Toward Efficient Robot Teach-In and Semantic Process Descriptions for Small Lot Sizes

Alexander Perzylo, Nikhil Somani, Stefan Profanter, Markus Rickert, Alois Knoll
fortiss GmbH, An-Institut Technische Universität München, Munich, Germany

***Abstract*—We present a novel robot programming methodology that is aimed at reducing the level of robotics expert knowledge needed to operate industrial robotic systems by explicitly modeling this knowledge and abstracting it from the user.**

Most of the current robot programming paradigms are either user-centric and fully-specify the robot's task to the lowest detail (used mostly in large industrial robotic systems) or fully autonomous solutions that generate the tasks from a problem description (used often in service and personal robotics). We present an approach that is user-centric and can interpret underspecified robot tasks. Such task descriptions make the system amenable for users that are experts in a particular domain, but have limited knowledge about robotics and are thus not able to specify low-level details and instructions. Semantic models for all involved entities enable automatic reasoning about underspecified tasks and missing pieces of information.

We demonstrate this approach on an industrial assembly use-case and present a preliminary evaluation—both qualitatively and quantitatively—vis-à-vis state-of-the-art solutions available from industrial robot manufacturers.

I. Introduction

After a long fordistic period of industrial production, there has been a trend in some parts of today's industry toward individualized products. Additionally, robot-based industrial automation increasingly diffuses into small and medium-sized enterprises (SMEs). In both cases, the industry has to adapt their production processes for small lot sizes and a high number of product variants. Hence, they require their robot systems to allow for rapid changeovers and efficient teaching. In this context, commercial viability of automated production is highly influenced by the time required to teach new processes and to adapt existing processes to variations of a product. However, most SMEs cannot build the necessary expertise in robotics in-house and have to rely on system integrators. This drastically reduces the usability of robot systems for small batch assembly.

Classical teaching concepts for robot systems offer dedicated robot programming languages, which are difficult to learn and far from intuitive. Human operators have to understand and use concepts like Euler angles and work with raw Cartesian coordinates, e.g., in order to define a grasp pose. An important shortcoming of those programming languages is the lack of semantic meaning. For instance, from looking at such robot programs it is not possible to see what kind of object is being manipulated or to easily keep track of the structure of the overall process.

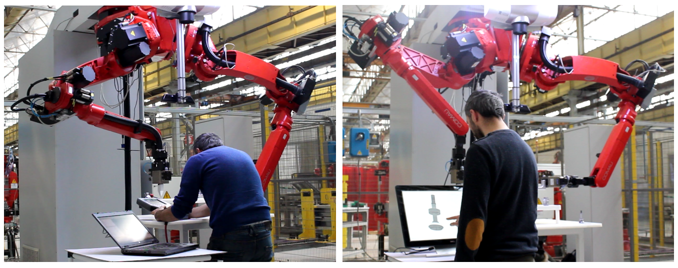

(a) (b)

Fig. 1. Toward efficient robot programming: moving from (a) a classical approach using the teach pendant using low-level programming languages to (b) efficient robot programming using high-level semantic process descriptions.

A more intuitive teaching paradigm is teaching through manual guidance. Based on physical interaction of the operator with the robot, robots can be quickly moved and the resulting poses can be stored and re-used. It spares the human operator the cumbersome effort of teaching positions through jogging the robot via a teach pendant. Obviously, this approach is only feasible for small robots. Besides, the resulting robot program still does not know anything about the operator's intention.

Industrial solutions such as *Delmia* or *Process Simulate* offer sophisticated robot programming interfaces that scale to a full production line with multiple robots. However, they require the programmer to specify robot tasks on a low level that requires expertise in robotics, thereby making these solutions difficult to use for untrained shop floor workers. The operators typically have to train for several weeks to learn how to use these interfaces. Even small changes in the robot program require significant programming effort.

In contrast to this, research—particularly in the domain of service robotics—is being carried out to develop fully autonomous cognitive robotic systems that are able to translate high-level task descriptions to appropriate robot actions. These systems are able to perceive their environment and to reason about the implications of their tasks. Many of them provide natural language interfaces, e.g., by exploiting task descriptions from wikis or by interpreting speech input from the operator. However, as of now, these systems have not had a real impact on industrial applications.

