Constraint-Based Task Programming with CAD Semantics: From Intuitive Specification to Real-Time Control

Nikhil Somani, Andre Gaschler, Markus Rickert, Alexander Perzylo, and Alois Knoll

***Abstract*— In this paper, we propose a framework for intuitive task-based programming of robots using geometric inter-relational constraints. The intended applications of this framework are robot programming interfaces that use semantically rich task descriptions, allow intuitive (re-)programming, and are suitable for non-expert users typically found in SMEs. A key concept in this work is the use of CAD semantics to represent geometric entities in the robotic workcell. The robot tasks are then represented as a set of geometrical inter-relational constraints, which are solved in real-time to be executed on the robot. Since these constraints often specify the target pose only partially, the robot can be controlled to move in the constraints' null space in order to handle external disturbances or further optimize the robot's pose during runtime. Geometrical inter-relational constraints are easy to understand and can be intuitively specified using CAD software. A number of applications common in industrial robotic scenarios have been chosen to highlight the advantages of the presented approach vis-à-vis the state-of-the-art approaches.**

I. Introduction

Current industrial robotic solutions are optimized for large production sizes and structured environments. They require the end-user to have expertise in robotics and cannot easily be re-programmed. This design is not suitable for small and medium sized enterprises (SMEs), where environments are largely unstructured and product cycles are short. Our aim is to enable a robotics non-expert to use the robotic system. Hence, an important aspect that we consider for the design of our framework is the level of intuitiveness in robot task-specification.

Most robot tasks do not have very strictly defined goals (Fig. 2), e.g., pick-and-place tasks where the robot grasps a cylindrical object at its rim (Fig. 2c). There are multiple target poses that can accomplish this task, and there is even more redundancy when these poses are mapped to robot configurations. From the viewpoint of intuitiveness, it is important for non-expert users to be able to specify tasks by referring to manipulation objects and their properties rather than domain-specific knowledge such as poses, Euler angles, or robot configurations (Fig. 1).

Geometric constraints are often used in CAD software to describe relations between parts in an assembly. This includes constraints such as the distance between two surfaces, the angle between two lines, or the matching of axes of two holes. In our proposed approach, geometric inter-relational constraints can be used to specify high-level robot tasks in an intuitive, robot-independent way. Their mapping and execution on the low-level control of a robot system is an important aspect presented in this work. In order to enable the low-level robot control system to understand information at this abstract level, we need semantic descriptions that describe the objects, the goals, the robot system, the environment, and common and domain-specific knowledge. Using these models, the semantic task-specific constraints are translated to constraints on the robot's pose.

All authors are with fortiss GmbH, An-Institut Technische Universität München, Munich, Germany. Correspondence should be addressed to somani@fortiss.org

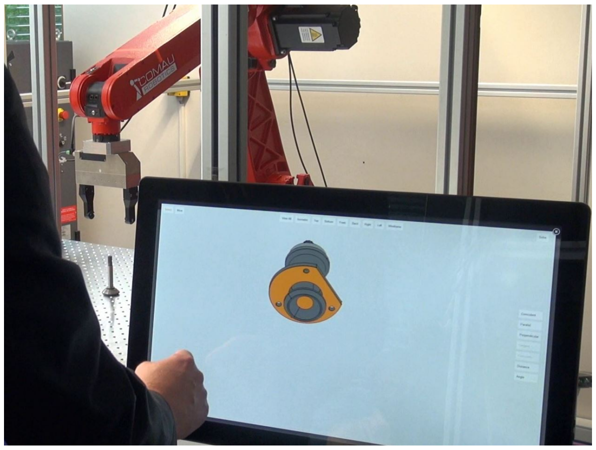
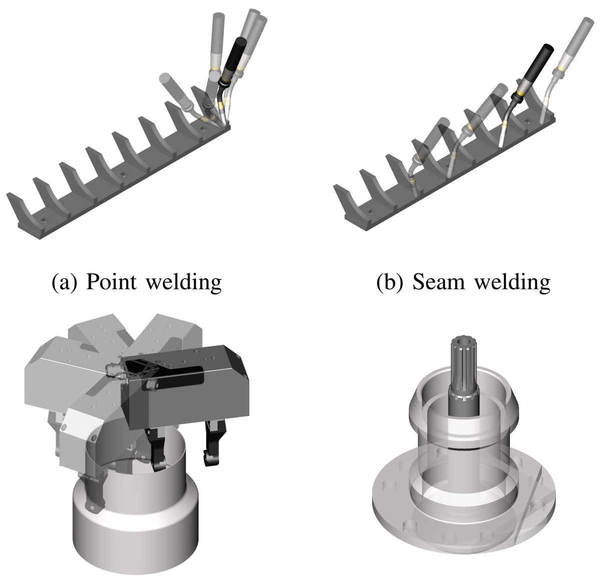


Fig. 1: Intuitive interface that allows users to specify semantically rich descriptions of robot tasks using geometric constraints.

