A Framework for the Simulation and Haptic Display of Dynamic Systems Subject to Holonomic Constraints

Adolfo Rodríguez, Luis Basañez, J. Edward Colgate, Eric L. Faulring

Abstract— **This paper presents a framework that enables an operator to haptically and visually interact with a simulated dynamic environment subject to virtual holonomic constraints. The framework combines a geometric constraint solver with a constrained dynamics simulation engine that controls an admittance-type haptic display. This system takes on relevant issues in the context of assisted teleoperated tasks, from providing an intuitive interface for creating and combining virtual constraints, to haptically displaying rigid motion constraints in simulated environments subject to desired inertial dynamics. Two experiments carried out using the Cobotic Hand Controller haptic display are presented.**

I. INTRODUCTION

In a teleoperated robot task, a robot executes the movements/actions commanded by an operator. The teleoperated execution of a task is justified because it is often not practical either to perform the task with an autonomous robot, or to perform it with a human operator for reasons as diverse as dangerous environments (exposure to radiation, space), physical separation of the execution site, as well as precision and scale issues (heavy-load manipulation, micro/nanometer-sized workspaces).

Oftentimes, the execution of a task can be decomposed into a series of movements that do not require using the six degrees of freedom (DOF) an object has in free space. Although operator skills are needed for the successful execution of a teleoperated task, maintaining the tool or the manipulated object inside a specific region of space can be both challenging and tiring (Rosenberg, 1993). Such regions can be described in terms of geometric constraints, that when satisfied define a submanifold of $SE(3)$ of allowed movements (Fig. 1). Haptic feedback can be used to assist the operator by restricting their movements to a submanifold of interest, lowering the mental burden needed to execute the task.

A teleoperation system that implements rigid motion constraints should then feature an intuitive interface for creating and changing constraint scenarios in real time, as well as a haptic device capable of displaying high-impedance constraints. To the authors' knowledge, such a system has not yet been reported in the literature. In what follows, existing approaches for constraint creation and display will be summarized, and the problem addressed by the present framework will be introduced.

A. Rodríguez and L. Basañez are with the Institute of Industrial and Control Engineering (IOC), Technical University of Catalonia (UPC) 08028 Barcelona, Spain. adolfo.rodriguez@ieee.org; luis.basanez@upc.edu.

J. Edward Colgate is with the Department of Mechanical Engineering, Northwestern University, Evanston IL 60208-3111 USA. colgate@northwestern.edu.

Eric L. Faulring is with Kinea Design, LLC, Evanston, IL 60201 USA. eric.faulring@ieee.org.

This work has been partially supported by CICYT projects DPI2005-00112 and DPI2007-63665, and the FPI predoctoral grant BES-2004-4070.

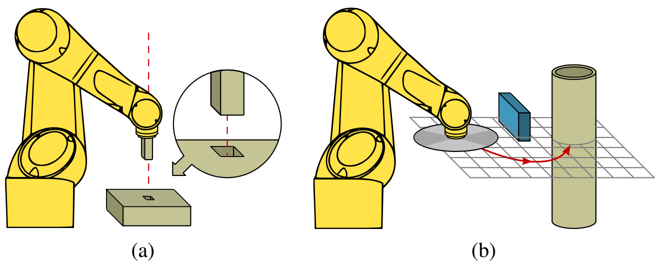


Fig. 1. Example constrained movement scenarios:
(a) The insertion of a square peg-in-hole only requires one DOF, translations along the hole's axis. This can be achieved by making the centerlines of the peg and hole coincide, and by additionally maintaining a lateral face of the peg parallel to the corresponding internal hole face.
(b) The cutting of a pipe with a circular saw takes place in a two-DOF flat subspace. This scenario is characterized by making the saw plane and the cutting plane coincide (there is an additional rotational DOF normal to the cutting plane which is of no use in this task).

A common disadvantage of existing approaches is the lack of an intuitive constraint creation interface. The creation of virtual constraints often requires knowledge of the underlying mathematical and software models, hence an experienced user. Furthermore, changing a constraint scenario (i.e., by adding, removing, or modifying constraints) is usually accomplished by manual reprogramming, and thus cannot be done online and interactively.

Turro et al. (2001) present a system that is able to generate forces with a haptic device to avoid virtual obstacles, to display repulsion forces when the slave robot approaches workspace singularities, and to restrict the movement of the haptic end-effector to curves and surfaces. The methodology proposed by Li et al. (2007) generates motion constraints for assisting teleoperated surgical tasks, where different movement restrictions that are meaningful to the application domain can be achieved from combinations of five primitive "Virtual Fixtures". De Schutter et al. (2007) describe a constraint-based methodology for the specification of complex sensor-based robot tasks, and its application to human-robot comanipulation scenarios is discussed. However, none of these systems have an intuitive constraint creation interface.

Haptic feedback quality largely depends on the characteristics of the haptic device and the corresponding control algorithm. Impedance displays (Salisbury et al., 1991; Massie and Salisbury, 1994; Force Dimension, 2008) tend to be backdrivable mechanisms with low inertia and friction, so they excel at displaying low inertia/damping environments, but have difficulty at stably rendering high-stiffness constraints.

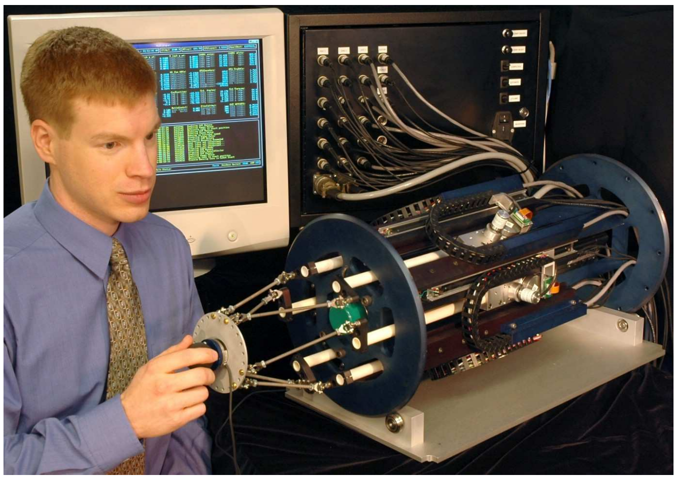


Fig. 2. The Cobotic Hand Controller.

Common control algorithms for impedance displays, like the god-object tracker (Zilles and Salisbury, 1995) and the virtual proxy (Ruspini and Khatib, 2001), leave the dynamics of the unconstrained directions unchanged and generate forces in the constrained directions based on the difference between the actual and desired positions of the end-effector. Alternatively, admittance displays (Taylor et al., 1999; Van der Linde et al., 2002) tend to be non-backdrivable mechanisms with high inertia and friction, so they are capable of displaying high stiffness constraints, but have difficulty displaying low inertia/damping environments. Control algorithms for admittance displays tend to be more complex than their impedance counterparts. Yun and Sarkar (1998) and Liu and Li (2002) use a dynamic model of the constrained system to compute the desired dynamics along the unconstrained directions. The pseudo-admittance bilateral teleoperation control scheme of Abbott and Okamura (2007) has the advantage that is amenable to both impedance and admittance haptic devices and can translationally constrain the device end-effector to continuous curves and surfaces. It relies on an external module for describing the geometric constraints, and does not handle rotational constraints.

The Cobotic Hand Controller (Faulring et al., 2004, 2006) is a six-DOF admittance haptic display that uses infinitely variable transmissions to relate joint motions (Fig. 2). These transmissions permit varying backdrivability, hence the ability to render both high and low impedance environments. Constraints are defined by the transmission ratios, and unlike most impedance displays, constraint forces are generated by the mechanical structure of the device rather than by its actuators, so high constraint forces can be sustained for long periods of time with little or no power consumption. Faulring et al. (2007) present an admittance control architecture for the simulation and haptic rendering of dynamic systems subject to holonomic and nonholonomic constraints. Tests have been performed on the Cobotic Hand Controller under a variety of scenarios, and although the methodology is quite general, the mathematical construction of the constraints that restrict the manipulated object requires significant effort.

The goal of the present work is to assist an operator to perform constrained movements while executing a teleoperated task. Its scope is that of providing the *local* site of a teleoperation system with means for the real time creation and haptic display of rigid motion constraints. These locally generated aids have the benefit of not depending on information sent from the *remote* site (hence are unaffected by the time-delays of the communication channel), and complement the haptic feedback that bilateral teleoperation schemes provide (Anderson and Spong, 1989; Chopra and Spong, 2006; Nuño et al., 2008).

The proposed system is based on the admittance control architecture of Faulring et al. (2007) and the geometric constraint solver of Rodríguez et al. (2008a), that permits the interactive definition of a wide range of holonomic constraints. Visual cues integrated with the geometric constraint solver interface complement the haptic feedback by displaying the simulated objects and the constraints they are subject to. Preliminary results have been published in Rodríguez et al. (2008b).

Geometric constraint solvers are used to find the map between constraint sets and solution submanifolds. Depending on the problem, this map may have multiple solutions or none at all, and is in general not injective, that is, there may exist multiple constraint sets associated to the same solution.

When designing a geometric constraint solver, a compromise must be made between completeness/generality and computational efficiency. Solvers oriented to CAD applications (Hoffmann, 2005), and mechanism or molecular modelling focus more on completeness/generality (Kramer, 1992; Porta et al., 2005), because solution computation seldom has tight time restrictions. Conversely, the constraint solver featured in this work has been designed with computational efficiency in mind, so that solutions can be computed and updated at high refresh rates. This allows dealing with situations such as updating the positions of moving obstacles during the execution of sensor-based tasks. Additionally, the ability to solve underconstrained problems is required, since in teleoperated tasks it is often desireable to restrict only partially the movements of an object, and still allow an operator to guide it using the remaining DOF.

