, \mathbf{P}_b) = p$ ,  $d(\mathbf{Q}_a, \Pi_b) = 0$

$d_b - d_a > p$	 <p><b>Solutions: 0 (Incompatible)</b></p>	$(d_b = p) \wedge (d_a = 0)$	 <p><b>Solutions: 1</b>  <math>d(\mathbf{P}_a, \mathbf{Q}_b) = 0</math></p>
$(d_b - d_a = p) \wedge (d_a \neq 0) \wedge (d_b \neq 0)$	 <p><b>Solutions: 1</b>  <math>d(\mathbf{P}_a, \mathbf{R}_b) = 0</math>  <math>d(\mathbf{Q}_a, \mathbf{Q}_b) = 0</math></p> <p>with  <math>\mathbf{R}_b = \mathbf{P}_b + p \hat{\mathbf{u}}_b</math>  <math>\mathbf{Q}_b = \text{proj}_{\Pi_b} \mathbf{P}_b</math>  <math>\hat{\mathbf{u}}_b = \overrightarrow{\mathbf{P}_b \mathbf{Q}_b} /  \mathbf{P}_b \mathbf{Q}_b </math></p>	$(d_b < p) \wedge (d_a = 0)$	 <p><b>Solutions: 1</b>  <math>d(\mathbf{P}_a, \mathcal{L}_b) = k</math>  <math>d(\mathbf{P}_a, \Pi_b) = 0</math></p> <p>with  <math>\mathcal{L}_b(\hat{\mathbf{n}}_{\Pi_b}, \mathbf{P}_b)</math>  <math>k = (p^2 - d_b^2)^{1/2}</math></p>
$(d_a \neq 0) \wedge (d_b = 0) \wedge (p = 0)$	 <p><b>Solutions: 1</b>  <math>d(\mathbf{P}_a, \mathbf{P}_b) = 0</math>  <math>d(\mathbf{Q}_a, \Pi_b) = 0</math>  <math>\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{n}}_{\Pi_b}) = \pi/2</math></p>	$(d_a > d_b) \wedge (p = 0)$	 <p><b>Solutions: 1</b>  <math>d(\mathbf{P}_a, \mathbf{P}_b) = 0</math>  <math>d(\mathbf{Q}_a, \Pi_b) = 0</math>  <math>\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = \alpha</math></p> <p>with  <math>\alpha = \cos^{-1}(d_b/d_a)</math>  <math>\mathbf{Q}_b = \text{proj}_{\Pi_b} \mathbf{P}_b</math></p>

with  $\hat{\mathbf{u}}_a = \overrightarrow{\mathbf{P}_a \mathbf{Q}_a} / |\mathbf{P}_a \mathbf{Q}_a|$      $\hat{\mathbf{u}}_b = \overrightarrow{\mathbf{P}_b \mathbf{Q}_b} / |\mathbf{P}_b \mathbf{Q}_b|$

TABLE VI  
POINT-LINE DISTANCE AND POINT-PLANE COINCIDENCE CONSTRAINTS  $d(\mathbf{P}_a, \mathcal{K}_b) = p$ ,  $d(\mathbf{Q}_a, \Pi_b) = 0$