There are several standard and well-understood approaches for controlling robots to fulfill tasks such as simple positioning, trajectory following or pick-and-place operations in structured or controlled environments. Some of these tasks are even supported by the standard commercial robot control software packages provided by robot manufacturers. However, these approaches focus on a complete specification of the robot's target poses and trajectories, rather than a semantic definition of the task. This makes it difficult and computationally expensive to re-calculate robot poses based on variations in the scene during task execution.

The proposed framework is based on a composable structure where several constraints, each describing a robot task or behaviour, can be combined. The framework supports several types of constraints such as temporal (e.g., picking before placing), spatial (e.g., reachability, collision avoidance) or logical (e.g., required objects present) constraints. Some constraints arise from the task definition (e.g., assembly 'mating' constraints), others from aspects of the robot and the workcell (e.g., obstacles, joint limits). These constraints



(a) Point welding

(b) Seam welding

(c) Grasping a cylindrical object at its rim

(d) Assembly of two objects

Fig. 2: Specification of geometric inter-relational constraints for different use-cases

can be specified at the pose or velocity levels in configuration space ($\boldsymbol{q}$) and operational space ($\boldsymbol{x}$). Using the null-space of geometric constraints, the robot pose is further optimized to satisfy the constraints posed by the robot's kinematics and the workcell chosen for execution.

II. Related Work

The essential elements of our work are inspired by ideas such as constraint-task based control [1], operational space control [2], instantaneous constraint-based task specification framework [3], robot skills definitions [4], geometric constraint solving for CAD models [5] and intuitive interfaces for robot programming [6], [7].

In [4], [8], the authors presented an approach for re-useable hardware-independent skills for robots. Rather than using CAD semantics, task-based programming approaches may use 3D pointing devices or other novel interfaces to program robots intuitively [9], [10], [6].

An approach for representing coordinate frames and geometric relations between them, along with an approach for using it in robotic applications was presented in [11], [12].

The instantaneous constraint-based task specification framework (iTaSC) [3] defined robot tasks using a definition of reference frames for the objects and features. It also featured an approach for representing and handling geometric uncertainties as a part of the task specification. In [13], the authors presented an extension of iTaSC that supports non-instantaneous task specifications and inequality constraints. In [14], the authors presented an alternative to iTaSC that separates task specification and execution concerns.

The idea of operational space control and its extensions for redundant robots (e.g., pose optimization) was introduced in [2] and extended for multiple end effectors and whole body control in [15]. In [16], the authors presented an approach where different manipulation primitives can be specified for different robot axes and combined together in the control framework. A hierarchical constraint-based task specification framework (similar to [15]) was presented in [1], where task definitions need not be specific to one robot axis and priorities between tasks can be defined. In this approach, the solutions from lower priority tasks are propagated to higher priority tasks, where they are merged by projection into the null-space of the higher priority task.

Our approach for intuitive robot programming presents three important contributions with respect to the state of art. Firstly, we present an approach that follows semantic definitions (using ontologies) from CAD-based representation of geometries, to robot execution and control. Hence, robot tasks can be defined at a semantic level using intuitive geometric constraints between geometrical entities and translated into low-level constraints on the robot's pose. Secondly, we go one step further than frame-based definitions of robot tasks [11], [12] and define constraints between the geometrical entities (e.g., points, curves, surfaces) that compose the manipulation objects and the robotic system. Finally, we propose to have a combination of the approaches from [2], [15], [1] and [3], [13], where several constraints can be minimized together ***and*** priorities between tasks can be specified and handled using the null-spaces of constraints.

III. Overview

We choose 4 typical industrial robotic scenarios for demonstrating and evaluating the proposed approach (Fig. 2): (a) Point welding scenario, where the robot is supposed to weld an object at a user-specified point. This task fixes the position of the welding tooltip (Fig. 4). However, its orientation is not fixed and can be optimized during runtime (b) Seam-welding scenario, where the welding tooltip should move along a user-specified line on the welding workpiece (Fig. 4). Not only the orientation of the welding tool but also its position along this line lies in the null-space of the constraint. (c) Grasping scenario, where a cylindrical object should be grasped at its rim (Fig. 5). Any point along the object's rim is a valid grasping pose. The orientation of the gripper is adjusted in a way that it is tangential to the cylinder's rim. (d) Assembly scenario, where constraints specified between the manipulation objects determine their relative poses in the final assembled object (Fig. 6).