The paper is laid out as follows: Section II gives a general overview of the proposed system; Section III describes the geometric constraint solver; Section IV describes the methodology used for simulating constrained dynamic systems and how it has been interfaced with the solver; Section V presents experiments performed using the Cobotic Hand Controller haptic display; Section VI comments on issues related to the implementation; and finally Section VII includes the conclusions.

II. System overview

The main components of the system, and the flow of signals between them are shown in Fig. 3. By means of an *intuitive user interface* (Fig. 4), the operator loads a virtual environment and interactively creates geometric constraints between its objects. He/she then selects one of the constrained objects to which the virtual tool of the haptic device $\mathbf{x}$ is attached and starts the simulation.

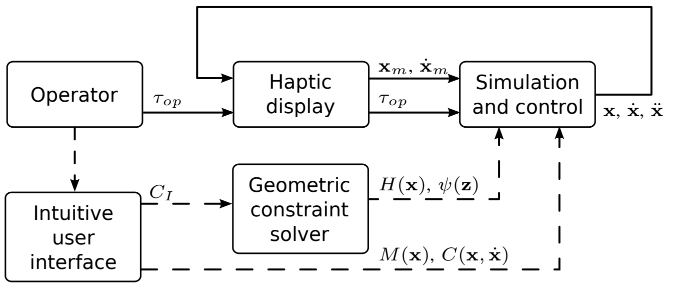

Operator
τ_{op}
Haptic display
$\mathbf{x}_m$, $\dot{\mathbf{x}}_m$
τ_{op}
Simulation and control
$\mathbf{x}$, $\dot{\mathbf{x}}$, $\ddot{\mathbf{x}}$
Intuitive user interface
C_I
Geometric constraint solver
$H(\mathbf{x})$, $\psi(\mathbf{z})$
$M(\mathbf{x})$, $C(\mathbf{x}, \dot{\mathbf{x}})$

Fig. 3. Flow of signals between the main components of the system. Dashed lines represent signals that are updated only when initializing the simulation environment. Solid lines represent signals that are updated at high-frequency during a simulation run.
The operator first sets the dynamic properties $M(\mathbf{x})$, $C(\mathbf{x}, \dot{\mathbf{x}})$ of the virtual object and the input geometric constraints C_I it is subject to. Then, once the simulation starts, he/she interacts with the virtual environment by exerting forces and torques τ_{op} on the end-effector of the haptic display. The measured state of the end-effector is represented by $\mathbf{x}_m$, $\dot{\mathbf{x}}_m$.

To initialize the simulation, the *geometric constraint solver* (Section III) takes the input constraints C_I, and computes a description of the associated constraint scenario $H(\mathbf{x})$, $\psi(\mathbf{z})$ that is compatible with the simulation algorithm. Then, the *simulation and control* module (Section IV) is set with the above computed constraints and the dynamic properties $M(\mathbf{x})$, $C(\mathbf{x}, \dot{\mathbf{x}})$ of the selected object[1].

Once the simulation is running, the operator feels the dynamic properties of the virtual object (and not those of the haptic device) and is allowed to move only along the unconstrained directions of the current constraint scenario. Not shown in Fig. 3, the positions of the objects in the graphical rendering of the virtual environment are updated in real time from the current state of the haptic end-effector. Extension 1 shows a real time example of the system in operation.

Compared to the previous work of Faulring et al. (2007), where constraint scenarios had to be individually hardcoded, the *simulation and control* module is now capable of switching at run-time between constraint scenarios that an operator creates and modifies by means of the *geometric constraint solver*. To this end, the constraint solving methodology of Rodríguez et al. (2008a) has been extended to compute as part of its solution process a description of the constraint scenario that is compatible with the simulation algorithm. In Section IV-B it will be shown that this description consists of an algebraic representation of the holonomic constraints $H(\mathbf{x})$ and a parametric representation of the solution submanifold $\psi(\mathbf{z})$, along with their first two derivatives.

The combination of the different components shown in Fig. 3 results in a system that can assist an operator to perform constrained movements while executing a teleoperated task.

III. PMF Geometric Constraint Solver

Positioning Mobile with respect to Fixed (PMF), is a geometric constraint solver that addresses the problem of

[1]The current implementation does *not* automatically compute dynamic properties from geometric and material data. They must be explicitly specified. Note also that $M(\mathbf{x})$ and $C(\mathbf{x}, \dot{\mathbf{x}})$ are in general configuration-dependent, so complex dynamic behaviors can be simulated (e.g., an articulated robot arm), although currently they have to be hardcoded.

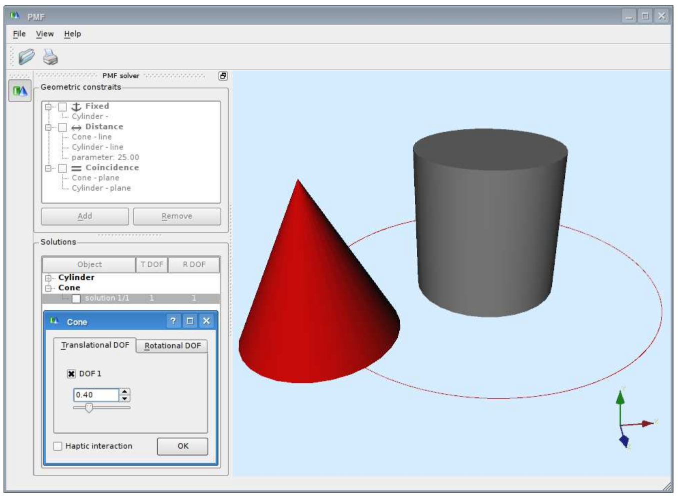


Fig. 4. Screenshot of the user interface.
The left side displays a control panel that permits the interactive definition of geometric constraints, shows information regarding the solutions of the current problem, and allows moving the selected object along its unconstrained directions by means of a mouse/keyboard or haptic display.
The right side displays a graphical rendering of the virtual environment. During constraint creation, it highlights the selected geometric elements, and during the editing of solutions, it shows the translational submanifold to which the active object is being constrained (the circular curve in the above example).

finding all possible configurations of a mobile object that satisfy a set of geometric constraints defined between the elements of the object and the elements of its surroundings, which are considered fixed. PMF accepts as input constraints distance (d) and angle ($\angle$) relations between points, lines, and planes. Geometric elements are usually anchored to boundary (vertex, edge, face) or reference (revolution axis, symmetry plane) features of the parent object. The actual shape of an object is used by the solver only for selecting the elements involved in a constraint and for visualization purposes. Object shape plays a relevant role in the handling of collisions and interferences with other objects (which are given by inequality constraints), a topic that is left to other application-specific modules. For instance, on a virtual environment, collision detection algorithms can be used to detect contact situations and enforce nonpenetration constraints (Section IV-C.3), while on a teleoperation system, haptic feedback from the remote site can be used directly to convey contact information.

When designing the solver, a strong emphasis was put on the efficient (real time) solution of constraint scenarios that are common to teleoperated tasks. These scenarios correspond to problems whose solution can be pictured qualitatively by the operator, and often involve constraints between simple geometric primitives like points, lines, planes, spheres, and cylinders.

The adopted notation for representing geometric entities throughout this presentation is: uppercase bold letters for points ($\mathbf{P}$, $\mathbf{Q}$), uppercase calligraphic letters for lines ($\mathcal{K}$, $\mathcal{L}$), uppercase Greek letters for planes (Π, Σ), and lowercase bold letters with a hat for unit vectors ($\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$). Vectors $\hat{\mathbf{d}}$ and $\hat{\mathbf{n}}$ represent line directions and plane normals, respectively. Also, the trigonometric functions $\sin\alpha$ and $\cos\alpha$ are abbreviated as s_α and c_α, respectively.

The solver exploits the fact that the rotational component of a set of input geometric constraints can often be separated from the translational one and solved independently. By means of logic reasoning and constraint rewriting, the solver is able to map a broad family of input problems to a few rotational and translational scenarios with known closed-form solution. The solution process consists of three main steps: input constraint decomposition, constraint combination, and solution synthesis.

A. *Solution process*

In the *input constraint decomposition* step, an input constraint set C_I is transformed into an equivalent set of pure rotational C_R and translational C_T constraints—called fundamental constraints—which contains fewer constraint types and is easier to work with. There are three fundamental translational constraints, which express the distance between a point and another geometric element (point, line, or plane):

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

and one fundamental rotational constraint:

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

Subindices "a" and "b" represent the object to which a geometric element belongs. One object is always fixed, while the other is mobile. Distance between two lines is the only input constraints that cannot be decomposed into the above fundamental constraints, although if the lines are also constrained to be parallel, the decomposition becomes possible.

In the *constraint combination* step a set of rules define a constraint rewriting engine that 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 that ill-defined cases are labeled as unsolvable; redundancies, removed; pairs of constraints, substituted with a single and equally restrictive constraint (hence the compactness); and rotational constraints implicitly defined by pairs of translational ones, identified (hence the explicitness).

The constraint rewriting rules are obtained by applying to each pair of constraint types to be tested the following process:

- Find the compatibility conditions that enable the two constraints to be satisfied simultaneously. They will depend on up to four distance or angle parameters.
- Create a rule that labels the problem as unsolvable if the compatibility condition is not satisfied.
- Consider the compatibility conditions in its general form, as well as at the boundary 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.

