An Exact Solver for Geometric Constraints with Inequalities

Nikhil Somani, Markus Rickert, Alois Knoll

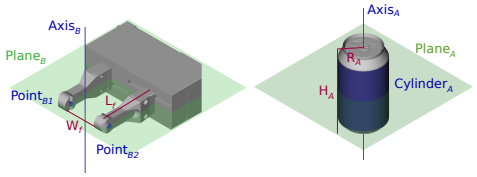
***Abstract*—CAD/CAM approaches have been used in the manufacturing industry for a long time, and their use in robotic systems is becoming more popular. One common element in these approaches is the use of geometric constraints to define relative object poses. Hence, approaches for solving these geometric constraints are critical to their performance. In this work, we present an exact solver for geometric constraints. Our approach is based on mathematical models of constraints and geometric properties of constraint nullspaces. Our constraint solver supports non-linear constraints with inequalities, and also mixed transformation manifolds, i.e., cases where the rotation and translation components of the constraints are not independent. Through several applications, we show how inequality constraints and mixed transformation manifolds increase the expressive power of constraint-based task definitions. The exact solver provides repeatable solutions with deterministic runtimes and our experiments show that it is also much faster than comparable iterative solvers.**

***Index Terms*—Control Architectures and Programming; Optimization and Optimal Control; Robust/Adaptive Control of Robotic Systems**

I. Introduction

Our main motivation in this work is to demonstrate how geometric information from CAD models can be exploited for constraint-based descriptions of robotic tasks. The use of CAD software to design mechanical parts and assemblies is established standard practice that has proven to be of immense value, especially with the increasing availability of computational power and maturity of such software. In popular CAD software such as SolidWorks, constraints between the individual geometric entities of each part can be defined to create assemblies. We show how such a constraint-based approach can be extended for other applications.

The study of geometric constraints is a central theme in this work. Geometric constraints can be categorized in several ways. Firstly, depending on equations representing the constraints between objects, they can be classified as linear or non-linear. Secondly, the rotation and translation components of the constraints may or may not be independent. This independence greatly simplifies the constraint solving algorithm [1]. However, as we have demonstrated in this paper, this limitation restricts the expressive power of constraint-based descriptions. We propose a solution for cases where the rotation and translation constraints depend on each other.



$$^{L_f}/_2 - R_A \leq \text{ParallelDistance}(\text{Axis}_A, \text{Axis}_B) \leq {}^{L_f}/_2$$
$$-{}^{H_A}/_2 \leq \text{Distance}(\text{Plane}_A, \text{Plane}_B) \leq {}^{H_A}/_2$$
$$R_A \leq \text{Distance}(\text{Axis}_A, \text{Point}_{B1}) \leq W_f - R_A$$
$$R_A \leq \text{Distance}(\text{Axis}_A, \text{Point}_{B2}) \leq W_f - R_A$$

Fig. 1: A cup grasping task expressed using geometric constraints with inequalities. The rotation and translation components in this example are not independent.

Although geometric constraint solving is a general problem that is relevant to several domains, we focus on robotics applications in this work. Through our experiments, we have observed that many robotic tasks are well-suited for constraint-based descriptions. This is due to the fact that robotic tasks are often under-constrained, i.e., they do not completely define the desired pose of the robot, but only impose certain constraints on it. As an example, a welding task requires the tool-tip to follow a line or a curve (Fig. 6). However, the orientation of the tool-tip need not be fully constrained. The nullspace of this task can be utilized to optimize the robot motion by staying away from joint limits or to satisfy secondary goals such as collision avoidance. With the inclusion of inequalities in constraints, restrictions in the form of allowed ranges can be defined for this orientation. This might be necessary to keep the welding tool within the acceptable angular limits such that the weld quality is maintained. This allows us to exploit the nullspace, but also impose restrictions on it based on requirements of the task.

There are two major categories geometric constraint solving approaches [7]: Iterative and exact. Iterative approaches model the constraints using cost functions and pose the solving process as an optimization problem. Exact solvers use geometric properties of the involved entities to create constraint simplification and combination rules, along with a mapping of constraints to geometric nullspaces. Iterative approaches are generally easier to model and can represent constraints which can be difficult to model geometrically. However, they

N. Somani and A. Knoll are with Robotics and Embedded Systems, Department of Informatics, Technische Universität München, Munich, Germany.
M. Rickert is with fortiss, An-Institut Technische Universität München, Munich, Germany.
Correspondence should be addressed to somani@in.tum.de.
Manuscript received: September, 10, 2016; Revised December, 04, 2016; Accepted January, 02, 2017.
This paper was recommended for publication by Editor Tamim Asfour upon evaluation of the Associate Editor and Reviewers' comments.
Digital Object Identifier (DOI): see top of this page.

TABLE I: Comparison of constraint solvers

Framework	Exact	Non-separable R,t	Inequalities	Efficient	Supported Use-Cases (Table IV)	Output Type
TSID [2], [3]		✓		+	A,C	$\boldsymbol{\tau}$
WBCF [3], [4]		✓		+	A,C	$\boldsymbol{\tau}$
Somani et al. (GN solver) [5]		✓		−−	A,C	$\boldsymbol{q}$
Rodriguez et al. [1]	✓			++	A,C	$\boldsymbol{x}$
Somani et al. (NLOpt) [6]		✓	✓	−	A,B,C,D,E,F	$\boldsymbol{q}$
This work	✓	✓	✓	++	A,B,C,D,E,F	$\boldsymbol{x}$

TABLE II: Classification of constraints

Non-separable R,t	Inequality	Example	Transformation Manifold (Translation, Rotation)
		Plane-Plane Coincident	Plane, OneParallel
	✓	Plane-Plane Distance Min-Max	Box, OneParallel
✓		Line-Line Distance, Tangent	1-DOF sub-manifold (Fig. 2c)
✓	✓	Line-Line Distance Min-Max, Cylinder-Plane Tangent	2-DOF sub-manifold (Fig. 2g)

inherit problems from the non-linear optimization domain, such as a lack of convergence guarantee with different starting values, local minima, numerical stability issues when using derivatives, and a non-deterministic runtime. Exact solvers are designed to give repeatable and guaranteed results from any starting pose, with a deterministic runtime which can be significantly faster than iterative approaches (Section VII).

