Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2019) 000–000

www.elsevier.com/locate/procedia

8th CIRP Conference of Assembly Technology and Systems

Leveraging Model Based Definition and STEP AP242 in Task Specification for Robotic Assembly

Shafi K. Mohammed[a,*], Mathias H. Arbo[b], Lars Tingelstad[a]

[a]*Dept. of Mechanical and Industrial Engineering, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway*
[b]*Dept. of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway*

Abstract

This article explores using aspects of STEP AP242 for constraint-based robot programming for assembly operations. Industry 4.0 envisions smart, connected factories where all the operations are connected by an unbroken thread of product and process data. As part of these efforts, many industries are adopting Model Based Definition and STEP AP242. STEP AP242 is an exchange format that allows Model Based Definition where Product Manufacturing Information is directly associated with the 3D CAD model. This article relates the geometric and assembly constraints from the CAD model to motion constraints on the robot during the assembly process. This article also discusses the use of Product Manufacturing Information from STEP AP242 files for automatic robot programming. The results are showcased with a prototype for a motor assembly scenario.

Keywords: Robotics; Assembly; Constraint-based Robot Programming; Model Based Definition; Product Manufacturing Information; STEP AP242; ISO 10303;

1. Introduction

The new industrial revolution, Industry 4.0, brings aspects of information technology into the manufacturing industries, with high product customization, information gathering cyber-physical systems, and a short time to deploy. The success of Industry 4.0 depends on the integration of design and manufacturing operations.

One of the key enablers for Industry 4.0 is the availability of product data to various stakeholders involved in the product life cycle process. The completeness of the available data affects the efficiency of product manufacturing operations both in terms of time and product quality.

A large amount of data is created in the initial phases of product development and design. The designers use CAD software to design a product and create 3D models. 2D manufacturing drawings are created from the 3D CAD models, and product manufacturing information (PMI) is added to the drawings. These manufacturing drawings are shared with the downstream operations, resulting in many communication gaps between the designer and the manufacturer that give rise to delays and quality issues.

Many downstream operations like inspection need manual intervention to study the 2D drawings, extract the necessary data, and recreate it to match their needs. The recreation of information is time-consuming and error-prone due to human involvement. In the scenario of Industry 4.0, the manual interpretation and recreation of data should be avoided as much as possible. The solution for this is reusing the data created in the product development stage in downstream manufacturing operations without recreation.

The recreation of information can be eliminated by employing model-based definition (MBD) in which PMI is directly appended to the 3D CAD model during the design phase. The critical dimensions, geometric dimensioning and tolerancing (GD&T), surface finish, and other needed information, are semantically added to the features of the 3D CAD model. This information is then available for direct use in downstream operations like NC machining and inspection [1, 2].

Traditionally there are three methods of programming robots: (1) teach-pendant programming, (2) offline programming, and (3) programming by demonstration. A limitation of these approaches is that they cannot be used for the automated programming of robot operations – the robots can be used to perform only pre-programmed tasks. They cannot be used to

* Corresponding author.
E-mail address: shafi.k.mohammed@ntnu.no (Shafi K. Mohammed).

control the manipulators to perform in new/unexpected scenarios. This severely limits the use of robots in small and medium enterprises that have a wide variety of products.

Future smart factories need intelligent robotic systems that can be used to complete various operations working in mixed environments with humans and other machines. One approach to solving this and overcome the limitations of traditional robot programming methods is constraint-based robot programming. In this approach, the robotic task is specified in terms of relations between components in the assembly, as well as with the environment, and not directly by point-to-point motion or other motion primitives. However, this requires that the robotic system is able to detect and extract the constraints involved in the task [3].

Integrating CAD in robot programming systems is a long-standing problem in robotics. Early systems capable of extracting relevant geometric information includes the Autopass, Archimedes 2, and HigLap frameworks.

Autopass is the earliest robot programming language based on CAD and focuses on the product being assembled, the tools, fixtures and the assembly tasks. An assembly world model is created from the geometric data and updated at each step of the assembly operation. The users plan the assembly sequence and specify the assembly operations as if they are preparing an assembly instruction manual without bothering about the actual motion of the manipulator [4].