The tasks for all chosen scenarios can be expressed in terms of geometric inter-relational constraints between the manipulation objects. Furthermore, they are underspecified tasks and the robot's pose can be optimized within the null-space of these constraints at runtime.

IV. Modeling of Geometric Constraints

The distinction between task-level constraints and workcell-specific constraints is an important concept in our approach. Hence, geometric constraints are modelled at different levels: task constraints at the semantic level

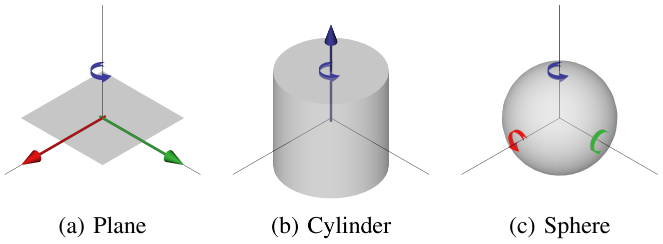
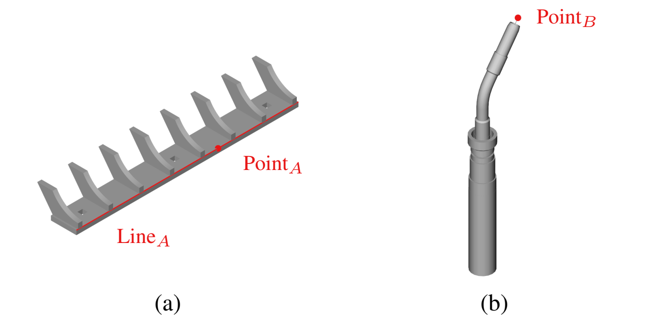


Fig. 3: Geometric constraints enforced by primitive shapes: Unconstrained axes (with x-axis in *red*, y-axis in *green*, and z-axis in *blue*) are indicated by arrows.



User	Solver
Point Welding	
Coincident(Point$_A$, Point$_B$)	Coincident(Point$_A$, Point$_B$)
Seam Welding	
Coincident(Line$_A$, Point$_B$)	Coincident(Line$_A$, Point$_B$)

Fig. 4: Geometric constraints for point and seam welding

(Sec. IV-A), constraints on the robot's pose (Sec. IV-B) and workpiece pose constraints (Sec. IV-D). In order to facilitate the combination and transformation of constraints between these levels, we use semantic descriptions (Sec. IV-C).

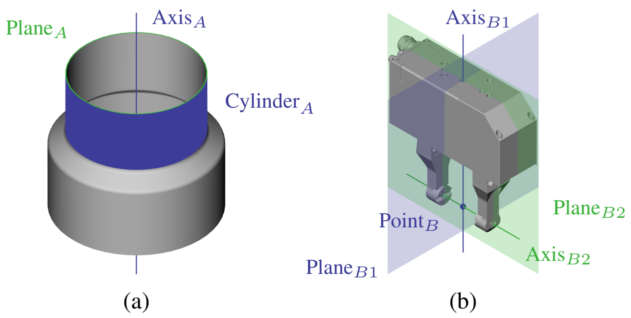
A. Geometric Constraints at Task Description Level

Robot tasks are expressed as geometric constraints between the manipulation objects and may be underspecified. Typical underspecified tasks for the applications targeted in this paper are illustrated in Fig. 2, where each robot pose satisfies the constraints required by the scenario. The task specification defines *user* constraints that are based on only the geometric properties of the manipulation objects (Fig. 3), which are then translated into constraints on the robot's pose (*solver* constraints). Figs. 4, 5 and 6 illustrate these *user* and *solver* constraints for the scenarios presented in Fig. 2.

B. Mathematical Models for Geometric Constraints

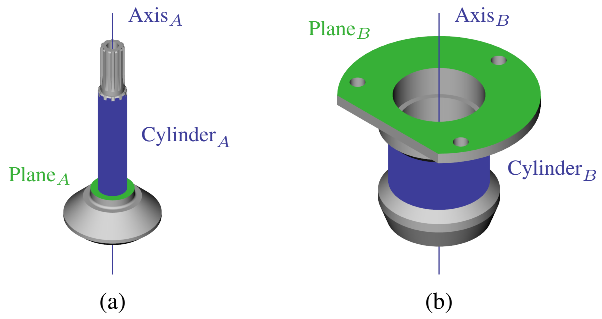
A rigid non-articulated object (workpiece, robot end-effector) can be decomposed into a collection of primitive geometric shapes in a particular configuration. Constraints on each of these geometric shapes add constraints on the overall pose of the object.