$d_b - d_a > p$	$(\mathcal{K}_b \parallel \Pi_b) \wedge (d_b - d_a = p) \wedge (d_a \neq 0) \wedge (d_b \neq 0)$
	<b>Solutions: 0 (Incompatible)</b>
	<b>Solutions: 1</b> $d(\mathbf{P}_a, \mathcal{L}_b) = 0$ $d(\mathbf{Q}_a, \mathcal{M}_b) = 0$ with $\mathbf{R}_b = \mathbf{P}_b + p\hat{\mathbf{u}}_b$
$(\mathcal{K}_b \parallel \Pi_b) \wedge (d_b = p) \wedge (d_a = 0)$	$(\mathcal{K}_b \parallel \Pi_b) \wedge (d_b < p) \wedge (d_a = 0)$
	<b>Solutions: 1</b> $d(\mathbf{P}_a, \mathcal{L}_b) = 0$ with $\mathbf{R}_b = \mathbf{Q}_b$
	<b>Solutions: 2</b> $d(\mathbf{P}_a, \mathcal{L}_b) = 0$ with $\mathbf{R}_b = \mathbf{Q}_b \pm k\hat{\mathbf{v}}_b$ $k = (p^2 - d_b^2)^{1/2}$ $\hat{\mathbf{v}}_b = \hat{\mathbf{d}}_{\mathcal{K}_b} \times \hat{\mathbf{n}}_{\Pi_b}$
$(\mathcal{K}_b \not\parallel \Pi_b) \wedge (d_a = 0) \wedge (d_b = 0) \wedge (p \neq 0)$	$(\mathcal{K}_b \not\parallel \Pi_b) \wedge (d_a = 0) \wedge (d_b = 0) \wedge (p = 0)$
	<b>Solutions: 1</b> $d(\mathbf{P}_a, \mathcal{K}_b) = p$ $d(\mathbf{P}_a, \Pi_b) = 0$
	<b>Solutions: 1</b> $d(\mathbf{P}_a, \mathbf{R}_b) = 0$ with $\mathbf{R}_b = \mathcal{K}_b \cap \Pi_b$
$(\mathcal{K}_b \parallel \Pi_b) \wedge (d_a \neq 0) \wedge (d_b = 0) \wedge (p = 0)$	$(\mathcal{K}_b \parallel \Pi_b) \wedge (d_a > d_b) \wedge (d_b \neq 0) \wedge (p = 0)$
	<b>Solutions: 1</b> $d(\mathbf{P}_a, \mathcal{K}_b) = 0$ $d(\mathbf{Q}_a, \Pi_b) = 0$ $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{n}}_{\Pi_b}) = \pi/2$
	<b>Solutions: 1</b> $d(\mathbf{P}_a, \mathcal{K}_b) = 0$ $d(\mathbf{Q}_a, \Pi_b) = 0$ $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = \alpha$ with $\alpha = \cos^{-1}(d_b/d_a)$

with  $\mathbf{P}_b \subset \mathcal{K}_b$      $\mathbf{Q}_b = \text{proj}_{\Pi_b} \mathbf{P}_b$      $\mathcal{L}_b(\hat{\mathbf{d}}_{\mathcal{K}_b}, \mathbf{R}_b)$      $\mathcal{M}_b(\hat{\mathbf{d}}_{\mathcal{K}_b}, \mathbf{Q}_b)$      $\hat{\mathbf{u}}_a = \overrightarrow{\mathbf{P}_a \mathbf{Q}_a} / |\mathbf{P}_a \mathbf{Q}_a|$      $\hat{\mathbf{u}}_b = \overrightarrow{\mathbf{P}_b \mathbf{Q}_b} / |\mathbf{P}_b \mathbf{Q}_b|$

TABLE VII  
TWO POINT-PLANE COINCIDENCE CONSTRAINTS  $d(\mathbf{P}_a, \Sigma_b) = 0, d(\mathbf{Q}_a, \Pi_b) = 0$

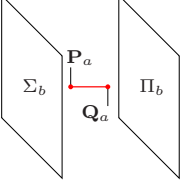
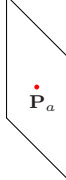
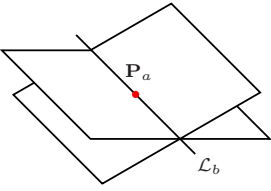

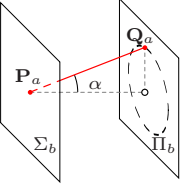
$d_b > d_a$	$(\Sigma_b \nparallel \Pi_b) \wedge (d_a = 0) \wedge (d_b = 0)$
	
<b>Solutions:</b> 0 (Incompatible)	<b>Solutions:</b> 1 $d(\mathbf{P}_a, \Sigma_b) = 0$
$(\Sigma_b \nparallel \Pi_b) \wedge (d_a = 0) \wedge (d_b = 0)$	$(\Sigma_b \parallel \Pi_b) \wedge (d_a \neq 0) \wedge (d_b = 0)$
	
<b>Solutions:</b> 1 $d(\mathbf{P}_a, \mathcal{L}_b) = 0$ with $\mathcal{L}_b = \Sigma_b \cap \Pi_b$	<b>Solutions:</b> 1 $d(\mathbf{P}_a, \Sigma_b) = 0$ $d(\mathbf{Q}_a, \Sigma_b) = 0$ $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{n}}_{\Sigma_b}) = \pi/2$
$(\Sigma_b \parallel \Pi_b) \wedge (d_a \geq d_b) \wedge (d_b \neq 0)$	
	