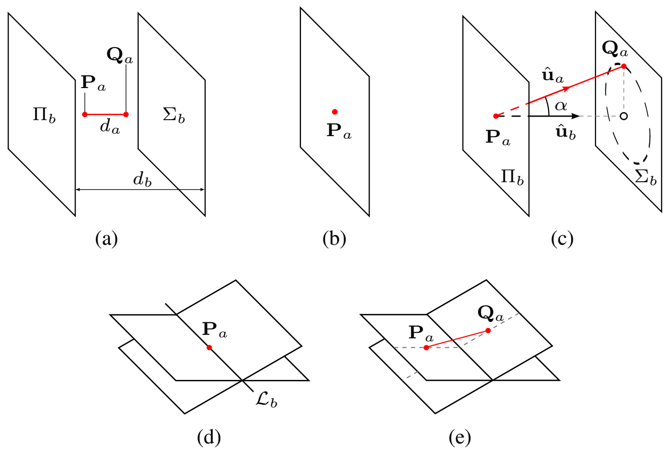


Fig. 5. Combination rules for two *point-plane* coincidence constraints $d(\mathbf{P}_a, \Pi_b) = 0$, $d(\mathbf{Q}_a, \Sigma_b) = 0$. The compatibility condition is $d_a \geq d_b$.
(a) $(d_a < d_b)$: Incompatible constraints. Label problem as unsolvable.
(b) $(\Pi_b \parallel \Sigma_b)$ and $(d_a = d_b = 0)$: Redundant constraints – remove one.
(c) $(\Pi_b \parallel \Sigma_b)$ and $(d_a \neq 0)$ and $(d_a \geq d_b \geq 0)$: Add to C_R the implicit rotation $\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = \alpha$, where $\alpha = \cos^{-1}(d_b/d_a)$.
(d) $(\Pi_b \nparallel \Sigma_b)$ and $(d_a = 0)$: Substitute both constraints with $d(\mathbf{P}_a, \mathcal{L}_b) = 0$, where $\mathcal{L}_b = \Pi_b \cap \Sigma_b$.
(e) $(\Pi_b \nparallel \Sigma_b)$ and $(d_a > 0)$: Leave constraints unchanged.

This approach has the limitation that no rules are extracted for cases in which the simultaneous satisfaction of two constraints cannot be expressed in terms of a single fundamental constraint, or when more than two constraints need to be simultaneously considered to extract them.

Table I lists the compatibility conditions for all combination scenarios handled by the solver. For brevity, the complete set of rules has been omitted from this paper, although they are provided in Rodríguez et al. (2007). Fig. 5 details the particular example of two *point-plane* coincidence constraints.

Finally, the *solution synthesis* step computes a transformation that positions the mobile object in a submanifold of $SE(3)$ that simultaneously satisfies *all* the imposed geometric constraints. The separation of rotational and translational fundamental constraints leads to a description of the submanifold that is given in terms of its rotational and translational components, R and T, respectively.

First, the rotational component R is solved using only the constraints in C_R, where R maps the initial orientation of the mobile object to a submanifold of the three-dimensional space of rotations that satisfies all the rotational constraints. Then, from a configuration that already satisfies R, the translational component T is solved using the constraints in C_T, where T maps the translation associated to an R-satisfying configuration of the mobile object to a submanifold of the three-dimensional space of translations that satisfies all the translational constraints. The dimension of the above submanifolds correspond to the number of DOF each solution component has.

The solution components R and T can be represented in the form of a parameterized rigid transformation—a homogeneous matrix, for example—that depends on as many parameters as available DOF, so that a sweep across the allowed parameters values will span the entire solution submanifold. Particular

TABLE I

COMPATIBILITY CONDITIONS FOR CONSTRAINT COMBINATION

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

Where d_a (d_b) represents the distance between the two elements belonging to object "a" ("b"). The same applies to σ_a (σ_b) but with angles instead of distances.

solutions are obtained by instantiating these parameters. The solution to the particular case of a well-constrained problem, which has no DOF is a constant rigid transformation. The specifics on how to obtain R and T from fundamental constraints can be found in Rodríguez et al. (2007).

When a solution submanifold has nonzero dimension there should exist an interface that provides a way to traverse it. The previous implementation of PMF had a user interface that allowed the editing of the DOF parameters only by means of the keyboard or mouse. This feature is useful for visualizing the effect of varying a single parameter, but is not satisfactory if the interest is in editing multiple parameters simultaneously. The next Section will explain how to achieve this by means of haptic interaction, and in the context of a dynamic simulation.

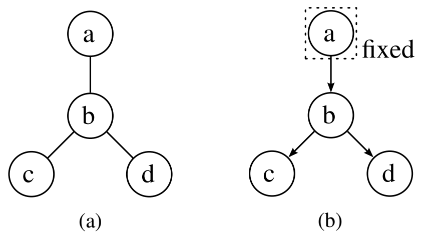


Fig. 6. Graph representation of a problem with multiple constrained objects. (a) Undirected graph where the the nodes represent objects and the arcs, constraints.
(b) Equivalent directed graph. Object "a" is fixed to an absolute reference frame, and since there are no cycles in the graph, the problem can be sequentially solved using a propagation approach. First, object "b" is positioned with respect to "a", and then objects "c" and "d" are positioned with respect to "b". Whenever object "b" moves along its degrees of freedom, the solutions of "c" and "d" are updated so that all constraints remain satisfied.

B. Positioning multiple objects

Oftentimes it is of interest to position not only one, but multiple objects at the same time. Such problems can be modeled as an undirected graph, where the the nodes represent objects and the arcs, constraints. This representation has the advantage that it displays in a straightforward manner the relations between the constrained objects, so topological features like cyclic dependencies and open chains can be easily detected. Graph analysis techniques can be then used to identify simple and solvable subproblems whose solutions can be combined while maintaining compatibility with the global problem. If at least one of the objects is fixed to an absolute reference frame, a propagation method can be used to sequentially solve each object and to direct the graph edges (Freeman-Benson et al., 1990; Latham and Middleditch, 1996), decomposing one complex problem into multiple simpler ones that can be solved by PMF, as shown in Fig. 6. One limitation of the solution scheme at its current state is that cyclic dependencies in the graph cannot in general be solved by PMF.

C. Example problems

Constraint combination is the core step in the solution of problems containing multiple constraints. The constraint rewriting process will be exemplified by means of constraint scenarios involving the objects depicted in Figs. 7a and 7b, and it will be shown how the problem geometry affects the number of solutions and the dimensionality of the resulting solution submanifolds. Three scenarios involving two *point-line* coincidence constraints will be considered. All three cases share a constraint in common: $d(\mathbf{P}_a, \mathcal{L}_b) = 0$.

1) $\{d(\mathbf{P}_a, \mathcal{L}_b) = 0,\ d(\mathbf{Q}_a, \mathcal{M}_b) = 0\}$: Let $d_a = \lvert\overrightarrow{\mathbf{P}_a\mathbf{Q}_a}\rvert$ represent the distance between the two selected points of object a, and d_b the distance between the two selected lines of object b such that $\overrightarrow{\mathbf{P}_a\mathbf{Q}_a} = \hat{\mathbf{u}}_a d_a$ and $\overrightarrow{\mathbf{P}_b\mathbf{Q}_b} = \hat{\mathbf{u}}_b d_b$. Vectors $\hat{\mathbf{u}}_a$ and $\hat{\mathbf{u}}_b$ have unit length and points $\mathbf{P}_b \in \mathcal{L}_b$ and $\mathbf{Q}_b \in \mathcal{M}_b$ are such that the distance between the two lines is minimal.

If $d_a = d_b$, then the two *point-line* constraints can be rewritten as two *point-point* coincidences:

$$\{d(\mathbf{P}_a, \mathbf{P}_b) = 0,\ d(\mathbf{Q}_a, \mathbf{Q}_b) = 0\}\,. \qquad (1)$$

Furthermore, (1) implies a parallelism rotational constraint, so it can be restated as

$$\{d(\mathbf{P}_a, \mathbf{P}_b) = 0,\ \angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = 0\}\,. \qquad (2)$$

The solution associated to these constraints considering object b fixed is depicted in Fig. 7c, and has only one rotational DOF. Notice that the two original constraits, which are purely translational, not only fully restrict the position of the mobile object a, but also implicitly constrain two of its three rotational DOF.

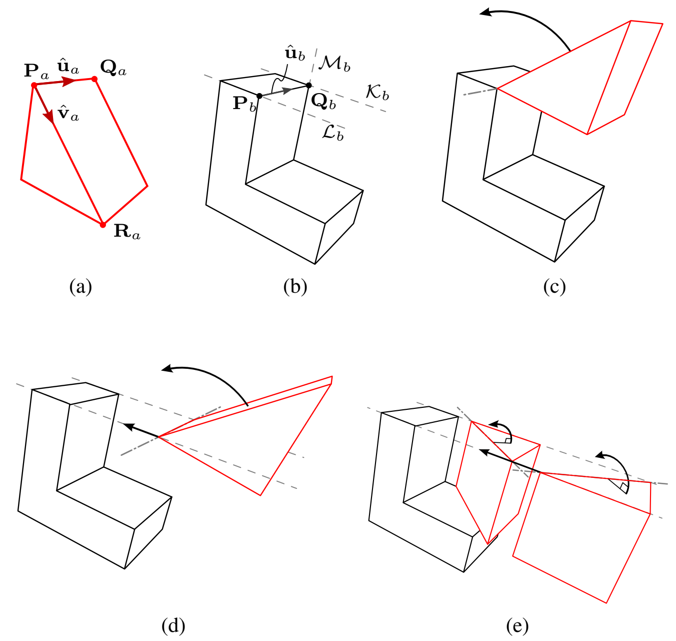


Fig. 7. Three different solutions for the simultaneous satisfaction of two *point-line* coincidence constraints between a mobile (a) and a fixed (b) object.
(c) $\{d(\mathbf{P}_a, \mathcal{L}_b) = 0,\ d(\mathbf{Q}_a, \mathcal{M}_b) = 0\}$: One rotational DOF.
(d) $\{d(\mathbf{P}_a, \mathcal{L}_b) = 0,\ d(\mathbf{Q}_a, \mathcal{K}_b) = 0\}$: One rotational and one translational DOF.
(e) $\{d(\mathbf{P}_a, \mathcal{L}_b) = 0,\ d(\mathbf{R}_a, \mathcal{K}_b) = 0\}$: Two solutions with one rotational and one translational DOF each.