Archimedes 2 is an assembly planning system that uses 3D CAD data to facilitate assembly planning and robot control. The 3D data from CAD files is used as an input to planning and simulation operations. The user can plan the assembly operations with the help of a planner and an illustrator, which simulates the operations. A translator converts these high-level assembly plans into control code for robotic assembly [5].

HighLap is an assembly planning system that automatically generates the assembly plans from the CAD data. It uses an assembly by disassembly method to decompose the assembly tasks into primitive robot skills [6, 7].

SMERobotics is an EU funded consortium aimed at enabling robotics for small to medium sized enterprises. This consortium presents a method for rapid robot programming based on mating constraints that are not extracted from the STEP or CAD file, but rather specified by a shop floor worker [8]. The part-oriented programming method showed good time-saving when compared to classical programming methods [9]. The system can plan assembly paths in the motion null-space formed by the mating constraints [10, 3].

More recent frameworks for constraint-based robot programming are the iTaSC [11] and eTaSL/eTC [12].

The iTaSC framework uses feature coordinates to define object constraints and uncertainty coordinates to estimate geometric uncertainties. This achieves instantaneous task specification for complex robotic applications with sensor interaction and geometric uncertainties [11, 13].

eTaSL is a task specification language, and eTC is a corresponding controller both based on expression graphs. Expression graphs are a tree-like data structure that specifies the geometric relations between objects and supports the computation of the Jacobians using automatic differentiation [12].

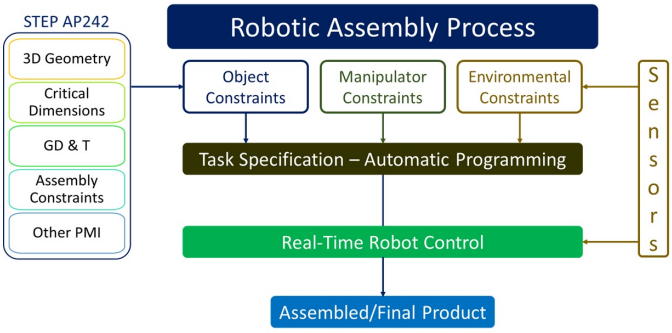

Fig. 1. Constraint-based robot programming using STEP AP242

In this paper, we explore the use of model-based definition and the use of the STEP AP242 exchange format for constraint-based robot programming for assembly operations. The product design information in the form of 3D CAD models (both the part and assembly) carries rich and useful information about the components and their constraints. The annotated models with the PMI enriches the geometric data of the 3D CAD and imparts meaning to these constraints. This data can be readily extracted and used to automate the programming and control of robots saving time and cost. The presented approach employs the eTaSL/eTC framework and is based on exporting the assembly constraints directly from the 3D CAD system and using these to generate the motion constraints of the robotic system automatically. A schematic presentation of our approach is shown in Figure 1.

The paper is structured as follows. Section 2 presents the STEP AP242 standard and how assembly constraints can be modelled and exported from 3D CAD data and Section 3 presents the eTaSL/eTC framework in more detail and shows how assembly constraints can be used to form motions constraints. A test case is presented in Section 4. A discussion of the presented approach is presented in Section 5. The paper is concluded in Section 6.

2. Extracting Assembly Constraint Information from STEP AP242 Exchange Files

The product data created during the design stage should be available to all the stakeholders in the downstream processes of the product life cycle phases. The 3D models can be shared with the downstream operations using native CAD files or neutral formats.

The standard for the Exchange of Product Model Data (STEP) is one of the standard neutral file formats used by the industry. It is a set of ISO (International Standards Organization) standards under ISO 10303. The two application protocols (APs) *STEP AP203 – Configuration controlled 3D designs of mechanical parts and assemblies* and *STEP AP214 – Core data for automotive mechanical design processes* are being used by aerospace and defense, and automotive industries, respectively. These two APs have overlapping scopes, and their latest edi-

tions started converging towards each other. Due to this, the application protocol *AP242 – Managed model based 3D engineering* was developed. This is a significant AP which combines and replaces AP203 and AP214 (along with few other APs under ISO 10303) [14, 15, 16].