In this paper, we introduce a concept tailored to intuitive teaching of tasks for industrial robotic workcells (Fig. 1) by utilizing approved techniques from the field of cognitive robotics. The operator is still in charge of controlling most aspects, whereas the cognitive capabilities of the robot system are used to increase the efficiency in human-robot communication. Compared to classical approaches, it presupposes less knowledge about the system. As a result, the required time to train a human worker can be significantly reduced. The system may resort to previously modeled knowledge about certain industrial domains, processes, interaction objects, and the workcell itself. Hence, the communication between the operator and the robot system can be lifted to a more abstract and semantically meaningful level, where the operator can talk about, e.g., which object to pick instead of specifying raw coordinates.

II. Related Work

The strive for an easier, intuitive, and more automated teaching process for robots dates back more than 40 years now: In 1972 a prototype was developed that was able to assemble simple objects from plan drawings using a vision system [8]. Another early approach defined constraints between two objects using planar and cylindrical matching and developed an offline object level language for programming robot assembly tasks [1, 5]. RALPH [11] is a similar system which uses information from CAD drawings to automatically generate a process plan for the manipulator to perform the task. A hierarchical graph-based planning algorithm for automatic CAD-directed assembly is described in [9]. These classical systems are not human-centric, i.e., the robot program is generated automatically giving no possibility to a human worker to alter or modify the proposed assembly. They also require accurate models to perform automatic plan generation. Additionally, the drawings or CAD models do not include any semantic description to do further reasoning about its properties, e.g., if a cylindrical part is the thread of a screw and needs to be fastened to assemble it.

Pardo-Castellote [13] described a system for programming a dual arm robot system through a graphical user interface. Despite being one of the more advanced systems at the time of publishing, it only supported simple pick and place actions and did not allow in-air assembly or, for example, horizontally placing a cylindrical object into a corresponding hole.

In recent years, there have been research efforts toward teaching robot programs by observing human demonstration [10, 7, 18]. They rely on a predefined set of speech commands and demonstrating actions or require various different sensors to detect human motions. Additional effort is required to teach human workers how to demonstrate tasks in a compatible way. Sensor requirements reduce the mobility of such systems since all sensors need to be set up in each working area.

A comprehensive overview of programming methods for industrial robots until the year 2010, including online (operator assisted, sensor guided), offline programming (CAD data), and augmented reality is presented in [12]. Current research mainly focuses on task-level programming that requires only a small library of predefined lower-level building blocks, called skills [4, 14, 15].

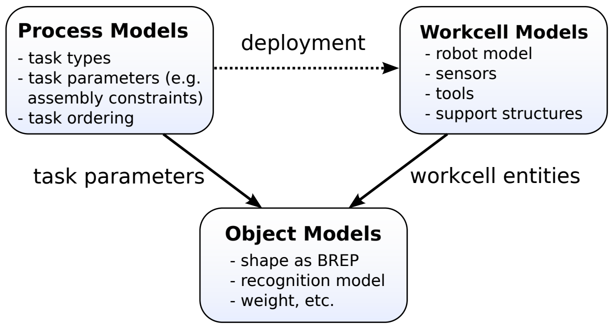


Fig. 2. Overview of semantic models and interrelations. Object models may serve as parameters for certain tasks specified in process models. A workcell model relies on object models for describing its entities. For executing process models, they are deployed on workcells exploiting the information in their workcell models.

With the rapid growth of the World Wide Web and widespread availability of knowledge, Knowledge Bases gained increasing importance for intuitively teaching robots [3]. RoboEarth [17] and RoboHow [16] developed robotic systems that use semantic descriptions to share knowledge between different robots. Using Knowledge Bases for defining additional skill parameters simplifies the whole teaching process even further [15, 2].