2) $\{d(\mathbf{P}_a, \mathcal{L}_b) = 0,\ d(\mathbf{Q}_a, \mathcal{K}_b) = 0\}$*:* This scenario differs from the previous one in that line $\mathcal{K}_b$ is parallel to line $\mathcal{L}_b$. Preserving the same notation as before, if $d_a = d_b$, the two *point-line* constraints imply a parallelism constraint yielding

$$\{d(\mathbf{P}_a, \mathcal{L}_b) = 0,\ \angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = 0\}\,. \tag{3}$$

In this case, object a not only has one rotational DOF, but also one translational DOF along the direction of $\mathcal{L}_b$, as shown in Fig. 7d.

3) $\{d(\mathbf{P}_a, \mathcal{L}_b) = 0,\ d(\mathbf{R}_a, \mathcal{K}_b) = 0\}$*:* In this last example $d_a > d_b$, with $d_a = |\overrightarrow{\mathbf{P}_a\mathbf{R}_a}|$ and $\overrightarrow{\mathbf{P}_a\mathbf{R}_a} = \hat{\mathbf{v}}_a d_a$. The two *point-line* constraints imply a parallelism constraint with two possible alternatives

$$\{d(\mathbf{P}_a, \mathcal{L}_b) = 0,\ \angle(\hat{\mathbf{v}}_a, \hat{\mathbf{v}}_b) = 0\}\,, \tag{4}$$

where

$$\hat{\mathbf{v}}_b = \overrightarrow{\mathbf{P}_b\mathbf{R}_b}/|\mathbf{P}_b\mathbf{R}_b|$$
$$\mathbf{R}_b = \mathbf{Q}_b \pm \hat{\mathbf{d}}_{\mathcal{L}_b}\sqrt{d_a^2 - d_b^2}.$$

The two possible solutions for object a have one rotational and one rotational DOF each, and are shown in Fig. 7e.

If for this, or any of the above scenarios d_a and d_b are such that $d_a < d_b$, the compatibility condition is not satisfied (third row from the top of Table I), and the problem has no solution.

IV. Simulation of Constrained Dynamic Systems

A. Dynamic Model and Simulation Algorithm

The constrained Euler-Lagrange dynamic equations of the simulated mechanical system have the form

$$M(\mathbf{x})\ddot{\mathbf{x}} + C(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} = \tau + A(\mathbf{x})^T\lambda \tag{5}$$
$$A(\mathbf{x})\dot{\mathbf{x}} = 0. \tag{6}$$

Vector $\mathbf{x}$ is a $\mathbb{R}^6$ representation of the configuration of the haptic display end-effector in $SE(3)$. Generally speaking, the dimension and parameterization of the space in which the dynamics simulation takes place need not coincide with that of $\mathbf{x}$ (e.g., a six-DOF master device controlling a virtual tool in three-DOF planar space), but for the sake of simplicity they will be assumed equal in the present analysis. Section VI-A discusses how task coordinates can be transformed into haptic display coordinates when the dimensions of the two workspaces are different. A more general and detailed description of the simulation scheme used in this paper can be found in Faulring et al. (2007).

The left side of (5) describes the simulated inertial, centrifugal, and Coriolis forces, while $A(\mathbf{x})^T\lambda$ accounts for the forces due to the holonomic and non-holonomic constraints, and τ represents all other forces that may be present (i.e., operator forces along with simulated gravity, springs and dampers). The rows of $A(\mathbf{x})$—also called the Pfaffian constraint matrix—point in the constrained directions, and the Lagrange multipliers vector λ determines the magnitude of the constraint forces. Constraints in (6) are defined in the velocity domain; so holonomic constraints, which are more naturally defined in the position domain and satisfy $H(\mathbf{x}) = 0$, must be differentiated once to be included in $A(\mathbf{x})$. In the absence of non-holonomic constraints $A(\mathbf{x})$ becomes

$$A(\mathbf{x}) = \partial H/\partial \mathbf{x}. \tag{7}$$

For an admittance-type haptic display, the forces applied by the operator (contained in τ) are the input to the dynamic simulation with current state $(\mathbf{x},\ \dot{\mathbf{x}})$. It is necessary to solve (5) for accelerations and integrate to obtain the new desired state. For this, (5) and the derivative of (6) are combined to obtain the values of the Lagrange multipliers λ, which are then substituted in the dynamic equation to obtain an expression for the acceleration that is independent of λ:

$$\ddot{\mathbf{x}} = -\tilde{A}\dot{A}\dot{\mathbf{x}} + M^{-1}P_u(\tau - C\dot{\mathbf{x}}) \tag{8}$$

where

$$\tilde{A} = M^{-1}A^T(AM^{-1}A^T)^{-1} \tag{9}$$
$$P_u = I_{n\times n} - A^T(AM^{-1}A^T)^{-1}AM^{-1}. \tag{10}$$

The simulation is propagated forward by computing the new state of the system. A parametric approach is used to integrate $\ddot{\mathbf{x}}$, so that the virtual tool remains on the solution submanifold. Let $\mathbf{x} = \psi(\mathbf{z})$ be a parameterization of the solution submanifold, where $\mathbf{z}$ is the parametric coordinates vector. Differentiating $\psi(\mathbf{z})$ twice

$$\ddot{\mathbf{x}} = \frac{\partial\psi}{\partial\mathbf{z}}\ddot{\mathbf{z}} + \dot{\mathbf{z}}^T\frac{\partial^2\psi}{\partial\mathbf{z}^2}\dot{\mathbf{z}} \tag{11}$$

and solving for $\ddot{\mathbf{z}}$

$$\ddot{\mathbf{z}} = \left(\frac{\partial \psi}{\partial \mathbf{z}}\right)^{\dagger}\left(\ddot{\mathbf{x}} - \dot{\mathbf{z}}^T \frac{\partial^2 \psi}{\partial \mathbf{z}^2}\dot{\mathbf{z}}\right). \tag{12}$$

Here $(\partial\psi/\partial\mathbf{z})^{\dagger}$ is the Moore-Penrose pseudo-inverse. Matrix $\partial\psi/\partial\mathbf{z}$ maps parameter space velocities onto task-space velocities, and has dimension $6 \times n$, where $n \leq 6$ is the dimension of the parameter space. For an arbitrary motion command, (12) finds a least-squares "closest" solution for $\ddot{\mathbf{z}}$ and does not respect energy conservation. However, since the desired motion computed in (8) satisfies the Euler-Lagrange equations, $(\partial\psi/\partial\mathbf{z})^{\dagger}$ is simply performing a change of coordinates (Faulring, 2005). If $\partial\psi/\partial\mathbf{z}$ has full column rank, then it is left-invertible and $(\partial\psi/\partial\mathbf{z})^{\dagger}$ can be computed as

$$\frac{\partial \psi}{\partial \mathbf{z}}^{\dagger} = \left[\left(\frac{\partial \psi}{\partial \mathbf{z}}\right)^T \frac{\partial \psi}{\partial \mathbf{z}}\right]^{-1}\left(\frac{\partial \psi}{\partial \mathbf{z}}\right)^T. \tag{13}$$

Numeric integration of $\ddot{\mathbf{z}}$ yields the parametric position and velocity of a point belonging to the solution submanifold, which can be then mapped to task space via $\mathbf{x} = \psi(\mathbf{z})$ and $\dot{\mathbf{x}} = (\partial\psi/\partial\mathbf{z})\dot{\mathbf{z}}$, respectively.

Control errors can cause the measured state of the end-effector $(\mathbf{x}_m, \dot{\mathbf{x}}_m)$ to differ from the computed reference command (although they should be close), so an additional feedback term, implemented as a PID controller is used to cancel the tracking error between the reference and measured state of the end-effector.

B. Geometric Constraint Solver Interface

In order to incorporate the solutions from the geometric constraint solver into the above simulation framework, the geometric constraint problem and its solution must be described in a way that is compatible with it. This description consists of an algebraic representation of the holonomic constraints $H(\mathbf{x})$, and a parametric representation of the solution submanifold $\psi(\mathbf{z})$, with the additional requirement that they should be twice-differentiable, as can be deduced from (7) and (8) for $H(\mathbf{x})$, and from (11) for $\psi(\mathbf{z})$.

Given that the handled constraints are separated into rotational and translational types, it will be shown that $H(\mathbf{x})$ and $\psi(\mathbf{z})$ can be written as

$$H(\mathbf{x}) = \begin{bmatrix} H_T(\mathbf{x}_T, \mathbf{x}_R) \\ H_R(\mathbf{x}_R) \end{bmatrix} \tag{14}$$

$$\psi(\mathbf{z}) = \begin{bmatrix} \psi_T(\mathbf{z}_T, \mathbf{z}_R) \\ \psi_R(\mathbf{z}_R) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_T \\ \mathbf{x}_R \end{bmatrix} \tag{15}$$

where $\mathbf{x} = [\mathbf{x}_T\ \mathbf{x}_R]^T$ such that $\mathbf{x}_T \in \mathbb{R}^3$ and $\mathbf{x}_R$ is an Euler angle parameterization of $SO(3)$, and $\mathbf{z} = [\mathbf{z}_T\ \mathbf{z}_R]^T$ such that $\mathbf{z}_T$ and $\mathbf{z}_R$ contain the parameters associated to the translational and rotational DOF, respectively.

The Pfaffian constraint matrix for the above holonomic constraints has the form

$$A(\mathbf{x}) = \frac{\partial H}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial H_T}{\partial \mathbf{x}_T} & \frac{\partial H_T}{\partial \mathbf{x}_R} \\ 0 & \frac{\partial H_R}{\partial \mathbf{x}_R} \end{bmatrix}. \tag{16}$$

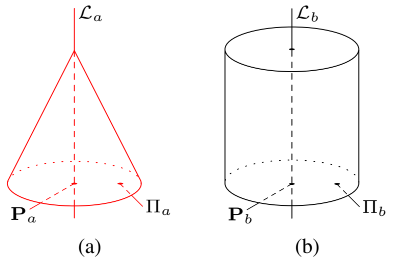


Fig. 8. Virtual environment composed of a cone (a) and a cylinder (b).