Each primitive shape P_i enforces a set of constraints $(\boldsymbol{S_{ti}}, \boldsymbol{S_{Ri}})$ on the position ($\boldsymbol{t}$) and orientation ($\boldsymbol{R}$) of the object respectively. Each primitive shape type also specifies



User	Solver
Tangent(Cylinder$_A$, Plane$_B$)	Coincident(Axis$_A$, Plane$_{B2}$)
Cylinder$_A$ ∥ Axis$_B$	Axis$_A$ ∥ Axis$_{B1}$
Tangent(Cylinder$_A$, Axis$_{B1}$)	Distance(Axis$_A$, Point$_B$) = radius(Cylinder$_A$)
Coincident(Plane$_A$, Axis$_{B2}$)	Distance(Plane$_A$, Axis$_{B2}$) = 0

Fig. 5: Geometric constraints for grasping a cylindrical object at its rim



User	Solver
Concentric(Cylinder$_A$, Cylinder$_B$)	Coincident(Axis$_A$, Axis$_B$)
Coincident(Plane$_A$, Plane$_B$)	Distance(Plane$_A$, Plane$_B$) = 0

Fig. 6: Geometric constraints for the assembly of two workpieces

its null-spaces (or free-spaces). Each row of $\boldsymbol{S_{ti}}$ and $\boldsymbol{S_{Ri}}$ contains a direction along which the constraint has been set. Examples of constraints set by each primitive shape are shown in Fig. 3 and explained below:

- Plane (point($\boldsymbol{p}$), normal($\boldsymbol{n}$)): $\boldsymbol{S_t} = [\boldsymbol{n}]$, $\boldsymbol{S_R} = [\boldsymbol{n}_{\perp 1}; \boldsymbol{n}_{\perp 2}]$ (where $\boldsymbol{n}_{\perp 1}$ and $\boldsymbol{n}_{\perp 2}$ are $\perp$ to $\boldsymbol{n}$).
- Cylinder (point($\boldsymbol{p}$), normal($\boldsymbol{n}$), radius(r)): $\boldsymbol{S_t} = [\boldsymbol{n}_{\perp 1}; \boldsymbol{n}_{\perp 2}]$, $\boldsymbol{S_R} = [\boldsymbol{n}]$.
- Sphere (point($\boldsymbol{p}$), radius(r)): $\boldsymbol{S_t} = [\boldsymbol{n}_{\perp 1}; \boldsymbol{n}_{\perp 2}; \boldsymbol{n}]$, $\boldsymbol{S_R} = \phi$

Geometric constraints can be specified between different types of geometrical entities. Table I lists the constraints that are most relevant to our target scenarios, and are supported by our implementation.

C. Semantic Description of Geometric Constraints

An important aspect of this work is the representation of all entities at a semantic level, using the Web Ontology Language (OWL) which is a standard supported by the World Wide Web Consortium (W3C).