STEP AP242 defines the use of semantic PMI for product definition. Under this standard, product data like GD&T, roughness specifications, welding details, direction features, contacting tangents, and hull features are readily available at both the part and assembly level.

As the industry is moving towards smart manufacturing, the adoption of STEP AP242 is increasing. AeroSpace and Defence Industries Association of Europe (ASD) recommends the use of latest editions of this standard for "exchange, long term archiving and transfer to downstream processes of CAD data (mechanical design, including composite) and associated configuration (PDM) data" [17].

2.1. Relevant Features of STEP AP242 Exchange Files

The following types of information are available in a STEP AP242 exchange file: Geometric data, assembly constraints, bill of materials, geometric dimensioning and tolerancing (GD&T) details, and other product manufacturing information (PMI) in the form of annotations.

Some of the information like the geometric details of the product are readily available in the STEP file but some information has to be inferred from the available data. For robotic assembly applications, geometric definition of the product alone is not much helpful. Product structure and assembly constraints are needed along with the geometric details.

2.1.1. Assembly Constraints

The types of assembly constraints available in STEP AP242 [15] are:

- *Fixed_constituent_assembly_constraint*: This is applied to the part which is fixed in the assembly and all other parts are placed with respect to this part. The coordinate system and origin of this part coincides with the world-coordinate system and origin of the assembly.
- *Parallel_assembly_constraint*: This is used to define the parallel relationship between the mating features of the parts. This is applicable to lines and planes.
- *Parallel_assembly_constraint_with_dimension*: This is a subtype of parallel_assembly_constraint with a 'distance_value' to define the distance between the parallel features.
- Surface_distance_assembly_constraint_with_dimension: This is used to define the normal distance between lines and planes of the mating parts.
- *Angle_assembly_constraint_with_dimension*: This defines the angle between the line and plane features.
- *Perpendicular_assembly_constraint*: This is used to apply the perpendicularity constraint between the mating lines and planes.
- *Incidence_assembly_constraint*: This is used when a line is incident on a plane or to define the concentricity of circular, cylindrical, conical or spherical features.
- *Coaxial_assembly_constraint*: This constraint is used when the axes of cylindrical or conical features have the same direction.
- *Tangent_assembly_constraint*: This defines the tangency of circular features with lines, planes or other circular features.

Table 1 shows all the assembly constraints available in STEP schema and the geometric features related by these constraints.

The assembly constraints available in the commercial CAD software correspond to these types of constraints. However, even though the STEP AP242 standards supports assembly constraints, when the assembly is exported as a STEP file from a CAD software, e.g., SolidWorks or Siemens NX, the constraints are not explicitly available in the STEP file.

2.1.2. Bill of Materials

The bill of materials (BoM) is beneficial information in understanding the product structure. It gives the number of components used in the assembly and how many times each component is used. BoM can be explicitly added to the assembly as a table in the annotations.

2.1.3. GD&T and Other PMI

GD&T and other manufacturing information can be added to the part file in the form of annotations, which increases the understanding of the product and helps in creating robust automated robot programming systems. In many assemblies, the mating fits can be inferred from the GD&T, which helps in understanding the force constraints on the robot during the assembly operation.

2.2. Extracting Constraint Information

The required constraint information is extracted from the STEP file and passed to the eTaSL framework for task specification. The constraint information includes parts in the assembly, the relative position of the parts to each other, assembly constraints, mating features of the parts, and their positions. The steps involved in this process are:

1. Identifying the parts in the assembly: The total number of parts and their instances are identified from the STEP file. This gives the overall product structure. Alternately this can be identified from the BoM if it is added as a table in the annotations.
2. Establishing the global coordinate frame: This is the coordinate frame to which the entire assembly is defined. By default this coordinate frame is defined at the origin.
3. Identifying the fixed constituent part: This is the immovable part in the assembly to which all other parts are placed in the assembly. Generally, the part coordinate frame of the fixed part coincides with the global coordinate frame of the assembly.

Table 1. Assembly Constraints and applicable geometric entities.