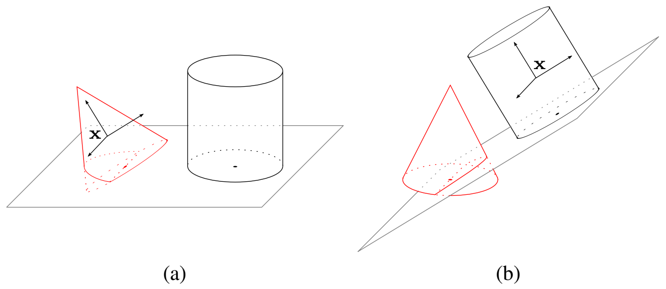


Fig. 9. Example constraint scenarios. Objects are constrained according to the *point-plane* coincidence $d(\mathbf{P}_a, \Pi_b) = 0$.
(a) The virtual tool $\mathbf{x}$ is attached to the cone and the cylinder remains fixed.
(b) The virtual tool $\mathbf{x}$ is attached to the cylinder and the cone remains fixed. Note that in (b), changing the orientation of the virtual tool changes the orientation of Π_b, and that the center of rotation is fixed at $\mathbf{P}_a$.

$A(\mathbf{x})$ must have *full row rank* (i.e., it cannot contain redundant constraints), otherwise the $AM^{-1}A^T$ term in (9) and (10) would not be invertible. This condition is always fulfilled since the solver removes all redundant constraints as part of the solution process (Section III-A), and the operator is allowed to create constrained scenarios with redundancies, if such a formulation is deemed more convenient.

The process for obtaining $H(\mathbf{x})$ and $\psi(\mathbf{z})$ will be first described for an example, and then extended to all the handled scenarios. Consider a virtual environment composed of the objects depicted in Figs. 8a and 8b, and let them be constrained according to the *point-plane* coincidence $d(\mathbf{P}_a, \Pi_b) = 0$, which only restricts one translational DOF. Since no rotational DOF is being constrained, $H_R(\mathbf{x}_R) = 0$ and $\psi_R(\mathbf{z}_R)$ can have the same parameterization of $SO(3)$ as $\mathbf{x}_R$. The translational constraint equation is then

$$\hat{\mathbf{n}}_{\Pi_b}(\mathbf{P}_a - \mathbf{P}_b) = 0 \tag{17}$$

and a parametric representation of the allowed positions of $\mathbf{P}_a$ is

$$\mathbf{P}_a = \mathbf{P}_b + \hat{\mathbf{d}}_1 z_1 + \hat{\mathbf{d}}_2 z_2 \tag{18}$$

where $\mathbf{P}_b$ is a point contained in Π_b and vectors $\hat{\mathbf{n}}_{\Pi_b}$, $\hat{\mathbf{d}}_1$, and $\hat{\mathbf{d}}_2$ are orthogonal.

If the virtual tool frame $\mathbf{x}$ is attached to the cone, and the cylinder remains fixed (Fig. 9a), the expression that relates the current position of the virtual tool $\mathbf{x}_T$ to that of $\mathbf{P}_a$ is given by

$$\mathbf{x}_T = \mathbf{P}_a + R_a \mathbf{e}_a \tag{19}$$

where R_a represents the rotation associated to object a, which is a function of $\mathbf{x}_R$ (and $\mathbf{z}_R$ as well through $\psi_R(\mathbf{z}_R) = \mathbf{x}_R$); and $\mathbf{e}_a = \mathbf{x}_{Ti} - \mathbf{P}_{ai}$ is a constant offset vector obtained from the initial values of $\mathbf{x}_T$ and $\mathbf{P}_a$.

Solving (19) for $\mathbf{P}_a$ and substituting it in (17) and (18) yields the expressions for H_T and ψ_T:

$$H_T(\mathbf{x}_T, \mathbf{x}_R) = \hat{\mathbf{n}}_{\Pi_b}(\mathbf{x}_T - R_a\mathbf{e}_a - \mathbf{P}_b) \tag{20}$$

$$\psi_T(\mathbf{z}_T, \mathbf{z}_R) = \mathbf{P}_b + \hat{\mathbf{d}}_1 z_1 + \hat{\mathbf{d}}_2 z_2 + R_a\mathbf{e}_a \tag{21}$$

When the virtual tool frame coincides with the constrained point $\mathbf{P}_a$—the rotation center of the object—the value of $\mathbf{e}_a$ becomes zero, and H_T and ψ_T no longer depend on $\mathbf{x}_R$ and $\mathbf{z}_R$, respectively. As a consequence, computing the derivatives of H_T and ψ_T becomes a much simpler task (e.g., the upper-right term in (16) vanishes $\partial H_T/\partial \mathbf{x}_R = 0$). In this particular configuration, orientation changes do *not* affect the position of the virtual tool.

If the virtual tool frame $\mathbf{x}$ is now attached to the cylinder, and the cone remains fixed (Fig. 9b), then $\mathbf{x}_T = \mathbf{P}_b + R_b\mathbf{e}_b$ and $\mathbf{e}_b = \mathbf{x}_{Ti} - \mathbf{P}_{bi}$; and the expressions for H_T and ψ_T become:

$$H_T(\mathbf{x}_T, \mathbf{x}_R) = R_b\hat{\mathbf{n}}_{\Pi_b}(\mathbf{P}_a - \mathbf{x}_T + R_b\mathbf{e}_b) \tag{22}$$

$$\psi_T(\mathbf{z}_T, \mathbf{z}_R) = \mathbf{P}_a + R_b(-\hat{\mathbf{d}}_1 z_1 - \hat{\mathbf{d}}_2 z_2 + \mathbf{e}_b) \tag{23}$$

Note that since the cylinder is now the mobile object, the orientation of plane Π_b will change as the operator rotates the virtual tool frame. Contrary to the previous scenario (virtual tool attached to the cone), there is no fixed position of the virtual tool where H_T and ψ_T become independent of $\mathbf{x}_R$ and $\mathbf{z}_R$.

Extending the previous example to the different translational and rotational constraint scenarios handled by the PMF solver yields the constraint equations and submanifold parameterizations listed in Table II. Once the corresponding table entries have been selected for a particular problem, the virtual tool is attached to one of the objects according to $\mathbf{x}_T = \mathbf{P}_* + R_*\mathbf{e}_*$, where $*$ identifies the mobile object. Then the expressions for $H(\mathbf{x})$ and $\psi(\mathbf{z})$ are computed. Note that if the virtual tool is attached to an object whose constrained element is not a point, but a line or a plane, then changes in the orientation of the virtual tool will affect the orientation of the element, as can be seen in (22) and (23).

The constraint equations for the one-dimensional translational and rotational submanifolds are not shown explicitly in Table II, but can be obtained by intersecting two *appropriately chosen* two-dimensional submanifolds:

1) Line: The line $\mathcal{L}$ with direction vector $\hat{\mathbf{d}}_\mathcal{L}$ that passes through the point $\mathbf{P}_\mathcal{L}$ can be obtained as the intersection of two planes, Π and Σ (Fig. 10a). Both planes contain the point $\mathbf{P}_\mathcal{L}$ and their normals satisfy $\hat{\mathbf{n}}_\Pi \times \hat{\mathbf{n}}_\Sigma = \hat{\mathbf{d}}_\mathcal{L}$.

2) Ellipse: The ellipse $\mathcal{E}$ with centerpoint $\mathbf{P}_\mathcal{E}$, semimajor and semiminor axis lengths p, q, and directions $\hat{\mathbf{d}}_1$, $\hat{\mathbf{d}}_2$, respectively, can be obtained as the intersection of a cylinder $\mathcal{C}$ and a plane Π (Fig. 10b). Plane Π is defined by the normal vector $\hat{\mathbf{n}}_\Pi = \hat{\mathbf{d}}_1 \times \hat{\mathbf{d}}_2$ and the point $\mathbf{P}_\mathcal{E}$. Cylinder $\mathcal{C}$ is defined by a radius equal to q and an axis $\mathcal{L}$ that passes through $\mathbf{P}_\mathcal{E}$ and its direction vector $\hat{\mathbf{d}}_\mathcal{L}$ satisfies $\hat{\mathbf{d}}_1 \times \hat{\mathbf{d}}_\mathcal{L} = \frac{q}{p}\hat{\mathbf{d}}_2$.

3) Parallel vectors: Vector $\hat{\mathbf{t}}$ is parallel to vector $\hat{\mathbf{w}}$ if it simultaneously satisfies $\angle(\hat{\mathbf{t}}, \hat{\mathbf{u}}) = \alpha$ and $\angle(\hat{\mathbf{t}}, \hat{\mathbf{v}}) = \beta$ (Fig. 10c), where α, β are two positive real numbers, $\hat{\mathbf{n}}$ is a unit vector normal to $\hat{\mathbf{w}}$, and $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$ satisfy $\hat{\mathbf{w}} \times \hat{\mathbf{u}} = s_\alpha\hat{\mathbf{n}}$ and $\hat{\mathbf{v}} \times \hat{\mathbf{w}} = s_\beta\hat{\mathbf{n}}$, respectively.

C. Handling other constraint types

The family of holonomic constraints that can be created online and haptically displayed by the system can be used to assist an operator in the execution of constrained movements that are common to many tasks (e.g., assembly/disassembly tasks). However, the dynamic equations of the simulated system (5) and (6) admit not only more general holonomic constraints, but also nonholonomic and inequality constraints. The simulation algorithm has been tested under such scenarios by means of hardcoded test examples (Faulring et al., 2007).

The following is a brief discussion on how these constraint types could be incorporated into the present system, so that they can also be created online (instead of hardcoded) and haptically displayed.

1) More general holonomic constraints: It would be of interest to restrict the haptic end-effector to more general curves and surfaces. The spray painting of a car hood, for example, requires the painter nozzle to remain at a fixed distance from a smooth surface as well as normal to it. Parametric curve/surface representations such as nonuniform rational B-splines (NURBS) could be used since they allow modelling quite general geometries, and have a relatively simple algebraic and parametric representation, which are required for computing $H(\mathbf{x})$ and $\psi(\mathbf{z})$, respectively. However, the simultaneous satisfaction of multiple NURBS-based constraints requires a detailed study in its own, so an initial effort would be to enforce the satisfaction of individual *point-curve* and *point-surface* constraints.