TABLE I: Summary of supported geometric constraints

Fixed	**Constrained**	**Constraint** (C)	**Controlled Space** ($\boldsymbol{S}$)	**Null Space** ($\boldsymbol{N}$)	**Cost Function** ($\mathrm{Cost_C}$)
Plane_1	Plane_2	Distance (d)	$\boldsymbol{S_R}:\ [\boldsymbol{n}_{1_{\perp 1}};\boldsymbol{n}_{1_{\perp 2}}]$ $\boldsymbol{S_t}:\ [\boldsymbol{n}_1]$	$\boldsymbol{N_R}:\ [\boldsymbol{n}_1]$ $\boldsymbol{N_t}:\ [\boldsymbol{n}_{1_{\perp_1}};\boldsymbol{n}_{1_{\perp_2}}]$	$[\boldsymbol{n}_1^{\mathrm{T}}(\boldsymbol{p}_1-\boldsymbol{p}_2);\boldsymbol{n}_2^{\mathrm{T}}(\boldsymbol{p}_1-\boldsymbol{p}_2);1-\boldsymbol{n}_1^{\mathrm{T}}\boldsymbol{n}_2]$
Plane_1	Plane_2	Parallel	$\boldsymbol{S_R}:\ [\boldsymbol{n}_{1_{\perp 1}};\boldsymbol{n}_{1_{\perp 2}}]$ $\boldsymbol{S_t}:\ []$	$\boldsymbol{N_R}:\ [\boldsymbol{n}_1]$ $\boldsymbol{N_t}:\ \mathbf{1}$	$[[\boldsymbol{n}_{1_{\perp 1}};\boldsymbol{n}_{1_{\perp 2}}]^{\mathrm{T}}\boldsymbol{n}_2]$
Line_1	Line_2	Parallel	$\boldsymbol{S_R}:\ [\boldsymbol{n}_{1_{\perp 1}};\boldsymbol{n}_{1_{\perp 2}}]$ $\boldsymbol{S_t}:\ [\boldsymbol{n}_{1_{\perp 1}};\boldsymbol{n}_{1_{\perp 2}}]$	$\boldsymbol{N_R}:\ [\boldsymbol{n}_1]$ $\boldsymbol{N_t}:\ [\boldsymbol{n}_1]$	$[[\boldsymbol{n}_{1_{\perp 1}};\boldsymbol{n}_{1_{\perp 2}}]^{\mathrm{T}}\boldsymbol{n}_2]$
Line_1	Line_2	Coincident	$\boldsymbol{S_R}:\ [\boldsymbol{n}_{1_{\perp 1}};\boldsymbol{n}_{1_{\perp 2}}]$ $\boldsymbol{S_t}:\ [\boldsymbol{n}_{1_{\perp 1}};\boldsymbol{n}_{1_{\perp 2}}]$	$\boldsymbol{N_R}:\ [\boldsymbol{n}_1]$ $\boldsymbol{N_t}:\ [\boldsymbol{n}_1]$	$[[\boldsymbol{n}_{2_{\perp 1}};\boldsymbol{n}_{2_{\perp 2}}]^{\mathrm{T}}\boldsymbol{p}_1;1-\boldsymbol{n}_1^{\mathrm{T}}\boldsymbol{n}_2]$
Line_1	Point_2	Coincident	$\boldsymbol{S_R}:\ []$ $\boldsymbol{S_t}:\ [\boldsymbol{n}_{1_{\perp 1}};\boldsymbol{n}_{1_{\perp 2}}]$	$\boldsymbol{N}_R:\ \mathbf{1}$ $\boldsymbol{N_t}:\ [\boldsymbol{n}_1]$	$[[\boldsymbol{n}_{1_{\perp 1}};\boldsymbol{n}_{1_{\perp 2}}]^{\mathrm{T}}\boldsymbol{p}_2]$
Line_1	Point_2	Distance (d)	$\boldsymbol{S_R}:\ [\boldsymbol{n}_{1_{\perp 1}};\boldsymbol{n}_{1_{\perp 2}}]$ $\boldsymbol{S_t}:\ []$	$\boldsymbol{N_R}:\ [\boldsymbol{n}_1]$ $\boldsymbol{N_t}:\ [\boldsymbol{n}_1]$	$\boldsymbol{p}=[[\boldsymbol{n}_{1_{\perp 1}};\boldsymbol{n}_{1_{\perp 2}}]^{\mathrm{T}}(\boldsymbol{p}_1-\boldsymbol{p}_2)]$ $\mathrm{Cost_C}=(d-\lVert\boldsymbol{p}\rVert)\boldsymbol{p}$

We use an ontological description of CAD semantics, i.e., a boundary representation (BREP) of points, curves, surfaces, and volumes to describe the geometric entities in the robotic workcell (manipulation objects, robots and tools, workcell layout, etc.) [17]. The robot tasks are also semantically described [7], containing the associated manipulation objects and the corresponding geometric inter-relational constraints as their parameters.

D. *Geometric Constraints used for Object Recognition*

Often the objects used in robotic scenarios are symmetric, which leads to underspecified object poses. [18] presents an approach for estimation of underspecified object poses, which we extend to use the constraint formulations presented in this paper (Table I). When executing task constraints a workcell, the poses of manipulation objects are obtained from this object recognition module, which returns not only the underspecified poses but also their geometric null-spaces.

V. EXECUTION OF CONSTRAINT-BASED ROBOT TASKS

Upon instantiation of a task on a workcell, the ontological database is queried to obtain the set of task-specific geometric constraints and workcell specific manipulation constraints. These constraints need to be solved to generate poses and null-spaces which will be used for execution on the robot.

A. *Solver for Geometric and Kinematic Constraints*

In this section, we formally define the robot manipulation problem and then derive a mathematical optimization procedure, whose solution generates the robot configurations that form the waypoints of the final robot trajectory.

Given are a set $\mathcal{R}$ of kinematic structures and a set $\mathcal{O}$ of objects to be manipulated. A kinematic structure can be thought of as a robot with a tool; if a robot carries multiple tools, each branch forms one kinematic structure. A kinematic structure $\mathbf{R} \in \mathcal{R}$ is a tuple $(\mathrm{FK}, \mathbf{P})$, composed of a forward kinematic function FK that maps from an n-dimensional configuration space to the pose of its tool $\mathbb{R}^n \mapsto \mathrm{SE}(3)$ and a set of primitive shapes $\mathbf{P}$.