We present an exact solver that supports non-linear geometric constraints with inequalities, where the rotation and translation components may not be independent. One example of a robotics use-case where both these properties are essential is shown in Fig. 1. We use the set of robotics applications that were presented in our previous work [6] using an iterative solver, and show how they can be solved using an exact solver. We get exact, repeatable solutions for the same applications, with an improvement in runtime by a factor of approximately 10. The runtime of approximately 50 µs that can be achieved using our exact solver makes it suitable for robot control applications. The deterministic nature of the runtime is very important for applications requiring hard real-time control with time slot management. Although environment and robot model constraints are also necessary for such applications (as shown in [6]), they are beyond the scope of this paper which focuses only on geometric inter-relational constraints.

II. Context

While using iterative solvers for robotic manipulation tasks, we have observed that geometric constraints often have the most significant effect on the solver in terms of the convergence properties as well as runtime. In this paper, we propose a method to improve both aspects by solving the geometric constraints exactly. The exact solution manifold of geometric constraints is essentially a geometric constraint on its own between the robot and the environment. An iterative solver can combine this geometric constraints with environment and other constraints as shown in our previous work [6]. By adding an inverse kinematics step after the exact solution of geometric constraints (see Section VII-B), a robot controller can be developed that is still within the requirements of real-time interfaces from most industrial robots (e.g., 400 µs for Comau C5G Open, 1 ms for KUKA FRI, 4 ms for KUKA RSI, 7.11 ms for Mitsubishi Electric, 8 ms for Universal Robots).

Although we present only robotics examples in this work, geometric constraints find use in several fields such as manipulation planning and computer vision. In our previous work [8] on object pose estimation, geometric constraints were used to generate object pose hypotheses. This typically requires generation of hundreds of hypothesis for each frame of sensor data. In manipulation planning [9], each step of each branch of the RRT requires a projection into the geometric nullspace. Hence, the runtime of geometric constraint solving is extremely important for each of these applications.

III. Related Work

Constraint-based methods for robot tasks have been studied for a long time, from operational space control concepts in the late 80s [10] to whole body control [4] and constraint between frames in iTaSC [11], with applications in safe human-robot collaboration [12] and motion primitives [13].

Applications based on representations of coordinate frames and geometric relations between them were presented in [14]. The work in [1], [5], and [6] defined relations between geometric entities (e.g., points, lines, surfaces) that comprise manipulation objects, and used an iterative solver for solving these geometric constraints. Borghesan et al. [15] defined formal models of geometric constraints and their control law for positioning tasks minimized distance functions derived from geometric constraints.

Berenson et al. [9] used task space regions (TSR) to represent constraints and affordances for robotic tasks. A TSR is essentially a subspace in SE(3) whose shape depends on the specific task. The sampling step for probabilistic motion

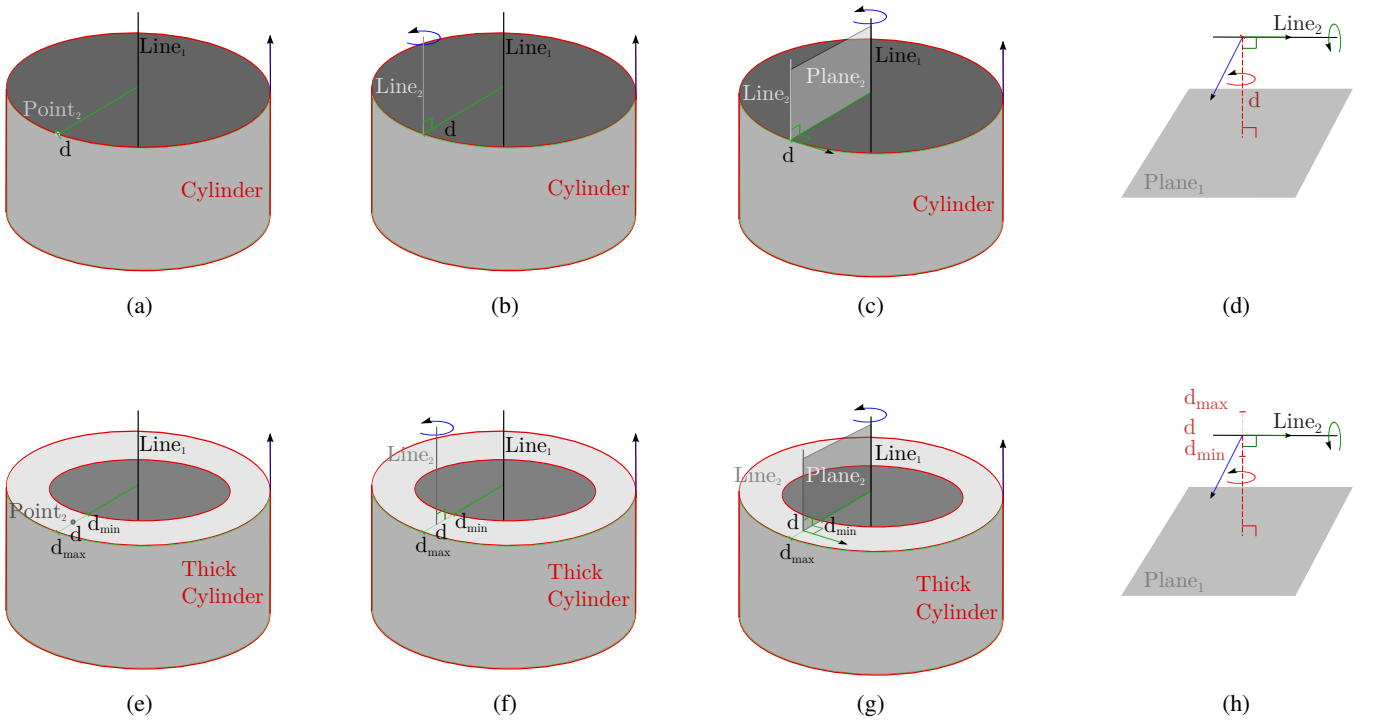
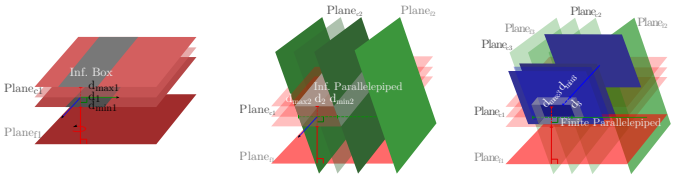