2) Nonholonomic constraints: While holonomic constraints reduce both the number of motion freedoms and the dimension of the solution submanifold, nonholonomic constraints only reduce the former, so they only affect $H(\mathbf{x})$, and leave $\psi(\mathbf{z})$ unchanged. In order to incorporate online nonholonomic constraint creation into the system, it would suffice to provide the means of specifying velocity-domain constraints, and additional logic for removing possible redundancies between them.

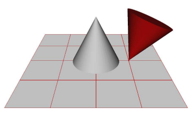
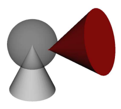
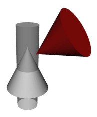
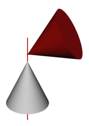
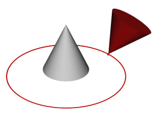
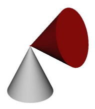
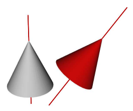
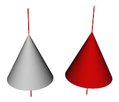
3) Inequality constraints: These constraints usually arise when modelling contact situations or specifying forbidden regions (e.g., a virtual wall). The usual way of haptically enforcing these constraints is by means of virtual spring/damper systems that depend on the penetration depth and speed of the virtual tool on the forbidden region. Incorporating collision detection capabilities into the simulation engine would make possible the computation of virtual spring and damper forces that can be added to the τ term in (5).

V. EXAMPLE HAPTIC DISPLAY SCENARIOS

Consider again the virtual environment composed of the cone and the cylinder shown in Figs. 8a and 8b. The end-

TABLE II

HOLONOMIC CONSTRAINTS AND PARAMETRIC REPRESENTATION OF TRANSLATIONAL AND ROTATIONAL SUBMANIFOLDS

Fundamental translational constraints	Constraint equation	Submanifold parameterization	Example
none	-	$\mathbf{P}_a = \mathbf{P}_b + \hat{\mathbf{d}}_1 z_1 + \hat{\mathbf{d}}_2 z_2 + \hat{\mathbf{d}}_3 z_3$ ($\mathbb{R}^3$, 3 translational DOF)	-
$d(\mathbf{P}_a, \Pi_b) = 0$	$\hat{\mathbf{n}}_{\Pi_b}(\mathbf{P}_a - \mathbf{P}_b)$	$\mathbf{P}_a = \mathbf{P}_b + \hat{\mathbf{d}}_1 z_1 + \hat{\mathbf{d}}_2 z_2$ (plane, 2 translational DOF)	
$d(\mathbf{P}_a, \mathbf{P}_b) = p$	$\lVert\mathbf{P}_a - \mathbf{P}_b\rVert^2 - p^2$	$\mathbf{P}_a = \mathbf{P}_b + p(c_{z_1}s_{z_2}\hat{\mathbf{d}}_1 + s_{z_1}s_{z_2}\hat{\mathbf{d}}_2 + c_{z_2}\hat{\mathbf{d}}_3)$ (sphere*, 2 translational DOF)	
$d(\mathbf{P}_a, \mathcal{L}_b) = p$	$\lVert\mathbf{P}_a - \mathbf{P}_b\rVert^2 - [\hat{\mathbf{d}}_{\mathcal{L}_b}(\mathbf{P}_a - \mathbf{P}_b)]^2 - p^2$	$\mathbf{P}_a = \mathbf{P}_b + z_1\hat{\mathbf{d}}_{\mathcal{L}_b} + r(c_{z_2}\hat{\mathbf{d}}_1 + s_{z_2}\hat{\mathbf{d}}_2)$ (cylinder, 2 translational DOF)	
$d(\mathbf{P}_a, \mathcal{L}_b) = 0$	Two *point-plane* coincidences	$\mathbf{P}_a = \mathbf{P}_b + \hat{\mathbf{d}}_{\mathcal{L}_b} z_1$ (line, 1 translational DOF)	
$d(\mathbf{P}_a, \mathcal{L}_b) = p$, $d(\mathbf{P}_a, \Pi_b) = 0$	*Point-line* distance, *point-plane* coinc.	$\mathbf{P}_a = \mathbf{P}_b + ac_{z_1}\hat{\mathbf{d}}_1 + bs_{z_1}\hat{\mathbf{d}}_2$ (ellipse†, 1 translational DOF)	
$d(\mathbf{P}_a, \mathbf{P}_b) = 0$	$\mathbf{P}_a - \mathbf{P}_b$	$\mathbf{P}_a = \mathbf{P}_b$ (point, 0 translational DOF)	
Fundamental rotational constraints	**Constraint equation**	**Submanifold parameterization**	**Example**
none	-	$R_a = R(\hat{\mathbf{d}}_1, z_4)R(\hat{\mathbf{d}}_2, z_5)R(\hat{\mathbf{d}}_3 z_6)$ ($SO(3)$, 3 rotational DOF)	-
$\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = \alpha$	$R_a(\hat{\mathbf{u}}_a)\hat{\mathbf{u}}_b - \alpha$	$R_a = R(\hat{\mathbf{u}}_b, z_4)R(\hat{\mathbf{u}}_a, z_5)$ (Angle vectors, 2 rotational DOF)	
$\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = 0$	Two *vector-vector* angles	$R_a = R(\hat{\mathbf{u}}_b, z_4)$ (Parallel vectors, 1 rotational DOF)	
$\angle(\hat{\mathbf{u}}_a, \hat{\mathbf{u}}_b) = 0$, $\angle(\hat{\mathbf{v}}_a, \hat{\mathbf{v}}_b) = 0$	$R_a - R_b$	$R_a = R_b$ (Fixed rotation‡, 0 rotational DOF)	-

where:
- $\mathbf{P}_b$ represents a reference point (e.g., a sphere or ellipse center, a point belonging to a plane or line).
- Vectors $\hat{\mathbf{d}}_1$, $\hat{\mathbf{d}}_2$, and $\hat{\mathbf{d}}_3$ are orthogonal.
- R_a and R_b are the rotations associated to objects a and b, respectively, $R(\hat{\mathbf{u}})$ represents a rotation applied to a vector, and $R(\hat{\mathbf{u}}, \alpha)$ represents a rotation given by an axis-angle pair.

*The spherical coordinates parameterization of a spherical surface (with nonzero radius) has two singularities at the poles. Singularity-free parametric representations of spherical surfaces exist, such as those that use overset grids (Kageyama and Sato, 2004), but involve more complex expressions.

†*Point-ellipse* constraints cannot be directly expressed in terms of a single fundamental constraint, but rather two.

‡Other constraint sets can also fix the orientation of an object, such as three angle constraints or one parallelism plus one angle constraint (provided that they are compatible and nonredundant). These cases are also handled by the system.

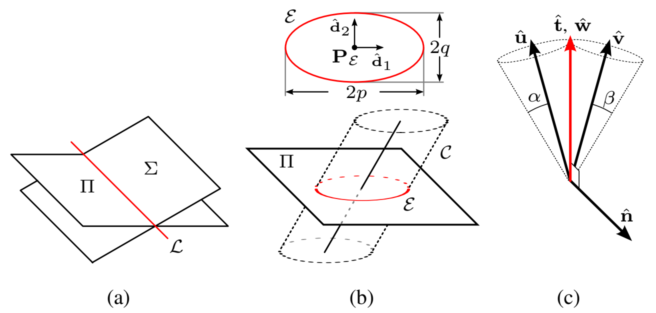


Fig. 10. Generation of one-dimensional submanifolds from the intersection of pairs of two-dimensional ones.
(a) Line $\mathcal{L}$ can be obtained by intersecting planes Π and Σ.
(b) Ellipse $\mathcal{E}$ can be obtained by intersecting cylinder $\mathcal{C}$ and plane Π.
(c) A vector $\hat{\mathbf{t}}$ that simultaneously satisfies $\angle(\hat{\mathbf{t}}, \hat{\mathbf{u}}) = \alpha$ and $\angle(\hat{\mathbf{t}}, \hat{\mathbf{v}}) = \beta$ is also parallel to $\hat{\mathbf{w}}$.

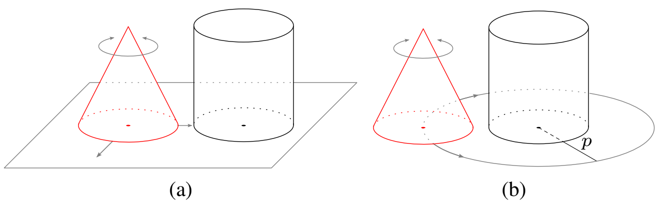


Fig. 11. Simulated constraint scenarios. The cone is being manipulated by the haptic device, and is initially constrained to (a) a planar workspace through $d(\Pi_a, \Pi_b) = 0$. Then, the cone is further constrained (b) with $d(\mathcal{L}_a, \mathcal{L}_b) = p$, so that it can only rotate about its axis, and translate along a circular path. Arrows indicate DOF directions.

effector of the haptic display is attached to the cone so that operator movements are mapped to movements of the cone.

A. Plane-plane coincidence

Starting from an unconstrained configuration, a *plane-plane* coincidence constraint $d(\Pi_a, \Pi_b) = 0$ is created (Fig. 11a), that when decomposed into fundamental constraints yields

$$C_T = \{d(\mathbf{P}_a, \Pi_b) = 0\}, \quad C_R = \{\angle(\hat{\mathbf{n}}_{\Pi_a}, \hat{\mathbf{n}}_{\Pi_b}) = 0\}$$

where $\mathbf{P}_a \subset \Pi_a$. PMF then computes a transformation that positions the cone in a constraint-satisfying configuration. It does so by first solving the rotational component R for one *parallelism* constraint, and then the translational component T for one *point-plane* coincidence constraint. The corresponding solution submanifold representations are the "Parallel vectors" (1 DOF) and "Plane" (2 DOF) rows of Table II.