A primitive shape $P \in \mathbf{P}$ may be one of the shapes defined in Sec. IV-B and serves as a useful reference for geometric relations, e.g. the axis of a drill head, or the center of a parallel gripper. Analogous to kinematic structures, a manipulation object $\mathbf{O} \in \mathcal{O}$ is also composed of a configuration and a set of primitives shapes, given by a tuple $(\mathbf{x} \in \mathrm{SE}(3), \mathbf{P})$.

A manipulation problem is then defined by a set of constraints $\mathbf{C}$ that relate primitive shapes of both kinematic structures and objects. Unlike the object recognition component, where constraints refer to a single object and are amenable to a closed-form solution [18], we follow a more general minimization of a cost function for the robot manipulation problem. Each constraint $C \in \mathbf{C}$ defines a cost function $\mathrm{Cost} : \mathrm{SE}(3) \times \mathrm{SE}(3) \mapsto \mathbb{R}^c$ that depends on the poses of two shapes and yields a zero vector iff the constraint is fulfilled.

The general manipulation problem is defined in Eq. 1, with q denoting the stacked configuration space of all kinematics:

$$\min_q \sum_{\mathbf{R}\in\mathcal{R}} \sum_{C\in\mathbf{C}} \mathrm{Cost}_C^2\left(\mathrm{FK}_{\mathbf{R}}(q), \boldsymbol{x}\right) = \min_q |F(q)|^2 \quad (1)$$

The cost of a kinematic configuration can be viewed as a vector-valued function $F : \mathbb{R}^n \mapsto \mathbb{R}^c$ (Eq. 1). To solve for a kinematic configuration q, $F(q)$ is optimized in a non-linear least squares minimization. In the current implementation, we apply the simple Gauss-Newton method, which only requires the Jacobian $\boldsymbol{J}_{i,j} = \delta F_i/\delta q_j$ to be computed.

To allow a stable and efficient numerical implementation, cost functions must be designed such that, instead of scalar values, they return a cost vector $\mathbb{R}^c$, whose values reflect the c locked degrees-of-freedom and should in general be independent. Furthermore, costs of translational and rotational distances, where no obvious metric exists, should be weighted to the same order of magnitude. Only under these conditions, the null space can stably be computed to allow further optimization of the robot pose, distance to obstacles, and length of the trajectory. An appropriate set of cost functions is given in Table I.

B. *Using the Kinematic Null-space*

In almost all robot tasks, only a few degrees-of-freedom are required to solve an *exact* geometric task, while the rest is used to achieve a qualitative, lower-priority goal. Some of these lower-priority goals may be formulated as optimization targets, e.g., a robot posture far from singularities or joint limits, a waypoint close to a previous one for shorter trajectories, or to maximize the distance to obstacles. Other lower-priority goals may be specific to the domain and/or the scenario, e.g., optimization of angles for point welding or grasping a particular object, or tracking the control signal from visual servoing. It is clear that the null-space of the exact solution of the geometric constraint can be used for further offline optimization, automatic online control, or interactive jogging by an operator.

Similar to null-space control applications [1] that operate on a regular Jacobian, our generalized Jacobian of the kinematic cost function F allows simple computation of the *null-space projection* matrix $\boldsymbol{N}$.

$$\boldsymbol{N}(\boldsymbol{q}) = \boldsymbol{1} - \boldsymbol{J}^{\dagger}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q}) \tag{2}$$

This null-space projection maps an arbitrary control signal $\Delta\boldsymbol{q}$ from the kinematic domain to a robot motion orthogonal to the constraints $\mathcal{C}$ by multiplying $\boldsymbol{N}(\boldsymbol{q})\Delta\boldsymbol{q}$. Of course, $\boldsymbol{N}$ is only a local linearization valid for small $\Delta\boldsymbol{q}$, so larger steps should be applied through multiple iterations. $\Delta\boldsymbol{q}$ can represent all the above types of null-space optimization: for interactive jogging, $\Delta\boldsymbol{q}$ is set to a unit vector, for posture optimization, a repulsive force from the joint limits may be implemented. Likewise, $\Delta\boldsymbol{q}$ may represent a local disturbance from control frameworks, e.g., visual servoing.

VI. Evaluation

We evaluated our approach on a selection of classic industrial robotic scenarios in simulation and in real-world setups (using a 6DoF industrial robot). While the constraint resolution parts of our implementation are new, inverse kinematics, trajectory generation and low-level robot control use the Robotics Library[1] by Rickert [19]. A video of the implementation of these applications on the robotic workcell can be found at `http://youtu.be/qRJ1JmNoFEw`

A. Welding

In the point welding scenario (Fig. 2a), the tip of the welding gun must exactly coincide with the target point on the object (Fig.4). This is represented by task constraints:

- **Point-Point Coincident Constraint**: Point$_A$ of workpiece is coincident with Point$_B$ of welding gun tool.