Assembly Constraint Entity	Line	Plane	Cylindrical/Conical/Spherical Surfaces
Parallel_assembly_constraint	*	*	
Parallel_assembly_constraint_with_dimension	*	*	
Surface_distance_assembly_constraint_with_dimension	*	*	
Angle_assembly_constraint_with_dimension		*	
Perpendicular_assembly_constraint	*	*	
Incidence_assembly_constraint	*		*
Coaxial_assembly_constraint			*
Tangent_assembly_constraint	*	*	*

4. Get the next part in the assembly: It is assumed that the assembly sequence is given by the order in which the parts are added to the assembly or the order in which the parts appear in the STEP file.
5. Identify the part coordinate frame: This coordinate frame gives the relative position of each part in the assembly. All the part features are defined with respect to this coordinate frame.
6. Find the assembly constraints: Identify all the assembly constraints applied on a part.
7. Establish feature coordinate frames: For each assembly constraint, find the corresponding mating features. Establish appropriate coordinate frames for these mating features. For simple surfaces, these can be directly identified from the STEP file. These feature coordinate frames are used in eTaSL to define the robotic tasks.
8. For all the parts in the assembly, repeat steps 4 to 7.

3. eTaSL/eTC for Assembly Constraints

eTaSL [12] is a Lua-based task specification language, eTC is a simultaneous hierarchical task controller realization for the task specifications. The controller works by inverting the differential kinematics of the constraints using a quadratic optimization program, which is solved for joint velocity commands that are sent to the robot. In the cases where a joint velocity command is not available, the joint velocities are integrated to achieve joint position commands.

3.1. Task Specification

The task specification is a Lua-based eTaSL script. A single eTaSL script defines a set of tasks, robot variables, support for continuous sensor inputs, event triggering when sensor or controller conditions are achieved, and continuous state outputs for monitoring and debugging. eTaSL defines tasks where the task equations are either driven to zero or driven to remain within an upper or lower limit. The set-based tasks enable defining work-cell related tasks such as remaining within a workspace or keeping the robot in an elbow up configuration. For more complex task specifications, eTaSL also supports feature variables that are controller-internal and sensor inputs. This allows for sensor-guided motion, such as compliant assembly strategies [18], and sensor-based event triggering. Relating CAD assembly constraints to force-based robot tasks is an ongoing research topic [19], which currently goes through a selection process of available assembly skills rather than using the assembly constraint directly.

3.2. Distance Measures

The assembly constraints are defined between geometric elements whose placement is defined relative to the part coordinate frame. The parts either have predefined locations in the work cell, are located with vision-based localization systems, or are rigidly grasped by the robot. The relative placement and parameters of the geometric elements are extracted from the STEP file, and used to describe the assembly relative to the robot gripper in the task specification.

Establishing a constraint between parts is considered to be the same as minimizing a distance measure between the geometric elements on the parts. This is achieved by selecting a joint velocity on the robot which imposes an exponential convergence of the distance measure. For joint variables $\mathbf{q} \in \mathbb{R}^n$, and scalar distance function e, this is achieved by ensuring

$$\dot{e}(t, \mathbf{q}) = \frac{\partial e}{\partial \mathbf{q}} \dot{\mathbf{q}} = -Ke(t, \mathbf{q}) - \frac{\partial e}{\partial t}$$

holds for the choice of $\dot{\mathbf{q}}$ that will command the robot. K is a tunable gain that defines the rate of convergence to the task. Slack variables can be included to handle incongruous tasks.

A sample of the distance measures used to implement the constraints described in Table 1 in eTaSL are given in Table 2. With $\mathbf{p} \in \mathbb{R}^3$ being a point relative to the shape representation, and $\mathbf{n} \in \mathbb{R}^3$ being a vector in the reference frame of the shape representation, lines are represented by $(\mathbf{p}, \mathbf{d})$ describing a point $\mathbf{p}$ on the line and its direction $\mathbf{d}$. These are implemented as eTaSL expressions using the `Vector` datatype, which is normalized to unit vectors for directions, and geometric entities implemented in eTaSL follow the parametrization described in the geometry schema ISO 10303-42.