The position of the virtual tool frame $\mathbf{x}_T$ is made to coincide with $\mathbf{P}_a$, a point contained in the axis of rotation for the current scenario, so that the translational constraint equation H_T and the parametric representation ψ_T do not depend on the orientation of the object, but on $\mathbf{x}_T$ and $\mathbf{z}_T$, respectively, yielding:

$$H_T(\mathbf{x}_T) = \hat{\mathbf{n}}_{\Pi_b}(\mathbf{x}_T - \mathbf{P}_b) \tag{24}$$
$$\psi_T(\mathbf{z}_T) = \mathbf{P}_b + \hat{\mathbf{d}}_1 z_1 + \hat{\mathbf{d}}_2 z_2 \tag{25}$$

where $\mathbf{P}_b \subset \Pi_b$. For brevity, details of the rotational component are omitted.

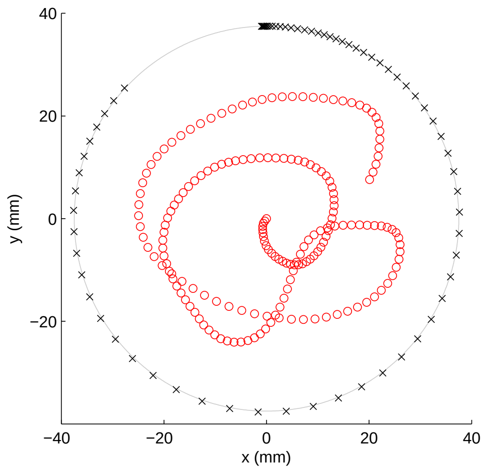


Fig. 12. Example of translational trajectories described by the haptic end-effector while constrained to a planar surface ("o" markers) and a circular curve ("x" markers). Movements normal to the figure plane are not allowed.

The dynamic simulation starts with the virtual tool in a constraint-satisfying configuration provided by the solver. The circular markers in Fig. 12 depict an example of the trajectories described by the end-effector of the Cobotic Hand Controller in the unconstrained translational directions, and Fig. 13 shows different measurements and performance metrics taken during the simulation.

B. Plane-plane coincidence and line-line distance

The cone is now further restricted by adding the additional constraint $d(\mathcal{L}_a, \mathcal{L}_b) = p$ (Fig. 11b). After decomposing into fundamental constraints the rotational subset of constraints C_R remains unchanged, but the translational subset becomes

$$C_T = \{d(\mathbf{P}_a, \Pi_b) = 0,\ d(\mathbf{P}_a, \mathcal{L}_b) = p\}$$

where $\mathbf{P}_a = \mathcal{L}_a \cap \Pi_a$. The translational component of the solution T now only has one DOF and is computed for the above constraint pair, that is equivalent to a *point-circle* coincidence, and whose submanifold representation corresponds to the "Ellipse" row of Table II, yielding:

$$H_T(\mathbf{x}_T) = \begin{bmatrix} \hat{\mathbf{n}}_{\Pi_b}(\mathbf{x}_T - \mathbf{P}_b) \\ \|\mathbf{x}_T - \mathbf{P}_b\|^2 - [\hat{\mathbf{d}}_{\mathcal{L}_b}(\mathbf{x}_T - \mathbf{P}_b)]^2 - p^2 \end{bmatrix} \tag{26}$$
$$\psi_T(\mathbf{z}_T) = \mathbf{P}_b + p(c_{z_1}\hat{\mathbf{d}}_1 + s_{z_1}\hat{\mathbf{d}}_2) \tag{27}$$

with $\mathbf{P}_b = \mathcal{L}_b \cap \Pi_b$. The rotational component of the solution R remains the same as before.

The cross markers in Fig. 12 depict an example of the circular path described by the end-effector, and Fig. 14 reports data relevant to the simulation. In Fig. 14a, forces applied by the operator normal and tangential to the circular constraint are shown. Tangential forces are small in magnitude because the virtual environment had little damping. At $t \sim 4.3s$ the direction in which the circle is being traversed is reversed, hence the tangential force peak (the simulated cone mass was 2kg), and at $t \sim 6s$ the operator lets go of the end-effector and

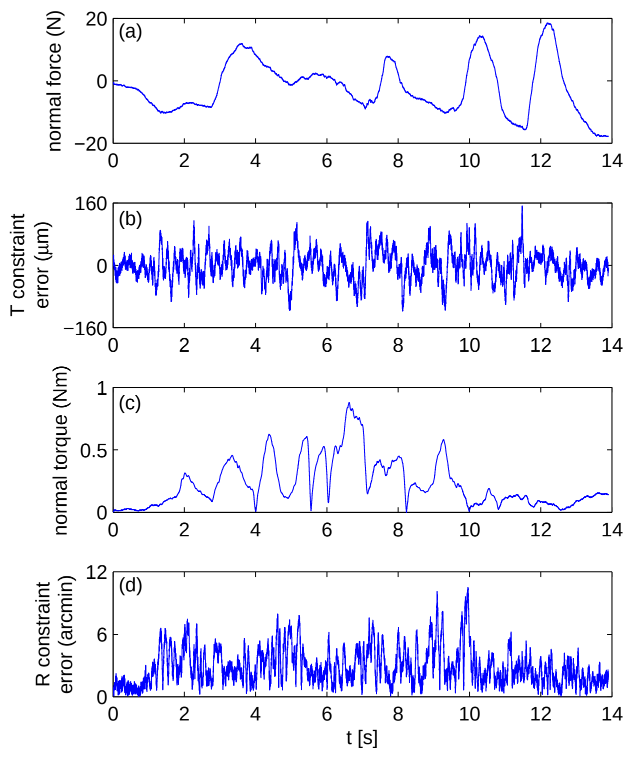


Fig. 13. Experimental data from the simulation of the *plane-plane* coincidence constraint $d(\Pi_m, \Pi_f) = 0$.
(a) Force applied by the operator normal to Π_f.
(b) Translational component error: distance from the end-effector to Π_f.
(c) Torque applied by the operator in directions nonparallel to the normal of Π_f.
(d) Rotational component error: angle between planes Π_m and Π_f.

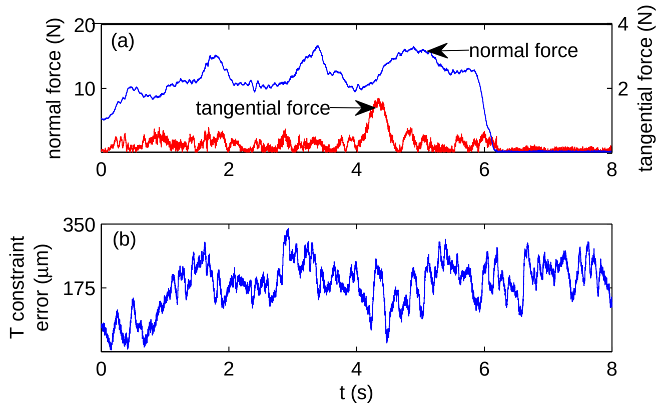


Fig. 14. Experimental data from the simulation of the constraint set $\{d(\Pi_m, \Pi_f) = 0, d(\mathcal{L}_m, \mathcal{L}_f) = p\}$, whose translational component restricts the haptic end-effector to a circular curve.
(a) Forces applied by the operator normal and tangential to the constraints. Notice that different vertical scales are used for each force component.
(b) Translational component error: distance from end-effector to circle.

the force values drop to zero. In Figs. 13 and 14, the ability of the Cobotic Hand Controller to render very rigid constraints is demonstrated by the maintenance of very small constraint errors even during the application of substantial forces and torques.

Further haptic display scenarios can be found in the video of Extension 1. They show how geometric constraint sets are created and modified in real time, and how the Cobotic Hand Controller is used to explore the associated solution submanifolds.

VI. Implementation Issues

A. Map from task space to haptic display workspace

A characterization of the workspace of the Cobotic Hand Controller can be found in Faulring et al. (2006), but for this discussion it suffices to say that its translational workspace can be approximated by a 8cm radius sphere, and that because of the parallel kinematic structure of the device, its rotational workspace is greatest at the workspace center, and decreases as the end-effector moves away from it.

Considering that the Cobotic Hand Controller has a finite workspace, and that there may exist a significant scale difference with the task workspace, task coordinates $\mathbf{x}_{task}$ are mapped to haptic display coordinates $\mathbf{x}$ by means of the following transformation:

$$\mathbf{x} = T_r\, T_s\, \mathbf{x}_{task} \tag{28}$$

where T_r is a rigid transformation that centers the current solution submanifold on the haptic workspace, and T_s is a translational scaling that maps task space dimensions into meaningful haptic workspace dimensions. For the bounded translational submanifolds (ellipses and spheres), T_s is calculated as a function of a characteristic length (radius and semimajor axis); and for the unbounded translational submanifolds ($\mathbb{R}^3$, lines, and planes), T_s can be either chosen by the user, or automatically computed by a heuristic method.

The computation of T_r and T_s for the examples described in Section V is depicted in Fig. 15. The first example (Subsection V-A) is translationally constrained to a plane, which is an unbounded surface. In this scenario T_r maps $\mathbf{x}_{task}$ to the origin of the haptic workspace o (Figs. 15a and 15b), and T_s applies a constant scaling factor to all directions given by the homogeneous matrix

$$T_s = \begin{bmatrix} k\, I_{n\times n} & 0 \\ 0 & 1 \end{bmatrix} \tag{29}$$

where

$$k = \frac{2nr_w}{\sqrt{d_1^2 + d_2^2 + d_3^2}}. \tag{30}$$

The heuristic used to compute the scaling factor k maps the major diagonal of the virtual environment bounding box[2] (measured before the simulation starts) to a fraction of the haptic workspace diameter ($2nr_w$, where $n > 0$). A value of $n = 0.5$ was used in the simulations. Situations where the

[2]The bounding box computation is performed by the graphics library used to render the virtual environment (Coin3D, 2008), an open source implementation of the Open Inventor Application Programming Interface (API).