Fig. 2: Illustration of geometric constraints and their nullspaces. (a) The Line-Point distance constraint (red) defines the distance of $Point_2$ from $Line_1$. The translation nullspace of this constraint is the $Cylinder$ (red). The rotation is completely unconstrained. (b) The line-line distance constraint (red) defines the distance of a point on $Line_2$ from $Line_1$. The Axis-Axis-parallel constraint (green) defines that the axes of the two lines should be parallel. The translation nullspace is the $Cylinder$ (red). The rotational nullspace is shown using the blue arrow. (c) Example of a set of constraints, where the rotation and translation are non-separable. The Line-Point distance constraint (red) defines the distance of the center point of the plane from the line. The Line-Circle-tangent constraint (green) defines that the normal direction of the plane should be tangential to the circle shown in red. In a combination of these constraints, the rotation and translation elements are non-separable. The composite nullspace of this transformation manifold is the rotation around the line (as shown in blue). (d) The Plane-Line distance constraint (red) defines the distance of $Line_2$ from $Plane_1$. The translation nullspace is indicated by the green and blue arrows. The rotation nullspace is around the axes marked by red and green arrows. (e)–(h) The constraints are extended with inequalities to define the minimum and maximum distances.

planning is then performed by either rejection, projection or direct sampling in the TSR. The exact geometric solver that we propose in this work generates a transformation nullspace that can be used as a TSR. Since each of these geometric nullspaces already has an exactly defined distance function, projection function, and a parametric representation, all three types of sampling strategies in [9] can be used.

An exact constraint solver for primitive shapes was presented in [1]. The concepts of translation and rotation manifolds, constraint combination rules, and some examples of their use in robotics applications were presented in this work. We build upon these concepts and extend them in several directions. Firstly, we remove the restriction of requiring independent rotation and translation constraints. As shown in this work, this dependence between rotation and translation manifolds is common in the constraint-based definition of robotic tasks. Secondly, we add definitions of inequalities for geometric constraints and the corresponding combination rules, which greatly enhances their expressiveness. The importance of inequalities in robotic tasks has already been studied in several works including [6], [16], [17], where it has been shown that many robotics tasks are naturally expressed using inequalities.

A qualitative evaluation of constraint-based approaches with respect to features and runtime is presented in Table I. It can be observed from this table, that the approach proposed in this work offers a unique set of features compared to the other approaches. The runtimes are roughly classified as $--$ being less than 4 ms, $-$ being between 4 ms and 1 ms, and $+$ being between 1 ms and 0.1 ms, and $++$ being less than 0.1 ms. It is clear that the exact approaches are generally much faster than the iterative solvers. Solvers based on local linearizations such as TSID and WBCF are also very efficient, but are restricted in their scope of supported geometric constraints.



(a) (b) (c)

Fig. 3: Visualization of nullspaces of combinations of geometric constraints: (a) One Plane-Plane distance-min-max constraint generates a Box with two sides of infinite length as the nullspace, where the constrained axis of the Box lies along the normal direction of the fixed plane, (b) two Plane-Plane distance-min-max constraints generate an infinite Parallelepiped as the nullspace, where the infinite axis of the Parallelepiped lies along the cross product of the normal directions of the fixed planes, (c) three Plane-Plane distance-min-max constraints generate a finite Parallelepiped as the nullspace.

IV. Constraint Modeling

In this section, the mathematical models of geometric constraints are presented. This includes definitions of constraint nullspaces and an analysis of their geometric properties.

A. Transformation submanifolds

Submanifolds for geometric constraints whose rotation and translation elements are decoupled were defined for relative positioning tasks in [1]. We add definitions for some additional submanifolds (e.g., Box, Parallelepiped, ThickCylinder) that are required for inequality constraints. Mathematical definitions and properties of translation submanifolds are summarized in Table VI. The equations and properties of rotation submanifolds are very similar to the ones defined in [1].

We also define full transformation submanifolds, where each DoF in the submanifold nullspace can be a mixture of rotation and translation DoFs. An example of such a manifold is shown in Fig. 2c where the rotation and translation elements depend on each other and cannot be separated. For any point lying on Line_2, the translation is restricted along the Cylinder submanifold. If the translation along the Cylinder submanifold is considered as a DoF, the rotation is fully defined by the two perpendicularity constraints. Conversely, if the rotation along axis Line_1 is considered to be the DoF, the translation is fully defined. Hence, the nullspace of this constraint is a 1-DoF manifold where the rotation and translation cannot be independently controlled at the same time.

B. Boundary representation of geometric entities

We use a boundary representation (BREP) [18] for all geometric entities in the robotic system. This representation decomposes each object into semantically meaningful primitive shapes (e.g., planes, cylinders). For each of these shape types, Table VI lists the parametric equation used to define the geometric nullspace, and a projection function to project a point onto the shape. Conversely, the nullspaces of geometric constraints can be expressed using transformation submanifolds, which can be modeled using BREP.

C. Geometric constraints

Given a fixed and a constrained shape, a geometric constraint adds restrictions to the relative pose of the constrained shape with respect to the fixed shape.