In this example, the orientation of the welding gun should be adjusted by an operator familiar with point welding. The operator was provided a simple interface, to jog the robot in the 3 degrees-of-freedom null-space, while always fulfilling the constraint (Fig.7a).

In the seam welding scenario (Fig. 2b), the tip of the welding gun must lie on the target line on the object (Fig.4). The task constraints on the robot's pose are:

- **Point-Line Coincident Constraint**: Line$_A$ of workpiece is coincident with Point$_B$ of welding gun tool.

[1]`http://www.roboticslibrary.org/`

In this simplified experiment, the orientation of the welding gun is unconstrained. The operator could choose the orientation as well as the position along the line by jogging the robot in the null-space of the constraints, while the controller ensured that the constraints (Fig.7b) are always satisfied.

B. Grasping of cylindrical objects

In this scenario, an industrial manipulator is supposed to grasp a cylindrical object at its rim using a parallel gripper (Fig. 2c). This task can be defined purely with CAD constraints (Fig. 5), without any scenario-specific parameters. These constraints are then translated into the following constraints on the robot's pose:

- **Line-Plane Coincident Constraint**: Plane$_{B\perp}$, which is perpendicular to Plane$_B$ contains Axis$_A$
- **Line-Point Distance Constraint**: Point$_B$, which is the point of intersection of Axis$_{B1}$ and Axis$_{B2}$, is at a distance radius(Cylinder$_A$) from Axis$_A$
- **Plane-Line Distance Constraint**: Axis$_{B2}$ is at a distance 0 from Plane$_A$

While the above constraints need to be fulfilled exactly, a residual degree of freedom is available as a path along the rim of the cylinder (Fig. 7c). Note that this path cannot be solved by operational space constraints [1], but requires our more general formulation of geometric constraints $\mathcal{C}$. In this scenario (Fig. 7c), where we solve the constraints to obtain the target pose closest to the previous robot waypoint.

C. Assembly

In this scenario (Fig. 2d), two workpieces from a gearbox need to be assembled together. This task consists of 3 steps: (a) picking object A, (b) moving to object B, and (c) assembling the objects together. The parameters for each of these steps can be specified using geometric constraints between the assembly objects (A and B, Fig. 6) and the parallel gripper in Fig. 5 (B):

Pick Approach Constraints:

- **Line-Line Coincident Constraint**: Axis$_A$ is coincident with Axis$_{B1}$ of Gripper.
- **Plane-Plane Distance Constraint**: Plane$_A$ is at a distance $0.1m$ from Axis$_{B2}$ of Gripper.

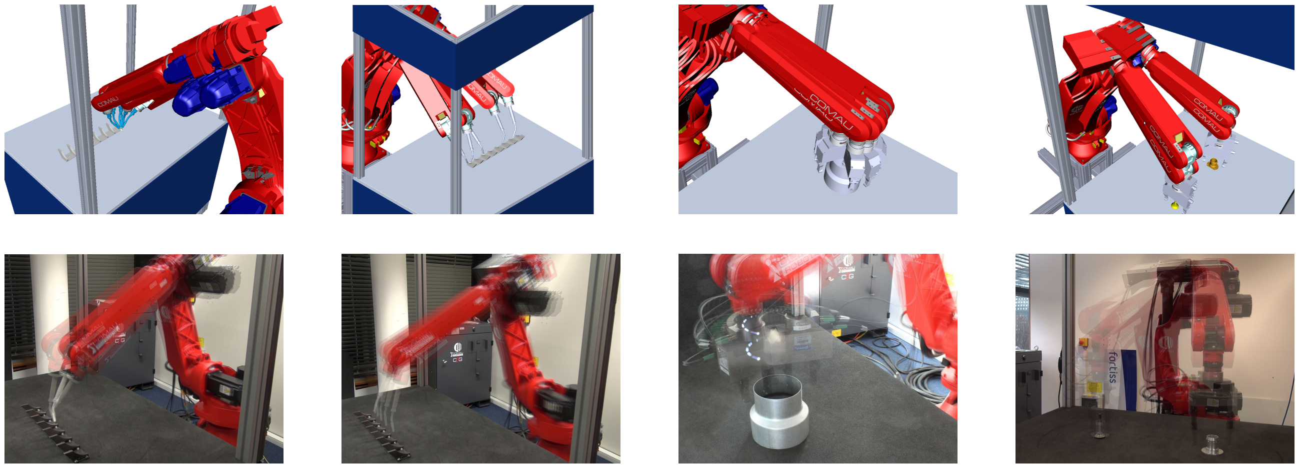
Place Approach Constraints:

- **Line-Line Coincident Constraint**: Axis$_B$ is coincident with Axis$_{B1}$ of Gripper.
- **Plane-Plane Distance Constraint**: Plane$_B$ is at a distance $0.1m$ from Axis$_{B2}$ of Gripper.