III. Toward a Natural Teaching Paradigm

Current industrial robotic systems require the user to be an expert not only in the application domain but also at robot programming. The key motivation in our approach is to substantially reduce the level of robotics expertise required to use such systems to a level where a shop floor worker with minimal knowledge about robotics can instruct, interact with, and operate the system. In this programming paradigm, the robotics and domain specific knowledge is modeled explicitly (Fig. 2) and the system needs to be able to understand, interpret, and reason about it. We have chosen the Web Ontology Language (OWL) for this knowledge representation, primarily because OWL is based on a formal specification, i.e., a description logic, which facilitates logical inference. Another advantage is the ease with which additional knowledge (e.g., new workpieces) can be added to the system. This facilitates the separation of knowledge and code, thus enabling addition of information without changing the implementation.

A key concept in this design is programming at object level. In this approach, the user specifies robot tasks in terms of the objects involved and the relevant parameters. This is in contrast to the traditional teach pendant-based approaches where tasks are specified in terms of raw coordinate frames. While there exist some approaches that involve task specification in terms of the coordinate frames attached to an object [6], they are restricted to coordinate frames only.

In several manufacturing domains, especially assembly, products are designed by domain experts using specialized CAD software. In this process, the information and ideas

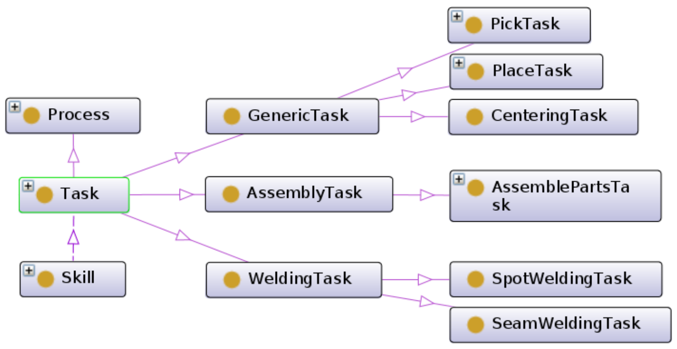


Fig. 3. Excerpt of task taxonomy. An abstract task description might have multiple actual implementations, i.e., robot or tool skills.

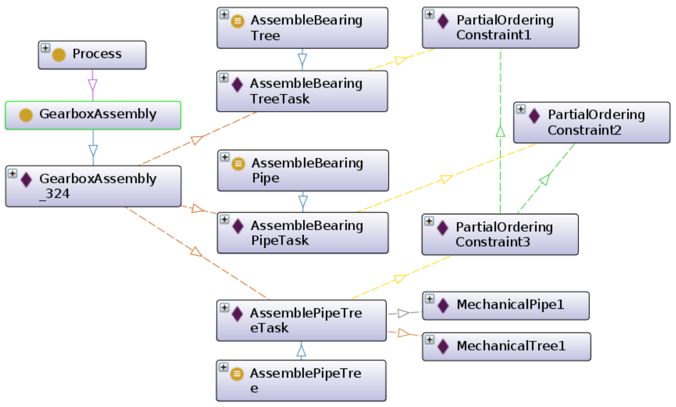


Fig. 4. Process model for the 3-step assembly of the core part of a gearbox. Boxes featuring a *yellow* circle or a *purple* rhombus represent classes and instances of those classes respectively.

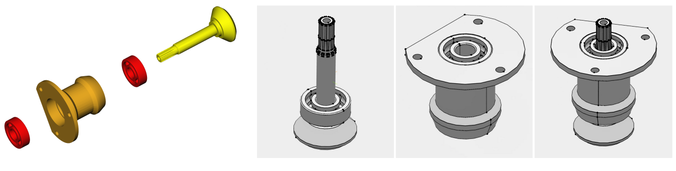
that the product designer had in mind while designing the product are lost and only the finished CAD model of the product is sent to manufacturing. In contrast to this, we aim to model, store, and exploit this information in order to ease the programming of the manufacturing process. The first step in this direction—most suitable for assembly tasks—is to include in the product description not only the final CAD model, but also the semantically relevant geometrical entities in it, the constraints between these geometrical entities, and the individual assembly steps that the designer took in designing the final product.