Table II lists the different types of geometric constraints based on two criteria: Inequality support and separability of rotation and translation components. This classification is important for the solving step in Section V.

Table III lists some of the geometric constraints that are supported by our system and are relevant to our use-cases. In case of separable constraints, the rotation and translation components have also been defined. The mapping of these constraints to transformation manifolds has also been defined.

D. Geometric properties of constraint nullspaces

The nullspace of geometric constraints can be expressed in the form of transformation submanifolds, which have a geometric meaning (with parametric equations and projection functions) of their own. Fig. 3 illustrates the geometric properties of the constraint manifolds for combinations of plane-plane-distance constraints with inequalities. Hence, the nullspace of such geometric constraints can often (but not always) be exported as CAD files and visualized using 3D rendering software.

V. Exact constraint solver

The geometric constraints solving problem can be defined as an approach to calculate a relative pose between two objects (in general), based on a set of constraints between their geometries. Hence, given a set of geometric constraints $\mathbb{C}$ between a fixed object O_f and a constrained object O_c and an initial transformation T_f^c, the goal is to compute a transformation $T_f^{ns(c)}$ that is the projection of T_f^c in the nullspace of the geometric constraints $\mathbb{C}$:

$$T_f^{ns(c)} = \text{projection}(\mathbb{C}, T_f^c)\,. \tag{1}$$

In case of iterative solvers, each constraint ($C_i \in \mathbb{C}$) defines a cost function $\text{CF}(C_i, T_f^c)$. The constraint solving problem is then expressed as an optimization problem:

$$T_f^{ns(c)} = \arg\min \sum_{C_i \in \mathbb{C}} \text{CF}(C_i, T_f^c)\,. \tag{2}$$

In case of exact geometric solvers, the set of constraints $\mathbb{C}$ is first decomposed into simpler constraints (and in case of separable constraints, their rotation and translation components) using the rules defined in Table III. These constraints are then combined according to specified constraint combination rules (see Table VII, Fig. 4, and Fig. 5). These constraint processing steps are important to simplify and reduce the set of input constraints to a set that can be easily mapped to submanifolds. Some of the rules for mapping sets of constraints to transformation submanifolds are shown in Table III. Given the transformation submanifold, the transformation T_f^c can be

TABLE III: Decomposition of geometric constraints and their mapping to Transformation Manifolds

Fixed	Constrained	Constraint (i)	Rotation	Translation	Transformation Manifold (Translation, Rotation)
Line_1	Point_2	Distance $(d_{\min}, d_{\max})$	-	$d_{\min} \leq \text{d}(\text{Line}_1, \text{Point}_2) \leq d_{\max}$	ThickCylinder, $\mathbb{R}^3$ (Fig. 2e)
Line_1	Line_2	Distance $(d_{\min}, d_{\max})$	$\angle(\text{a}_1,\text{a}_2) = 0$[1]	$d_{\min} \leq \text{d}(\text{Line}_1, \text{Point}_2) \leq d_{\max}$	ThickCylinder, OneParallel (Fig. 2f)
Line_1	Line_2	Angle $(a_{\min}, a_{\max})$	$a_{\min} \leq \angle(\text{a}_1,\text{a}_2) \leq a_{\max}$	–	$\mathbb{R}^3$, OneAngleMinMax
Plane_1	Point_2	Distance $(d_{\min}, d_{\max})$	–	$d_{\min} \leq \text{d}(\text{Plane}_1, \text{Point}_2) \leq d_{\max}$	Inf. Box, $\mathbb{R}^3$
Plane_1	Line_2	Distance $(d_{\min}, d_{\max})$	$\angle(\text{n}_1,\text{n}_2) = \pi/2$	$d_{\min} \leq \text{d}(\text{Plane}_1, \text{Point}_2) \leq d_{\max}$	Inf. Box, OneAngle (Fig. 2h)
Plane_1	Line_2	Angle $(a_{\min}, a_{\max})$	$\pi/2 - a_{\min} \leq \angle(\text{n}_1,\text{n}_2) \leq \pi/2 - a_{\max}$	–	$\mathbb{R}^3$, OneAngleMinMax
Plane_1	Plane_2	Distance $(d_{\min}, d_{\max})$	$\angle(\text{n}_1,\text{n}_2) = 0$	$d_{\min} \leq \text{d}(\text{Plane}_1, \text{Point}_2) \leq d_{\max}$	Plane, OneParallel
Plane_1	Plane_2	Angle $(a_{\min}, a_{\max})$	$a_{\min} \leq \angle(\text{n}_1,\text{n}_2) \leq a_{\max}$	–	$\mathbb{R}^3$, OneAngleMinMax
Line_1	Plane_2	Tangent	non-separable	non-separable	see Fig. 2g

Algorithm 1: Constraint solving approach: separation is a set of constraint decomposition rules defined in Table III. combinationRule is a set of constraint combination rules, some of which have been defined in Table VII and illustrated in Figs. 4 and 5. manifoldMap is the mapping of constraints to transformation manifolds, as defined (not exhaustively) in Table III. geomProject is a projection function for each translation and rotation transformation submanifold.

Input: Constraints $\mathbb{C}$, initial constrained object pose T_f^c

Output: Solved constrained object pose $T_f^{ns(c)}$

solvable $\leftarrow$ false
foreach $C_i \in \mathbb{C}$ **do**
  if $\exists$ separation(C_i) **then**
    $\mathbb{C} \leftarrow (\mathbb{C} - C_i) \cup \text{separation}(C_i)$
foreach *pair* $C_i, C_j \in \mathbb{C}$ **do**
  if $\exists$ combinationRule$(C_i \cup C_j)$ **then**
    $\mathbb{C} \leftarrow (\mathbb{C} - (C_i \cup C_j)) \cup \text{combinationRule}(C_i \cup C_j)$
  if $\exists$ manifoldMap$(\mathbb{C})$ **then**
    solvable $\leftarrow$ true
    break
if *solvable* **then**
  manifold $\leftarrow$ manifoldMap(C)
  if separable(manifold) **then**
    $R_f^{ns(c)} \leftarrow \text{project}(R_f^c) * (R_f^c)^T$
    $t_f^{ns(c)} \leftarrow \text{project}(t_f^c) - (\text{project}(R_f^c) * (R_f^c)^T * t_f^c$
    $T_f^{ns(c)} \leftarrow (R_f^{ns(c)}, t_f^{ns(c)})$
  else
    $T_f^{ns(c)} \leftarrow \text{geomProject}(T_f^c)$

projected to $T_f^{ns(c)}$ using the submanifold's projection function (geomProject defined in Table VI).