Assembly Constraints:

- **Line-Line Coincident Constraint**: Axis$_A$ is coincident with Axis$_{B1}$.
- **Plane-Plane Distance Constraint**: Plane$_B$ is at a distance 0 from Plane$_B$.

Fig. 7d shows the execution of this scenario. The grasping and assembly poses of the objects are only partially defined, where the rotation along the Axis$_A$ and Axis$_B$ lies in the null-space. The robot can be jogged in the null-space to choose an orientation as required by other constraints from the workcell (e.g., reachability, singularities).



(a) Point welding (b) Seam welding (c) Grasping (d) Assembly

Fig. 7: Execution of constraint-based tasks in a robotic workcell. CAD semantics allow these tasks to be defined in terms of geometric constraints between features of robot tools and objects, allowing intuitive robot programming.

VII. Conclusion

This work presents an approach for semantically rich, intuitive definitions of under-specified robot tasks using geometric constraints. The mapping and execution of these task descriptions on the low-level robot controller is an important contribution of this paper. Furthermore, the task-based constraints can be combined with environment and robot constraints, by utilizing the null-space of geometric constraints. Thereby the robot pose can be optimized according to the scenario, while always fulfilling the task constraints.

Supporting more geometric constraint types and modeling environment constraints, e.g., collision avoidance could be future additions to this work.

References

[1] C. Lenz, M. Rickert, G. Panin, and A. Knoll, "Constraint task-based control in industrial settings," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA, 2009, pp. 3058–3063.

[2] 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.

[3] 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, 2007.

[4] R. H. Andersen, T. Solund, and J. Hallam, "Definition and initial case-based evaluation of hardware-independent robot skills for industrial robotic co-workers," in *Proceedings of the International Symposium on Robotics*, June 2014, pp. 1–7.

[5] 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.

[6] N. Somani, E. Dean-Leon, C. Cai, and A. Knoll, "Scene perception and recognition for human-robot co-operation," in *Proceedings of the International Conference on Image Analysis and Processing, Workshop on Assistive Computer Vision and Robotics*, Sept. 2013.

[7] A. Perzylo, N. Somani, S. Profanter, M. Rickert, and A. Knoll, "Toward efficient robot teach-in and semantic process descriptions for small lot sizes," in *Proceedings of Robotics: Science and Systems (RSS), Workshop on Combining AI Reasoning and Cognitive Science with Robotics*, Rome, Italy, July 2015, http://youtu.be/B1Qu8Mt3WtQ.

[8] A. Bjorkelund, L. Edstrom, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. G. Robertz, D. Storkle, A. Blomdell, R. Johansson, *et al.*, "On the integration of skilled robot motions for productivity in manufacturing," in *Proceedings of the IEEE International Symposium on Assembly and Manufacturing*. IEEE, 2011, pp. 1–9.

[9] 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 International Conference on Intelligent Robots and Systems*, 2015.

[10] A. Gaschler, M. Springer, M. Rickert, and A. Knoll, "Intuitive robot tasks with augmented reality and virtual obstacles," in *Proceedings of the IEEE International Conference on Robotics and Automation*, June 2014, pp. 6026–6031.

[11] 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.

[12] T. De Laet, S. Bellens, H. Bruyninckx, and J. De Schutter, "Geometric relations between rigid bodies (part 2): From semantics to software," *IEEE Robotics Automation Magazine*, vol. 20, no. 2, pp. 91–102, June 2013.

[13] W. Decre, R. Smits, H. Bruyninckx, and J. De Schutter, "Extending iTaSC to support inequality constraints and non-instantaneous task specification," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2009, pp. 964–971.

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

[15] 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. 01, no. 01, pp. 29–43, 2004.

[16] 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, D. Schtz and F. Wahl, Eds. Springer, 2011, vol. 67, pp. 293–313.

[17] 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 International Conference on Intelligent Robots and Systems*, 2015.

[18] N. Somani, C. Cai, A. Perzylo, M. Rickert, and A. Knoll, "Object recognition using constraints from primitive shape matching," in *Proceedings of the International Symposium on Visual Computing*. Springer, Dec. 2014.

[19] M. Rickert, "Efficient motion planning for intuitive task execution in modular manipulation systems," Dissertation, Technische Universität München, 2011.