IV. Semantic Process Models

Semantic process models are descriptions of the steps required for manufacturing a product, built by arranging tasks that have been hierarchically defined in a potentially constrained order. Each of these tasks are object-level descriptions of the individual steps in the process, which might be underspecified and have pre-post-conditions. Due to this abstract description, they can be understood and executed not only by a machine but also by a human operator. Robotic systems provide corresponding low-level implementations (called *skills*) that require a complete parametrization with actual numerical values.

We have implemented a taxonomy of industrial task types to deal with different industrial domains, e.g., wood working, welding, and assembly (Fig. 3). Every type of task has a certain set of parameters that are required to be set by the user during the Teach-In phase. There are optional parameters which can either be specified by the operator or automatically inferred by the system upon deployment.

Fig. 4 visualizes an excerpt of a semantic process model that describes the process of assembling the core part of a gearbox as shown in Fig. 5. It shows the class level concept *GearboxAssembly* and its instantiation *GearboxAssembly_324*. Every attempt at executing the assembly would result in an additional instantiation of the class level description of the task. This allows to log all relevant information collected during execution, e.g., in case of an anomaly we can determine which task in which process instance failed, when it happened, and due to what error. The *GearboxAssembly_324* process consists of three *Assembly* tasks, i.e., *AssembleBearingTreeTask*, *AssembleBearingPipeTask*, and *AssemblePipeTreeTask*.



(a) Exploded view (b) Assembly steps

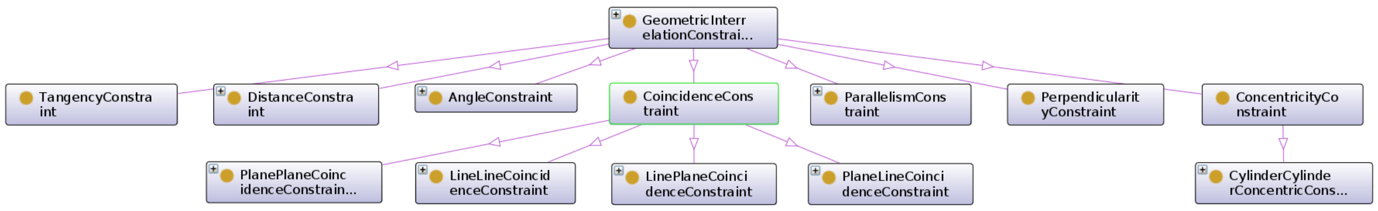
Fig. 5. Industrial use-case based on assembling four work pieces in three steps to form the core part of a gearbox

The order of these tasks is not yet fully specified. Instead, three partial ordering constraints have been specified, which assert that the task instance associated with *PartialOrderingConstraint**3*** has to succeed the tasks associated with *PartialOrderingConstraint**2*** and *PartialOrderingConstraint**1***. No further restrictions have been modeled, which results in the order of the two tasks *AssembleBearingTreeTask* and *AssembleBearingPipeTask* not being constrained. The *AssembleBearingPipeTask* links two interaction objects *MechanicalPipe1* and *MechanicalTree1* as its parameters. The geometric constraints specifying the assembly pose have been defined on a sub-object level between single faces of the two object models as explained in Section V-B.

V. Semantic Object Models

Object models form one of the key pillars of our proposed robot programming approach. They are modeled in a hierarchical fashion, wherein properties of generic object types can be re-used in the more specific ones. Process descriptions refer to objects and their properties as parameters for each robot task. The requirements of a task can help filter the type of object suitable for it.

Simple object properties include information such as the name, weight, pose, or material. Additional information such as specialized appearance models that can be exploited by computer vision modules to detect the objects in a workcell can also be linked to it. Basic geometric properties include the



GeometricInterrelationConstrai...
TangencyConstraint
DistanceConstraint
AngleConstraint
CoincidenceConstraint
ParallelismConstraint
PerpendicularityConstraint
ConcentricityConstraint
PlanePlaneCoincidenceConstrain...
LineLineCoincidenceConstraint
LinePlaneCoincidenceConstraint
PlaneLineCoincidenceConstraint
CylinderCylinderConcentricCons...