<b>Solutions:</b> 1 $d(\mathbf{P}_a, \Sigma_b) = 0$ $d(\mathbf{Q}_a, \Pi_b) = 0$ $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = \alpha$ with $\alpha = \cos^{-1}(d_b/d_a)$	
with $\mathbf{P}_b \subset \Sigma_b$ $\mathbf{Q}_b = \text{proj}_{\Pi_b} \mathbf{P}_b$ $\hat{\mathbf{u}}_a = \overrightarrow{\mathbf{P}_a \mathbf{Q}_a} /  \mathbf{P}_a \mathbf{Q}_a $ $\hat{\mathbf{u}}_b = \overrightarrow{\mathbf{P}_b \mathbf{Q}_b} /  \mathbf{P}_b \mathbf{Q}_b $	

TABLE VIII  
POINT-LINE AND LINE-POINT DISTANCE CONSTRAINTS  $d(\mathbf{P}_a, \mathcal{K}_b) = p, d(\mathbf{Q}_b, \mathcal{L}_a) = q$

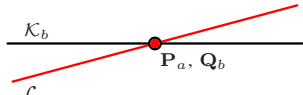
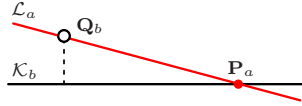
$(d_a = 0) \wedge (d_b = 0) \wedge (p = 0) \wedge (q = 0)$	$(d_b > 0) \wedge (d_a = 0) \wedge (p = 0) \wedge (q = 0)$
	
<b>Solutions:</b> 2 <b>First solution:</b> $d(\mathbf{P}_a, \mathbf{Q}_b) = 0$ <b>Second solution:</b> $d(\mathbf{P}_a, \mathcal{K}_b) = 0$ $\angle(\hat{\mathbf{d}}_{\mathcal{L}_a}, \hat{\mathbf{d}}_{\mathcal{K}_b}) = 0$	<b>Solutions:</b> 1 $d(\mathbf{P}_a, \mathcal{K}_b) = 0$ $d(\mathbf{Q}_b, \mathcal{L}_a) = 0$ $\angle(\hat{\mathbf{d}}_{\mathcal{L}_a}, \hat{\mathbf{v}}_b) = \pi/2$ with $\hat{\mathbf{v}}_b = \hat{\mathbf{d}}_{\mathcal{L}_b} \times \hat{\mathbf{u}}_b$
with $\mathbf{P}_b = \text{proj}_{\mathcal{K}_b} \mathbf{Q}_b$ $\hat{\mathbf{u}}_b = \overrightarrow{\mathbf{P}_b \mathbf{Q}_b} /  \mathbf{P}_b \mathbf{Q}_b $	

TABLE IX  
POINT-LINE DISTANCE AND PLANE-POINT COINCIDENCE CONSTRAINTS  $d(\mathbf{P}_a, \mathcal{K}_b) = p, d(\mathbf{Q}_b, \Pi_a) = 0$

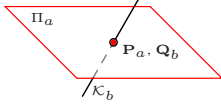
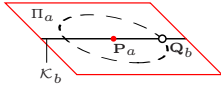
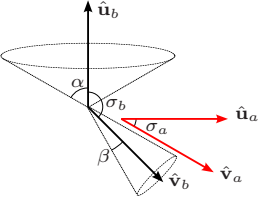
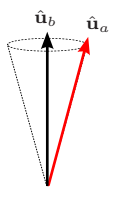
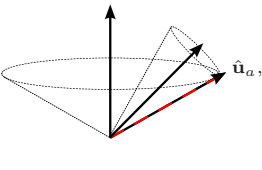
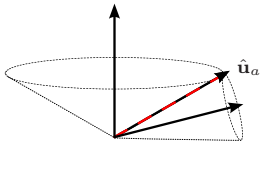
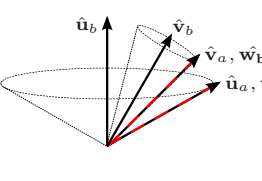
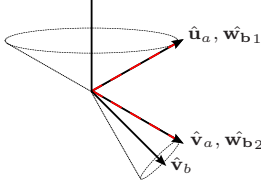
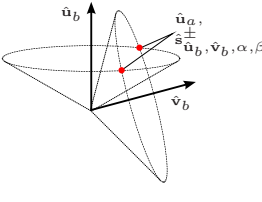
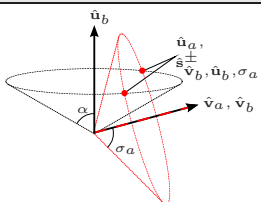
$(d_a = 0) \wedge (d_b = 0) \wedge (p = 0)$