Table 2. Summary of Distance Measures - eTaSL

Description	Distance Measure expression
Parallel (direction/normal/axis $\mathbf{d}_1, \mathbf{d}_2$	$\|\mathbf{d}_1 \times \mathbf{d}_2\|$
Perpendicular (direction/normal/axis $\mathbf{d}_1, \mathbf{d}_2$)	$\|\mathbf{d}_1 \cdot \mathbf{d}_2\|$
Distance point ($\mathbf{p}_1$) - line ($\mathbf{p}_1$,$\mathbf{d}_2$)	$\|\mathbf{p}_1 - \mathbf{p}_2 - ((\mathbf{p}_1 - \mathbf{p}_2)) \cdot \mathbf{d}_2)\mathbf{d}_2\|$
Distance line ($\mathbf{p}_1, \mathbf{d}_1$) - line ($\mathbf{p}_2, \mathbf{d}_2$)	$\begin{cases} \text{Distance point } (\mathbf{p}_1) \text{ - line } (\mathbf{p}_2, \mathbf{d}_2), & \text{if } \|\mathbf{d}_1 \times \mathbf{d}_2 = 0\| \\ \frac{\|(\mathbf{p}_1 - \mathbf{p}_2) \cdot (\|\mathbf{d}_1 \times \mathbf{d}_2\|)\|}{\|\mathbf{d}_1 \times \mathbf{d}_2\|}, & \text{otherwise} \end{cases}$
Distance line ($\mathbf{p}_1$,$\mathbf{d}_1$) - plane($\mathbf{p}_2$, $\mathbf{n}_2$)	$\begin{cases} \|(\mathbf{p}_1 - \mathbf{p}_2) \cdot \mathbf{d}_1\|, & \text{if } \mathbf{d}_1 \cdot \mathbf{n}_2 = 0 \\ 0, & \text{otherwise} \end{cases}$

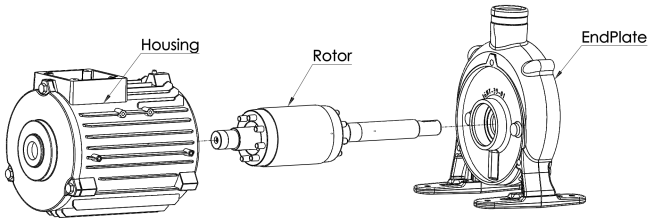

Fig. 2. Motor Assembly - Exploded View

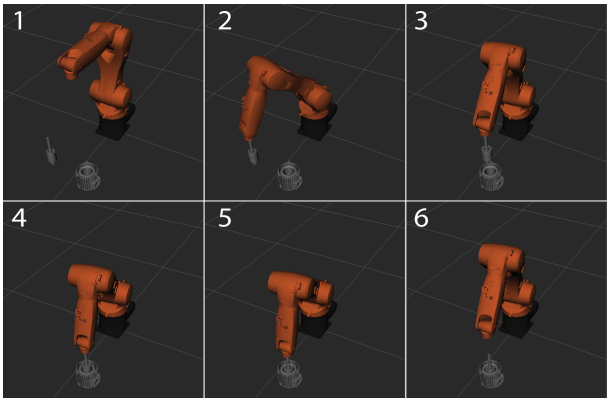

Fig. 3. Assembly sequence, frames 1 and 2 show the part being grasped and lifted above the housing, frames 3-6 show the motion being guided by the assembly constraints for inserting the rotor.

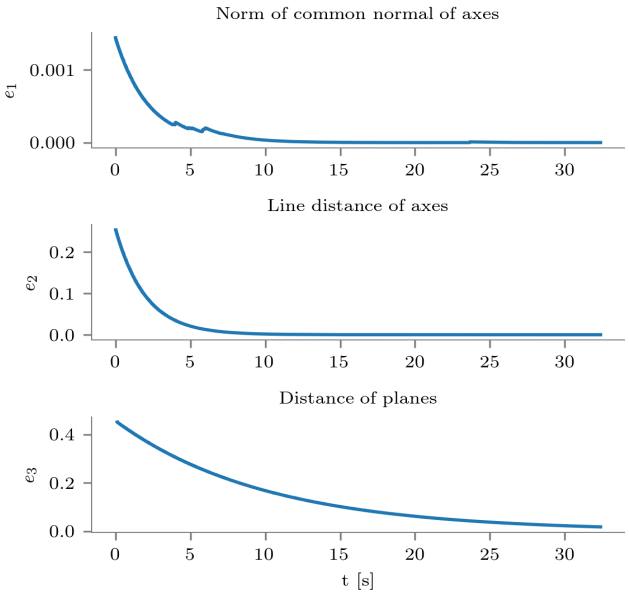

Fig. 4. The distance measures over time: the common normal of the rotor's cylinder axis and the housing's hole cylinder axis, the line distance of the axes, and the distance between the bottom plane of the rotor and the bottom plane of the housing.