The algorithm for the exact geometric solver is explained in Algorithm 1. The most important steps, i.e., the constraint processing rules and solution synthesis, are explained next.

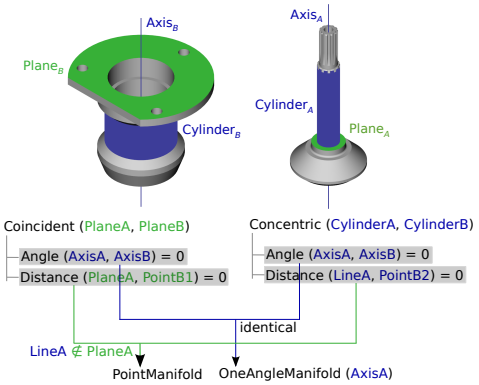


Fig. 4: Example of constraint combination rules for assembly.

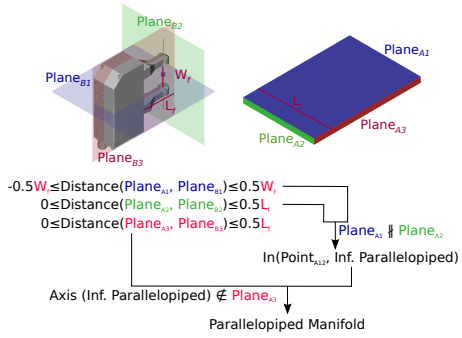


Fig. 5: Constraint combination rules for a task with inequality constraints (tray grasping)

A. Constraint processing rules

The simplification and combination of constraints is an important step in the constraint solving process. By separating complex constraints into a combination of simpler ones, the number of constraint combination scenarios and rules can be reduced. The combination rules for a constraint pair also perform a consistency check by analyzing their geometric properties and determining whether they are compatible or redundant. By combining constraints in this way, the number of mappings needed from constraint sets to transformation manifolds can also be reduced.

Some constraint simplification and combination rules have been presented in [1]. We created additional rules in this work to enable the combination of constraints with inequalities (Table VII). Also, for constraints where the rotation and translation sub-manifolds are dependent on each other (Table III), full transformation manifolds are required to represent their nullspace (Section IV-A).

After simplification, the constraints are recursively combined till a set of constraints that can be directly mapped to a transformation manifold is obtained (Table III).

Fig. 4 presents an example of constraint processing rules from an assembly use-case (Section VI-A), where both constraints are separable into rotation and translation components (Table III). The rotation components from both constraints are identical, and are mapped to a OneParallel rotation manifold. The translation components can be combined according to the case where Line_A is not contained in the Plane_A. The resulting translation constraint is a point-point coincident constraint, which is mapped to a Point translation manifold. Fig. 5 presents an example of constraint processing rules for a task that requires inequality constraints (see Section VI-B). More examples of constraint processing rules based on robotics use-cases have been illustrated in Section VI.

B. Solution synthesis

Once the transformation manifold has been determined, the closest operational pose for the robot is calculated by projecting the current pose onto the transformation manifold.

VI. Robotic Applications

Constraint-based approaches are effective at representing robot tasks, especially in the case of robots interacting with rigid objects of known geometry. We have shown how such geometric constraints can be defined using intuitive user interfaces in our previous work [5], [19]. For benchmarking our solving approach with other approaches (including our previous work [3], [5], [6]), we choose some common robotic applications and derive their constraint-based representations.

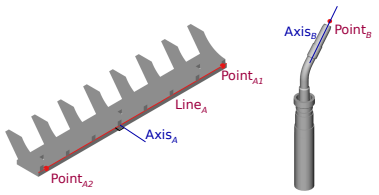
A. Assembly of two workpieces

In this scenario (Fig. 4), two workpieces from a gearbox need to be assembled together. The parameters for each of these steps can be specified using geometric constraints between the assembly objects (A and B, Fig. 4).

B. Tray grasping

This task involves grasping of a tray using a two-fingered gripper. The constraint-based definition of this task is shown in Fig. 5. ParallelDistance constraints between three sets of non-parallel Planes would, in the absence of inequalities, result in a fully specified robot pose. Hence, inequality constraints are necessary to model the nullspace of this task.

The rotation and translation components of the constraints are separable. Two ParallelDistance constraints between sets of non-parallel Planes generate an infinite Parallelepiped, whose infinite axis is the line of intersection of the two Planes. Three ParallelDistance constraints between sets of non-parallel Planes generate a finite Parallelepiped. The rotation in this case is fully defined.



Coincident(Point$_B$, Line$_A$)
$0 \leq$ Angle(Axis$_B$, Axis$_A$) $\leq \theta_{max}$
Coincident(Point$_B$, Point$_{A1}$) (start point)
Coincident(Point$_B$, Point$_{A2}$) (end point)

Fig. 6: Constraint for seam welding.

C. Seam welding

The seam welding task requires the tip of the welding tool (a point) to move along a specified path on the target object (e.g., a line in case of Fig. 6). This task can be expressed using a *line-point-coincident* constraint, where the translation of the point is restricted along the line but the rotation is free. Limits on the rotation of the tool with respect to the object can also be defined using inequality constraints, ensuring that the tool is oriented within the angular limitations required by the welding process.