<b>Solutions:</b> 2 <b>First solution:</b> $d(\mathbf{P}_a, \mathbf{Q}_b) = 0$ <b>Second solution:</b> $d(\mathbf{P}_a, \mathcal{K}_b) = 0$ $\angle(\hat{\mathbf{n}}_{\Pi_a}, \hat{\mathbf{d}}_{\mathcal{K}_b}) = \pi/2$

TABLE X  
TWO ANGLE CONSTRAINTS  $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = \alpha$ ,  $\angle(\hat{\mathbf{v}}_a, \hat{\mathbf{v}}_b) = \beta$

$(\sigma_a + \sigma_b <  \alpha - \beta ) \vee ( \sigma_a - \sigma_b  > \alpha + \beta)$ <b>Solutions: 0</b> (Incompatible) 	$(\sigma_a = \sigma_b = n\pi) \wedge (\alpha = \beta)$ <b>Solutions: 1</b> $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = \alpha$ 
$(\sigma_a + \sigma_b =  \alpha - \beta ) \wedge (\sigma_a = n\pi) \wedge (\sigma_b \neq n\pi)$ <b>Solutions: 1</b> $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{w}}_b) = 0$ with $\hat{\mathbf{w}}_b = \mathbf{R}(k_1 \hat{\mathbf{t}}_b, \alpha) \hat{\mathbf{u}}_b$ 	$( \sigma_a - \sigma_b  = \alpha + \beta \neq 0) \wedge (\sigma_a = n\pi) \wedge (\sigma_b \neq n\pi)$ <b>Solutions: 1</b> $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{w}}_b) = 0$ with $\hat{\mathbf{w}}_b = \mathbf{R}(-k_2 \hat{\mathbf{t}}_b, \alpha) \hat{\mathbf{u}}_b$ 
$(\sigma_a + \sigma_b =  \alpha - \beta ) \wedge (\sigma_a \neq n\pi) \wedge (\sigma_b \neq n\pi)$ <b>Solutions: 1</b> $\angle(\hat{\mathbf{u}}_m, \hat{\mathbf{w}}_{b1}) = 0$ $\angle(\hat{\mathbf{v}}_m, \hat{\mathbf{w}}_{b2}) = 0$ with $\hat{\mathbf{w}}_{b1} = \mathbf{R}(k_1 \hat{\mathbf{t}}_b, \alpha) \hat{\mathbf{u}}_b$ $\hat{\mathbf{w}}_{b2} = \mathbf{R}(k_1 \hat{\mathbf{t}}_b, \beta) \hat{\mathbf{v}}_b$ 	$( \sigma_a - \sigma_b  = \alpha + \beta \neq 0) \wedge (\sigma_a \neq n\pi) \wedge (\sigma_b \neq n\pi)$ <b>Solutions: 1</b> $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{w}}_{b1}) = 0$ $\angle(\hat{\mathbf{v}}_a, \hat{\mathbf{w}}_{b2}) = 0$ with $\hat{\mathbf{w}}_{b1} = \mathbf{R}(-k_2 \hat{\mathbf{t}}_b, \alpha) \hat{\mathbf{u}}_b$ $\hat{\mathbf{w}}_{b2} = \mathbf{R}(k_2 \hat{\mathbf{t}}_b, \beta) \hat{\mathbf{v}}_b$ 
$(\sigma_b >  \alpha - \beta ) \wedge (\sigma_b < \alpha + \beta) \wedge (\sigma_a = 0) \wedge (\sigma_b \neq n\pi)$ <b>Solutions: 2</b> $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{s}}_{\hat{\mathbf{u}}_b, \hat{\mathbf{v}}_b, \alpha, \beta}^{\pm}) = 0$ 	$(\sigma_a + \sigma_b > \alpha) \wedge ( \sigma_a - \sigma_b  < \alpha) \wedge (\sigma_a \neq n\pi) \wedge (\sigma_b \neq n\pi) \wedge (\alpha \neq 0) \wedge (\beta = 0)$ <b>Solutions: 2</b> $\angle(\hat{\mathbf{v}}_a, \hat{\mathbf{v}}_b) = 0$ $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{s}}_{\hat{\mathbf{v}}_b, \hat{\mathbf{u}}_b, \sigma_a, \alpha}^{\pm}) = 0$ 

$$\text{with } \hat{\mathbf{t}}_a = \hat{\mathbf{u}}_a \times \hat{\mathbf{v}}_a$$