Fig. 6. Upper taxonomy of geometric interrelation constraints

bounding box, the corresponding dimensions, and the polygon mesh used for rendering the object.

Most classical approaches rely only on basic geometric information. In our approach, we aim to preserve all relevant information produced while designing the object. This includes the CAD model used for creating the polygon mesh. We support geometric representations at multiple levels of detail, from points and coordinate frames, to semantically meaningful entities such as lines and circles or planes and cylinders. Constraints between these geometric entities can also be used to describe parameters for robot tasks in an intuitive way.

A. Boundary Representation of Objects

A Boundary Representation (BREP) of CAD data describes the geometric properties of points, curves, surfaces and volumes using mathematical models as its basis. CAD models are created by defining boundary limits to given base geometries. The BREP specification distinguishes geometric and topological entities, as illustrated in Fig. 7. Geometric entities hold the numerical data, while the topological entities group them and arrange them in a hierarchical fashion.

1) Topological Entities: The BREP standard specifies eight kinds of topological entities that are connected through properties which resemble the relations depicted in Fig. 7: *Vertex*, *Edge*, *Face*, *Wire*, *Shell*, *Solid*, *CompSolid*, and *Compound*. Only *Vertices*, *Edges*, and *Faces* have direct links to geometric entities. A *Vertex* is represented by a point. An *Edge* is represented by a curve and bounded by up to two *Vertices*.

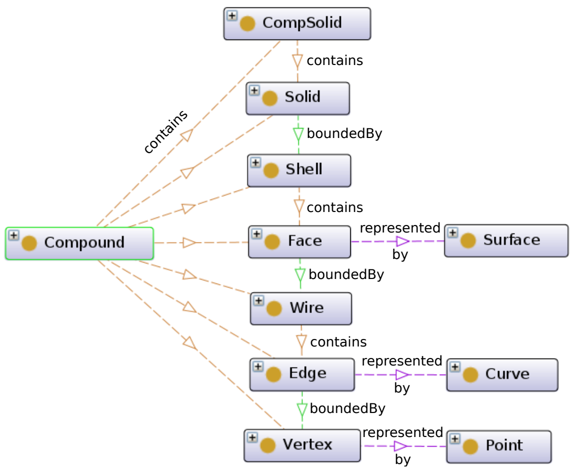


Fig. 7. Illustration of the basic BREP structure. Its data model comprises geometric entities (Point, Curve, Surface) and topological entities (the rest).

A *Wire* is a set of adjacent *Edges*. When the *Edges* of a *Wire* form a loop, the *Wire* is considered to be closed. A *Face* is represented by a surface and bounded by a closed *Wire*. A *Shell* is a set of adjacent *Faces*. When the *Faces* of a *Shell* form a closed volume, the *Shell* can be used to define a *Solid*. *Solids* that share common *Faces* can be grouped further into *CompSolids*. *Compounds* are top-level containers and may contain any other topological entity.

2) Geometric Entities: The topological entities may link to three types of geometric entities, which are *Points*, *Curves*, and *Surfaces*. They represent 0-, 1-, and 2-dimensional geometries respectively. *Curves* and *Surfaces* are defined through parameterizable mathematical models. Supported curve types can be categorized as unbounded curves (e.g., lines, parabolas, or hyperbolas) and bounded curves (e.g., Bezier curves, B-spline curves, circles, or ellipses). Offset curves represent a translated version of a given base curve along a certain vector, whereas trimmed curves bound a given base curve by limiting the minimum and maximum parameters of their mathematical model. In case the exact model is unknown, a curve might also be approximated by a polygon on triangulated data. The geometric representation of an *Edge* can be specified by a 3D curve, or a 2D curve in the parameter space of each surface that the *Edge* belongs to.

Surfaces rely on unbounded mathematical models (e.g., planes, cones, or cylindrical surfaces) and bounded models (e.g., Bezier surfaces, B-spline surfaces, spheres, or toruses). Surfaces can also be defined as linearly extruded curves. An offset surface translates a base surface along a given vector, and a revoluted surface is created by rotating a given base curve around a given direction vector. Again, if the exact mathematical model of a surface is unknown, an approximation based on triangulation might be specified.