D. Grasping of a soda can

This task involves the grasping of a cylindrical object (a soda can) from its sides (Fig. 1). This task can be described using a *line-line-distance-min-max* constraint, that results in a translation nullspace described by a *ThickCylinder*. To ensure proper grasping, the gripper plane also needs to be tangential to this cylinder, resulting in a *cylinder-plane-tangent* constraint. A combination of these constraints is a non-separable constraint, whose nullspace is described in Fig. 2g. To incorporate grasping possibilities along the length of the can and to utilize the gripper span and finger length, additional inequality constraints can be defined (Fig. 1).

E. Manipulating a tray carrying an assembly

This task presents an example with multiple objects. The assembly described in Section VI-A is placed on a tray. The assembly itself has one rotational DoF along its symmetrical axis in its nullspace. Additionally, the object can translate in two directions along the tray and rotate around its normal direction. The tray in turn is manipulated by a robotic arm and can translate in all 3 axes as well as rotate along the normal direction of tray plane. The other two rotational axes are restricted to prevent the object from falling off the tray. Since there are three consecutive sets of constraints, the solution is obtained by solving the problem in three steps. In total, this system has 4 constraints and 8 DoFs in the nullspace (see Fig. 7).

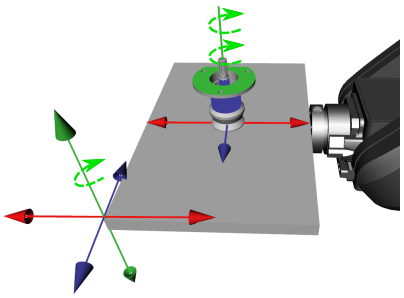


Fig. 7: Multi-object task: manipulating a tray that carries an assembled part (Section VI-A)

TABLE IV: Runtime evaluation on application scenarios

Scenario	#Constraints	Nullspace DoF	Iterative [6] Runtime (in ms)	Exact Geom. Runtime (in ms)
A. Plane Distance	1	3	0.6 ± 0.1	**0.05 ± 0.01**
B. Seam welding	1	3	0.8 ± 0.1	**0.06 ± 0.01**
C. Cylinder Grasp	4	1	0.8 ± 0.1	**0.06 ± 0.01**
D. Cup Grasp	4	1	1.2 ± 0.2	**0.06 ± 0.01**
E. Tray Grasp	3	0	3.0 ± 0.2	**0.06 ± 0.01**
F. Tray with object	4	8	N/A	**0.17 ± 0.01**

VII. Evaluation

We present quantitative evaluations of our approach in terms of solver runtimes for different tasks in the following Sections. In each of these evaluations, the mean and standard deviation of the runtime for our approach was calculated for 1000 repetitions from different random initial poses of the robot.

A. Runtime evaluation for different robotic tasks

To assess the time efficiency of our approach, we evaluated the solver runtimes for each application mentioned in this paper. The results are summarized in Table IV. The evaluation covers a wide variety of tasks having 1–4 constraints and 1–8 degrees-of-freedom in the task nullspace. The constraints include separable and mixed transformation manifolds. For one pair of objects, the runtime ranges from 0.05 ms for the simplest task to 0.06 ms for the most complicated one. We compare this to our previous work [6]. Since the other frameworks mentioned in Table I do not completely support the types of constraints used in our applications, they are skipped for this evaluation.

TABLE V: Runtime evaluation of control frameworks

Framework	Runtime 1 constraint (in ms)	Runtime 2 constraints (in ms)	Output
This work	**0.05 ± 0.01**	**0.06 ± 0.01**	$\boldsymbol{x}$
Somani et al. (NLOpt) [6]	0.54 ± 0.07	0.62 ± 0.08	$\boldsymbol{x}$
Somani et al. (GN solver) [5]	3.21 ± 0.70	3.89 ± 0.50	$\boldsymbol{x}$
This work+IK	**0.28 ± 0.03**	**0.29 ± 0.03**	$\boldsymbol{q}$
TSID [2], [3]	0.50 ± 0.10	0.50 ± 0.10	$\boldsymbol{q}$
WBCF [3], [4]	0.80 ± 0.10	0.80 ± 0.10	$\boldsymbol{q}$
Somani et al. (NLOpt) [6]	0.80 ± 0.10	0.90 ± 0.10	$\boldsymbol{q}$
Somani et al. (GN solver) [5]	4.00 ± 1.00	5.00 ± 0.80	$\boldsymbol{q}$

B. Comparison with other constraint-based approaches

Based on reference implementations from our previous works [3], [5], [6], we compare our proposed approach to TSID [2], WBCF [4], and our previous approaches in terms of average controller runtime and task errors. The first test case involves a *plane-plane-coincident* constraint where 3 DoF are fixed. The second test case has two *plane-plane-coincident* constraints, making 5 DoF fixed. The tolerance for task errors is set to 10^{-6}. The results are summarized in Table V.

Note that the frameworks TSID and WBCF generate robot torques $\boldsymbol{\tau}$ (Table I) and a forward dynamics step was added in our implementation [3] to obtain robot joint positions $\boldsymbol{q}$. [5] and [6] generate robot joint positions $\boldsymbol{q}$, but we modified our implementations so that they can also generate only target Cartesian positions $\boldsymbol{x}$. In this evaluation, we used the inverse kinematics approach from [20] to generate robot joint positions $\boldsymbol{q}$ from Cartesian positions $\boldsymbol{x}$ provided by the exact solver.

The timing for generating robot joint positions including the additional IK step is reported separately for this evaluation. This is because (1) many controllers directly support input in the form of end-effector Cartesian positions and (2) runtimes for IK can vary immensely depending on the kinematics of the robot and the IK algorithm (symbolic or iterative).