$$\hat{\mathbf{t}}_b = \hat{\mathbf{u}}_b \times \hat{\mathbf{v}}_b$$

$$k_1 = \text{sgn}(\alpha - \beta)$$

$$k_2 = \text{sgn}(\sigma_a - \sigma_b)$$

## REFERENCES

- [1] N. Turro, O. Khatib, and E. Coste-Maniere, "Haptically augmented teleoperation," in *IEEE Int. Conf. Robot. Automat.*, Seoul, Korea, May 21-26 2001, pp. 386–392.
- [2] B. P. DeJong, E. L. Faulring, J. E. Colgate, and M. A. Peshkin, "Lessons learned from a novel teleoperation testbed," *Industrial Robot: An International Journal*, vol. 33, no. 3, pp. 187–193, 2006.
- [3] E. L. Faulring, K. M. Lynch, J. Colgate, and M. A. Peshkin, "Haptic display of constrained dynamic systems via admittance displays," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 101–111, February 2007.
- [4] C. Hoffmann and R. Joan-Arinyo, "A brief on constraint solving," *CAD&A*, vol. 2, no. 5, pp. 655–664, 2005.
- [5] I. Fudos and C. Hoffmann, "A graph-constructive approach to solving systems of geometric constraints," *ACM Transactions on Graphics*, vol. 16, no. 2, pp. 179–216, 1997.
- [6] J. Owen, "Algebraic solution for geometry from dimensional constraints," in *Symposium on Solid Modeling Foundations and CAD/CAM Applications*, R. Rossignac and J. Turner, Eds. Austin, TX: ACM Press, June 5-7 1991, pp. 397–407.
- [7] G. Kramer, *Solving Geometric Constraint Systems*. MIT Press, 1992.
- [8] B. Bruderlin, "Symbolic computer geometry for computer aided geometric design," in *Advances in Design and Manufacturing Systems*. Tempe, AZ: NSF conference, January 8-12 1990.
- [9] ———, "Using geometric rewrite rules for solving geometric problems symbolically," in *Theoretical Computer Science 116*. Elsevier Science Publishers B.V., 1993, pp. 291–303.
- [10] G. Nelson, "Juno, a constraint-based graphics system," in *SIGGRAPH*, San Francisco, July 22-26 1985, pp. 235–243.
- [11] H. Lamure and D. Michelucci, "Solving geometric constraints by homotopy," in *Third Symposium on Solid Modeling and Applications*, C. Hoffmann and J. Rossignac, Eds. Salt Lake City, Utah: ACM Press, May 17-19 1995, pp. 263–269.
- [12] K. Kondo, "Algebraic method for manipulation of dimensional relationships in geometric models," in *Computer-Aided Design*, March 1992, pp. 141–147.
- [13] A. Kumar and L. Yu, "Sequential constraint imposition for dimension-driven solid models," *Computer-Aided Design*, vol. 33, no. 6, pp. 475–486, 2001.
- [14] J. M. Porta, L. Ros, F. Thomas, and C. Torras, "A branch-and-prune solver for distance constraints," *IEEE Trans. Robot.*, vol. 21, no. 2, pp. 176–187, April 2005.
- [15] A. Rodríguez, L. Basañez, and E. Celaya, "Robot task specification and execution through relational positioning," in *IFAC Workshop on Intelligent Manufacturing Systems*, Alicante, Spain, May 23-25 2007.
- [16] E. Celaya, "Geometric reasoning for the determination of the position of objects linked by spatial relationships," Ph.D. dissertation, Universitat Politècnica de Catalunya, Barcelona, Spain, 1992.
- [17] ———, "LMF: A program for positioning objects using geometric relationships," in *VII International Conference on Applications of Artificial Intelligence in Engineering*, D. G. et al., Ed. Computational Mechanics and Elsevier Applied Science, 1992, pp. 873–881.
- [18] E. Nuño, A. Rodríguez, and L. Basañez, "Force reflecting teleoperation via ipv6 protocol with geometric constraints haptic guidance," in *Springer Trans. in Advanced Robotics – Advances in Telerobotics*, S. Verlag, Ed., 2007, vol. 31, ch. 26, pp. 445–458.
- [19] E. Nuño and L. Basañez, "Haptic guidance with force feedback to assist teleoperation systems via high speed networks," in *37th International Symposium on Robotics*, IFR, Ed. Munich, Germany: VDI Wissensforum IWB GmbH, May 2006, pp. 1–14.
- [20] J. Duffy, *Analysis of Mechanisms and Robot Manipulators*. London: Edward Arnold, 1980.