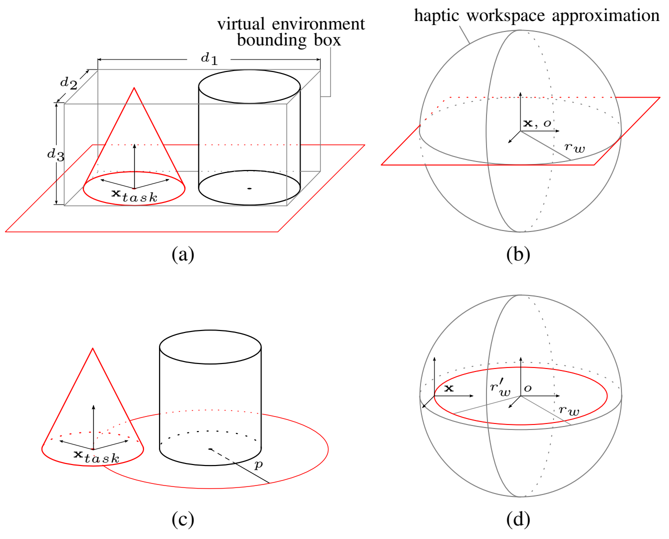


Fig. 15. Map between task and haptic display coordinates: (a) → (b), (c) → (d).

values of d_1, d_2, and d_3 differ considerably may benefit from independent direction scaling.

On the other hand, the second example (Subsection V-B) is translationally constrained to a circle, which is a bounded curve. Here, T_r transforms $\mathbf{x}_{task}$ so that the center of the circular constraint submanifold coincides with the origin of the haptic workspace o (Figs. 15c and 15d), and T_s is calculated using (29) with $k = r'_w/p$. A fraction of the workspace radius $r'_w = 0.5r_w$ has been used used instead of r_w so that the rotational workspace along the circular curve is sufficiently large to allow acceptable actuation of the rotational DOF.

B. *Software and computational performance*

From am implementation standpoint, all code was written in C++. The constraint solver and the GUI were executed on a desktop PC running Linux, although the code is multi-platform and can also be run under Windows and Mac operating systems. The constrained dynamics simulation engine and control module were executed on a PC located in the control box of the Cobotic Hand Controller running the QNX real time operating system. Communications between the two computers were handled by a client/server architecture using TCP/IP sockets.

In terms of computational performance, PMF took $130\mu s$ and $170\mu s$ to compute the solutions for the two scenarios discussed in Section V (Pentium 4 processor with a 3.5GHz CPU clock), which was a one-time operation. The dynamics simulation was updated at a frequency of 1kHz.

VII. Conclusions

The present work integrates into a single framework a constrained dynamics simulation engine that permits simulating very general dynamic behaviors, with a geometric constraint solver that serves as an intuitive interface for creating and combining virtual holonomic constraints. Two constraint scenarios are provided and the results of their implementation on the Cobotic Hand Controller haptic display examined.

The quality of haptic feedback from a remote location and the usefulness of locally generated constraints are important for teleoperation. The use of rigid virtual holonomic constraints was demonstrated, in which the task space is reduced to allow only movements that are meaningful for the present task, so that operator burden may be reduced. The experiments presented here also demonstrate simulation of desired inertial dynamics, in which an operator is able to feel the dynamic behavior of the teleoperated object and not that of the master device.

Future lines of work include extending the constraint solver interface to admit more general (e.g., NURBS-based) curve and surface representations, and incorporating collision detection capabilities into the dynamic simulation engine.

Acknowledgements

The authors would like to acknowledge Kevin Lynch for his help in developing the dynamics simulation method, as well as the anonymous reviewers for their valuable suggestions for improving the contents of this paper.

Appendix

The multimedia extensions to this article can be found online by following the hyperlinks from www.ijrr.org.

TABLE III
Table of multimedia extensions

Extension	Media type	Description
1	Video	Haptic and visual display of constraint scenarios involving the objects of a virtual environment (Part I).
2	Video	Haptic and visual display of constraint scenarios involving the objects of a virtual environment (Part II).

References

Abbott, J. J. and Okamura, A. M. (2007). Pseudo-admittance bilateral telemanipulation with guidance virtual fixtures. *Int. Journal of Robotics Research*, 26(8):865–884.

Anderson, R. and Spong, M. (1989). Bilateral control of teleoperators with time delay. *IEEE Transactions on Automatic Control*, 34(5):494–501.

Chopra, N. and Spong, M. (2006). Output synchronization of nonlinear systems with time delay in communication. In *Proc. of the IEEE Conf. on Decision and Control*, pages 4986–4992.

Coin3D (2008). 3D graphics development tools. http://www.coin3d.org.

De Schutter, J., De Laet, T., Rutgeerts, J., Decré, W., Smits, R., Aertbeliën, E., Claes, K., and Bruyninckx, H. (2007). Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *Int. Journal of Robotics Research*, 26(5):433–455.

Faulring, E. L. (2005). *The Cobotic Hand Controller: Design, Control and Analysis of a Novel Haptic Display*. PhD thesis, Northwestern University.

Faulring, E. L., Colgate, J., and Peshkin, M. A. (2004). A high performance 6-dof haptic cobot. In *IEEE Int. Conf. Robot. Automat.*, pages 1980–1985, New Orleans.

Faulring, E. L., Colgate, J., and Peshkin, M. A. (2006). The cobotic hand controller: Design, control and performance of a novel haptic display. *Int. Journal of Robotics Research*, 25(11):1099–1119.

Faulring, E. L., Lynch, K. M., Colgate, J., and Peshkin, M. A. (2007). Haptic display of constrained dynamic systems via admittance displays. *IEEE Trans. Robot.*, 23(1):101–111.

Force Dimension (2008). http://www.forcedimension.com/products.

Freeman-Benson, B., Maloney, J., and Borning, A. (1990). An incremental constraint solver. *Communications of the ACM*, 33(1):54–63.

Hoffmann, C. (2005). Constraint-based CAD. *Journal of Computing and Information Science in Engineering*, 5(3):182–197.

Kageyama, A. and Sato, T. (2004). "yin-yang grid": An overset grid in spherical geometry. *Geochemistry Geophysics Geosystems*, 5(Q09005):doi:10.1029/2004GC000734.

Kramer, G. (1992). *Solving Geometric Constraint Systems*. MIT Press.

Latham, R. and Middleditch, A. (1996). Connectivity analysis: A tool for processing geometric constraints. *Computer-Aided Design*, 28(11):917–928.

Li, M., Kapoor, A., and Taylor, R. H. (2007). Telerobotic control by virtual fixtures for surgical applications. In Verlag, S., editor, *Springer Trans. in Advanced Robotics – Advances in Telerobotics*, volume 31, chapter 22, pages 381–401.

Liu, G. and Li, Z. (2002). A unified geometric approach to modelling and control of constrained mechanical systems. *IEEE Trans. Robot. Automat.*, 18(4):574–587.

Massie, T. and Salisbury, J. (1994). The PHANToM haptic interface, a device for probing virtual objects. In *ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 295–302, Chicago, IL.

Nuño, E., Basañez, L., Ortega, R., and Spong, M. (2008). On position tracking for nonlinear teleoperators with variable time-delay. *The Int. Journal of Robotics Research (In press)*.

Porta, J. M., Ros, L., Thomas, F., and Torras, C. (2005). A branch-and-prune solver for distance constraints. *IEEE Trans. Robot.*, 21(2):176–187.

Rodríguez, A., Basañez, L., and Celaya, E. (2007). Description of a robotics-oriented relational positioning methodology. In *Technical Report*, Technical University of Catalonia. Available: https://upcommons.upc.edu/e-prints/handle/2117/1531?locale=en.

Rodríguez, A., Basañez, L., and Celaya, E. (2008a). A relational positioning methodology for robot task specification and execution. *IEEE Trans. Robot.*, 24(3):600–611.

Rodríguez, A., Basañez, L., Colgate, J. E., and Faulring, E. L. (2008b). Haptic display of dynamic systems subject to holonomic constraints. In *IEEE Int. Conf. Intell. Robots Syst.*, Nice, France.

Rosenberg, L. B. (1993). Virtual fixtures: Perceptual tools for telerobotic manipulation. In *Proc. of the IEEE Annual Int. Symposium on Virtual Reality*, pages 76–82.

Ruspini, D. and Khatib, O. (2001). Haptic display for human interaction with virtual dynamic environments. *Journal of Robotic Systems*, 18(12):769–783.

Salisbury, K., Eberman, B., Levin, M., and Townsend, W. (1991). The design and control of an experimental whole-arm manipulator. In *The Fifth International Symposium on Robotics Research*, pages 233–241.

Taylor, R. H., Jensen, P., Whitcomb, L. L., Barnes, A., Kumar, R., Stoianovici, D., Gupta, P., Wang, Z., d. Juan, E., and Kavoussi, L. (1999). A steady-hand robotic system for microsurgical augmentation. In *Presented at the Medical Image Computing and Computer-Assisted Intervention*, Cambridge, UK.

Turro, N., Khatib, O., and Coste-Maniere, E. (2001). Haptically augmented teleoperation. In *IEEE Int. Conf. Robot. Automat.*, pages 386–392, Seoul, Korea.

Van der Linde, R., Lammerste, P., Frederiksen, E., and Ruiter, B. (2002). The HapticMaster, a new high-performance haptic interface. In *Eurohaptics*, Edinburgh, UK.

Yun, X. and Sarkar, N. (1998). Unified formulation of robotic systems with holonomic and nonholonomic constraints. *IEEE Trans. Robot. Automat.*, 14(4):640–650.

Zilles, C. and Salisbury, J. (1995). A constraint-based god-object method for haptic display. In *IEEE Int. Conf. Intell. Robots Syst.*, pages 146–151.