4. Test Case

In this test case a rotor is inserted into a motor housing as part of a motor. The motor assembly is shown in Fig. 2. The test case is implemented using the *etasl_ros_control* library [20].

The purpose of this test case is to show the generation of robot motion derived directly from STEP AP242 assembly constraints translated into eTaSL tasks, using the above-mentioned distance measures, for programming robotic assembly. The rotor has an *Incidence_assembly_constraint* between its cylinder axis and the cylinder axis of the hole in the housing, and the bottom plane of the rotor has an *Incidence_assembly_constraint* with the bottom plane it is touching. The assembly sequence is shown in Fig. 3. The part is grasped and lifted above the housing in frames 1 and 2; the assembly constraints are then used to guide the motion during frames 3 – 6.

The norm of the common normal between the axes, the distance between the axes, and the distance between the planes during assembly are given in Fig. 4. The incidence constraint between the planes also imposes the two plane normals to be parallel, but that is not included as it has the same behavior as the norm of the common normal between the cylinder axes.

5. Discussion

By using the STEP AP242 directly as tasks in the reactive eTaSL controller, rather than indirectly informing the choice of assembly skill [19], or to define spaces to plan within [3], specific issues arise.

The task convergence formulation of eTaSL is an exponential convergence; this means that one must tune the task specification such that the tasks that define the alignment process, e.g., the line-line incidence of the axes in the rotor example, have converged before the tasks that ensure the mating and surface contacts, e.g., the plane-plane incidence constraint. The exponential convergence formulation will also incur a high velocity at the beginning and a low velocity at the end of the pro-

cess. The high initial velocity, and discontinuity in acceleration, might present some problems with physical equipment.

These issues suggest that a more advanced controller formulation than the simple exponential convergence may be required, or that a planning process must be used to figure out which constraints must be achieved before others.

As mentioned earlier, the force requirements for the assembly tasks can be estimated from the mating fits based on the GD&T information. Other process-specific requirements and constraints can be added to the STEP file in the form of annotations and can be used in the task specification. The annotations can be added as per the ontological approaches to improve data reuse in downstream operations.

The second edition of STEP AP242 will add more PMI features for welding, GD&T of threads, complex holes, and other enhancements for new manufacturing features [16]. These new enhancements, along with other features like kinematics, can be explored, and methods can be developed to use them for effective automation of manufacturing and assembly.

6. Conclusions and Future Research

In this paper, we have presented how product information can be extracted from STEP AP242 exchange files and used in task specification for robotic assembly in general, and a motor assembly task in particular. This is the first step towards leveraging model-based definition and STEP AP242 in automatic programming of robotic assembly tasks.

The motor assembly test case was implemented using eTaSL/eTC – a framework for constraint-based robot programming – where the assembly constraints derived from the exchange file were converted to distance measures in the task specification. That is, the assembly constraints of the 3D CAD model of the motor were converted to motion constraints of the robot.

One issue that arose in the presented approach was due to the exponential convergence formulation: The task has a high initial velocity and low final velocity and thus slow convergence when the distance measures, or task errors, are low. This high initial velocity might present some problems with physical equipment, while the slow convergence is unfortunate due to possibly high cycle times. These issues will be addressed in future research.

Acknowledgements

The work reported in this paper is partially based on activities within centre for research based innovation SFI Manufacturing, and partially funded by the Research Council of Norway under contract number 237900.

References

[1] J. B. Herron, *Re-Use Your CAD: The Model-Based CAD Handbook*. CreateSpace Independent Publishing Platform, 2013.