VIII. Conclusion

We have presented an approach for exactly solving geometric constraints by exploiting the geometric properties of their nullspaces. Qualitatively, the solver supports several features that are not supported by other state-of-art approaches (both exact and iterative). Quantitatively, the solving time (see Section VII-A) is good for hard real-time control applications. From Section VII-B, it can be seen that the exact approach is much faster than other comparable frameworks.

There are several limitations to our approach that can form directions for future work. Firstly, we have studied constraint definitions only between a small set of primitive shapes. Extending this to generalized and more complex descriptors such as B-Splines or Bezier curves, in order to support the full B-REP standard is not trivial. Secondly, the number of constraint combination rules required to cover all corner cases can be relatively large and cumbersome to implement. This is especially relevant for our approach since the addition of inequalities to the combination rules presented in [1] already significantly increased the number of combination cases.

Appendix

Table VI lists the geometric properties of some translation manifolds used in this work, including their parametric representation and projection functions that calculate the nearest point on the translation manifold from a given input point.

Table VII shows some constraint combination rules.

TABLE VI: Parametric representation and point projection operations for geometric shapes

Shape/submanifold	**Parametric representation**	**Projection of Point:** geomProject($\boldsymbol{u}$)
$\mathbb{R}^3$	$\boldsymbol{q}(s_1, s_2, s_3) = \boldsymbol{p} + s_1\hat{\boldsymbol{d}}_1 + s_2\hat{\boldsymbol{d}}_2 + s_3\hat{\boldsymbol{d}}_3$	$\boldsymbol{u}$
Point ($\boldsymbol{p}$)	$\boldsymbol{q} = \boldsymbol{p}$	$\boldsymbol{p}$
Line ($\boldsymbol{p}, \hat{\boldsymbol{a}}$)	$\boldsymbol{q}(s) = \boldsymbol{p} + s\hat{\boldsymbol{a}}$	$\boldsymbol{p} + ((\boldsymbol{u} - \boldsymbol{p}) \cdot \hat{\boldsymbol{a}})\hat{\boldsymbol{a}}$
Circle ($\boldsymbol{p}, \hat{\boldsymbol{n}}, r$)	$\boldsymbol{q}(\alpha) = \boldsymbol{p} + r(c_\alpha \hat{\boldsymbol{n}}_{\perp 1} + s_\alpha \hat{\boldsymbol{n}}_{\perp 1})$	$\boldsymbol{p} + r[\text{geomProject}_{\text{Plane}(\boldsymbol{p},\hat{\boldsymbol{n}})}(\boldsymbol{u}) - \boldsymbol{p}]$[1]
Plane ($\boldsymbol{p}, \hat{\boldsymbol{n}}$)	$\boldsymbol{q}(s_1, s_2) = \boldsymbol{p} + s_1\hat{\boldsymbol{n}}_{\perp 1} + s_2\hat{\boldsymbol{n}}_{\perp 2}$	$\boldsymbol{u} + ((\boldsymbol{p} - \boldsymbol{u}) \cdot \hat{\boldsymbol{n}})\hat{\boldsymbol{n}}$
Sphere ($\boldsymbol{p}, r$)	$\boldsymbol{q}(\alpha, \beta) = \boldsymbol{p} + r(c_\alpha c_\beta \hat{\boldsymbol{d}}_1 + s_\alpha s_\beta \hat{\boldsymbol{d}}_2 + c_\beta \hat{\boldsymbol{d}}_3)$	$\boldsymbol{p} + r[\boldsymbol{u} - \boldsymbol{p}]$[1]
Cylinder ($\boldsymbol{p}, \hat{\boldsymbol{a}}, r, h$)	$\boldsymbol{q}(s, \alpha) = \boldsymbol{p} + s\hat{\boldsymbol{a}} + r(c_\alpha \hat{\boldsymbol{a}}_{\perp 1} + s_\alpha \hat{\boldsymbol{a}}_{\perp 1})$	$\text{geomProject}_{\text{Plane}(\boldsymbol{p},\hat{\boldsymbol{n}})}(\boldsymbol{u}) + r[\boldsymbol{u} - \text{geomProject}_{\text{Plane}(\boldsymbol{p},\hat{\boldsymbol{n}})}(\boldsymbol{u})]$[1]
Box ($\boldsymbol{p}, \boldsymbol{d}, \boldsymbol{l}$)	$\boldsymbol{q}(s_1, s_2, s_3) = \boldsymbol{p} + s_1\hat{\boldsymbol{d}}_1 + s_2\hat{\boldsymbol{d}}_2 + s_3\hat{\boldsymbol{d}}_3, s_i \in [-l_i/2, l_i/2]$	
Parallelepiped ($\boldsymbol{p}, \boldsymbol{n}, \boldsymbol{l}$)	$\boldsymbol{q}(s_1, s_2, s_3) = \boldsymbol{p} + s_1\hat{\boldsymbol{n}}_1 + s_2\hat{\boldsymbol{n}}_2 + s_3\hat{\boldsymbol{n}}_3, s_i \in [-l_i/2, l_i/2]$	
ThickCylinder($\boldsymbol{p}, \hat{\boldsymbol{a}}, r, h, w$)	$\boldsymbol{q}(s_1, s_2, \alpha) = \boldsymbol{p} + s_1\hat{\boldsymbol{a}} + (r + s_2)(c_\alpha \hat{\boldsymbol{a}}_{\perp 1} + s_\alpha \hat{\boldsymbol{a}}_{\perp 1})$	

[1] vectors enclosed in [] are considered normalized

TABLE VII: Constraint combination rules[1]

Constraint 1	**Constraint 2**	**Combination condition**	**Combined constraint**
$\text{Coinc}(\text{Plane}_1^f, \text{Point}_1^c)$	$\text{Coinc}(\text{Plane}_2^f, \text{Point}_2^c)$	$\text{Plane}_1^f \nparallel \text{Plane}_2^f$, $\text{Point}_1^c = \text{Point}_2^c$	$\text{Coinc}(\text{Line}_{12}^f, \text{Point}_1^c)$
$\text{Dist}(\text{Plane}_1^f, \text{Point}_1^c) \in [\text{d1}_{\min}, \text{d1}_{\max}]$	$\text{Dist}(\text{Plane}_2^f, \text{Point}_2^c) \in [\text{d2}_{\min}, \text{d2}_{\max}]$	$\text{Plane}_1^f \nparallel \text{Plane}_2^f$, $\text{Point}_1^c = \text{Point}_2^c$	$\text{In}(\text{Inf.Ppd}_{12}^f, \text{Point}_1^c)$
$\text{Dist}(\text{Plane}_1^f, \text{Point}_1^c) \in [\text{d}_{\min}, \text{d}_{\max}]$	$\text{In}(\text{Inf.Ppd}_2^f, \text{Point}_2^c) \in [\text{d}_{\min}, \text{d}_{\max}]$	$\text{Ppd}_2^f \notin \text{Plane}_1^f$, $\text{Point}_1^c = \text{Point}_2^c$	$\text{In}(\text{Ppd}_{12}^f, \text{Point}_1^c)$
$\text{Coinc}(\text{Plane}_1^f, \text{Point}_1^c)$	$\text{Coinc}(\text{Line}_2^f, \text{Point}_2^c)$	$\text{Line}_2^f \cup \text{Plane}_1^f$, $\text{Point}_1^c = \text{Point}_2^c$	$\text{Coinc}(\text{Point}_{12}^f, \text{Point}_1^c)$
$\text{Dist}(\text{Plane}_1^f, \text{Point}_1^c) \in [\text{d1}_{\min}, \text{d1}_{\max}]$	$\text{Dist}(\text{Line}_2^f, \text{Point}_2^c) \in [\text{d2}_{\min}, \text{d2}_{\max}]$	$\text{Line}_2^f \cup \text{Plane}_1^f$, $\text{Point}_1^c = \text{Point}_2^c$	$\text{In}(\text{Obl.Cyl.}_{12}^f, \text{Point}_1^c)$

[1] Ppd = Parallelepiped, Inf.Ppd = Infinite Parallelepiped, Obl.Cyl = Oblique Cylinder

REFERENCES

[1] A. Rodriguez, L. Basaez, and E. Celaya, "A relational positioning methodology for robot task specification and execution," *IEEE Transactions on Robotics*, vol. 24, no. 3, pp. 600–611, June 2008.

[2] A. D. Prete, F. Nori, G. Metta, and L. Natale, "Prioritized motion-force control of constrained fully-actuated robots: "Task space inverse dynamics"," *Robotics and Autonomous Systems*, vol. 63, pp. 150–157, Jan. 2015.

[3] C. Cai, N. Somani, M. Rickert, and A. Knoll, "Prioritized motion-force control of multi-constraints for industrial manipulators," in *Proc. of the IEEE Intl. Conf. on Robotics and Biomimetics*, Zhuhai, China, Dec. 2015.

[4] O. Khatib, L. Sentis, J. Park, and J. Warren, "Whole-body dynamic behavior and control of human-like robots," *International Journal of Humanoid Robotics*, vol. 1, no. 1, pp. 29–43, Mar. 2004.

[5] N. Somani, A. Gaschler, M. Rickert, A. Perzylo, and A. Knoll, "Constraint-based task programming with CAD semantics: From intuitive specification to real-time control," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Sep. 2015, pp. 2854–2859.

[6] N. Somani, A. Perzylo, A. Gaschler, C. Cai, M. Rickert, and A. Knoll, "Task level robot programming using prioritized non-linear inequality constraints," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Oct. 2016.

[7] B. Bettig and C. M. Hoffmann, "Geometric constraint solving in parametric computer-aided design," *Journal of Computing and Information Science in Engineering*, vol. 11, no. 2, pp. 1–26, June 2011.

[8] N. Somani, A. Perzylo, C. Cai, M. Rickert, and A. Knoll, "Object detection using boundary representations of primitive shapes," in *Proc. of the IEEE Intl. Conf. on Robotics and Biomimetics*, Dec. 2015, pp. 108–113.

[9] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *Int. Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, Oct. 2011.

[10] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal of Robotics and Automation*, vol. 3, no. 1, pp. 43–53, Feb. 1987.

[11] J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 433–455, May 2007.

[12] C. Lenz, M. Rickert, G. Panin, and A. Knoll, "Constraint task-based control in industrial settings," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Oct. 2009, pp. 3058–3063.

[13] T. Kröger, B. Finkemeyer, and F. M. Wahl, "Manipulation primitives — A universal interface between sensor-based motion control and robot programming," in *Robotic Systems for Handling and Assembly*, ser. Springer Tracts in Advanced Robotics. Springer, 2011, vol. 67, pp. 293–313.

[14] T. De Laet, S. Bellens, R. Smits, E. Aertbelien, H. Bruyninckx, and J. De Schutter, "Geometric relations between rigid bodies (Part 1): Semantics for standardization," *IEEE Robotics Automation Magazine*, vol. 20, no. 1, pp. 84–93, Mar. 2013.

[15] G. Borghesan, E. Scioni, A. Kheddar, and H. Bruyninckx, "Introducing geometric constraint expressions into robot constrained motion specification and control," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1140–1147, July 2016.

[16] E. Aertbelien and J. De Schutter, "eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Sept. 2014, pp. 1540–1546.

[17] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, June 2014.

[18] A. Perzylo, N. Somani, M. Rickert, and A. Knoll, "An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions," in *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, Sept. 2015, pp. 4197–4203.

[19] S. Profanter, A. Perzylo, N. Somani, M. Rickert, and A. Knoll, "Analysis and semantic modeling of modality preferences in industrial human-robot interaction," in *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, Sept. 2015, pp. 1812–1818.

[20] P. Beeson and B. Ames, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics," in *Proc. of the IEEE-RAS Intl. Conf. on Humanoid Robots*, Nov. 2015, pp. 928–935.