[2] Uros Urbas, Rok Vrabic, Nikola Vukasinovic, "Displaying Product Manufacturing Information in Augmented Reality for Inspection," *Procedia CIRP*, vol. 81, pp. 832–837, 2019.

[3] N. Somani, *Constraint-based Approaches for Robotic Systems: from Computer Vision to Real-Time Robot Control*. PhD Thesis, Technical University, Munich, 2018.

[4] L. I. Lieberman, M. A. Wesley, "AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly," *IBM Journal of Research and Development*, vol. 21, no. 4, pp. 321–333, 1977.

[5] Stephen G. Kaufman, Randall H. Wilson, Rondall E. Jones, Terri L. Calton, "The Archimedes 2 Mechanical Assembly Planning System," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3361–3368, 1996.

[6] Heiko Mosemann, Friedrich M. Wahl, "Automatic Decomposition of Planned Assembly Sequences Into Skill Primitives," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 709–718, 2001.

[7] U. Thomas, F. M. Wahl, "A System for Automatic Planning, Evaluation and Execution of Assembly Sequences for Industrial Robots," *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 1458–1464, 2001.

[8] A. Perzylo, M. Rickert, B. Kahl, N. Somani, C. Lehmann, A. Kuss, S. Profanter, A. B. Beck, M. Haage, M. Rath Hansen, M. T. Nibe, M. A. Roa, O. Sornmo, S. Gestegard Robertz, U. Thomas, G. Veiga, E. A. Topp, I. Kessler, and M. Danzer, "Smerobotics: Smart robots for flexible manufacturing," *IEEE Robotics Automation Magazine*, vol. 26, pp. 78–90, March 2019.

[9] A. Perzylo, N. Somani, S. Profanter, I. Kessler, M. Rickert, and A. Knoll, "Intuitive instruction of industrial robots: Semantic process descriptions for small lot production," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2293–2300, Oct 2016.

[10] N. Somani, M. Rickert, and A. Knoll, "An exact solver for geometric constraints with inequalities," *IEEE Robotics and Automation Letters*, vol. 2, pp. 1148–1155, April 2017.

[11] Joris De Schutter, Tinne De Laet, Johan Rutgeerts, Wilm Decre, Ruben Smits, Erwin Aertbelien, Kasper Claes, Herman Bruyninckx, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *International Journal of Robotic Research*, vol. 26, no. 5, pp. 433–455, 2007.

[12] Erwin Aertbelien, Joris De Schutter, "eTaSL/eTC: A constraint-based Task Specification Language and Robot Controller using Expression Graphs," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1540–1546, 2014.

[13] Ruben Smits, Tinne De Laet, Kasper Claes, Herman Bruyninckx, Joris De Schutter, "iTASC: a Tool for Multi-Sensor Integration in Robot Manipulation," *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 426–433, 2008.

[14] T. Kramer and X. Xu, *STEP in a Nutshell*, pp. 1–22. London: Springer London, 2009.

[15] ISO 10303-242:2014, "Industrial automation systems and integration Product data representation and exchange - Part 242: Application protocol: Managed model-based 3D engineering," *International Organization for Standardization, Geneva, Switzerland*.

[16] "Development of STEP AP 242 ed2: Managed Model Based 3D Engineering," *Whitepaper; STEP AP242 Project Committee*, 2014.

[17] Jean-Yves Delaunay, "Use of ISO 10303 AP242 in the European A&D industries," *Global Product Data Interoperability Summit 2018*, 2018.

[18] Mathias Hauan Arbo, Yudha Pane, Erwin Aertbeliën, Wilm Decré, "A System Architecture for Constraint-Based Robotic Assembly with CAD Information," *IEEE International Conference on Automation Science and Engineering*, pp. 690–696, 2018.

[19] Yudha Pane, Mathias Hauan Arbo, Erwin Aertbeliën, Wilm Decré, "A System Architecture for Constraint-Based Robotic Assembly with Sensor-Based Skills," *IEEE Transactions on Automation Science and Engineering*, Submitted.

[20] Lars Tingelstad, Mathias Hauan Arbo, "etasl_ros_control." https://doi.org/10.5281/zenodo.3610916, Jan. 2020.