B. Geometric Constraints between Objects

Given the rich semantic description of CAD models, it is possible to refer to arbitrary parts of a CAD model and to link it with additional information. Fig. 6 shows the upper taxonomy of the geometric constraints that have been designed to represent assembly constraints. Those constraints are meant to be specified between points, curves, and surfaces of objects.

In our representation a geometric constraint refers to two geometric entities: A base entity with a defined pose and a constrained entity whose pose depends on the fixed entity and the constraint itself.

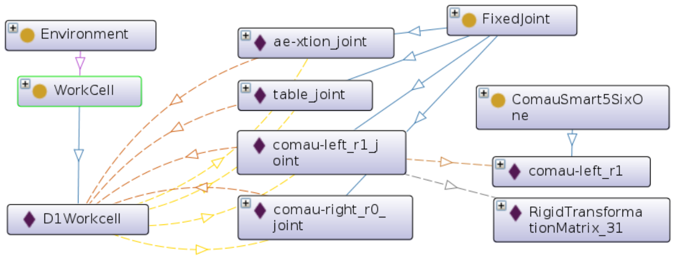


Fig. 8. Excerpt of a semantic workcell model. From left to right, this illustration shows the class concept of a workcell and a particular instantiation of it, called *D1Workcell*. In this example, a rigidly attached RGBD sensor, work table, and two robots are associated with the workcell through *FixedJoint* instances. The visualization exemplary shows the robot class (*ComauSmart5SixOne*) and instance (*comau-left-r1*) of one of the two robots.

VI. Semantic Workcell Models

A workcell model describes the physical setup of the workcell, including robots, tools, sensors, and available skills. For instance, the dual-arm robot workcell depicted in Fig. 1 contains two robot arms, a work table, and an RGBD sensor. Fig. 8 shows an excerpt of the corresponding semantic workcell description. It asserts the *Workcell* instance called *D1Workcell* that links to its contained entities. These links are represented through *FixedJoint* instances. In order to specify the poses of the sensor, table, and robot bases with respect to the workcell origin, the *FixedJoint*s refer to instances of *RigidTransformationMatrix*.

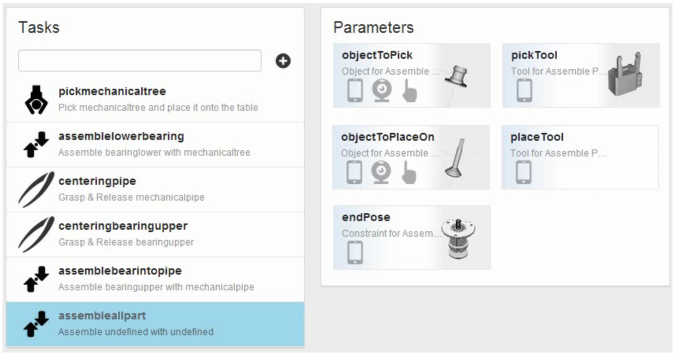
VII. Teaching Interface

Having to manipulate the semantic process descriptions manually would only shift the required expert knowledge for using the robot system from the domain of robotics to the domain of knowledge engineering. Therefore, we include a graphical user interface for the human operator based on HTML5 and JavaScript, that abstracts from the semantic modeling language and can run in any modern web browser.

Fig. 9 depicts the main view of the GUI showing a process plan consisting of six tasks and the parameters of the selected task. Parameters *objectToPick* and *objectToPlaceOn* can be set through selecting the desired objects from a list (or other modalities that are not described in this paper). *pickTool* and *placeTool* are optional parameters, which might be used to specify compatible tools for grasping the two objects involved in the assembly. Parameter *endPose* defines the assembly pose and can be set through a dedicated 3D interface. In the example shown in Fig. 10, two cylindrical surfaces have been selected and a concentricity constraint specified.

VIII. Execution Framework

The process plan designed using the intuitive teaching interface is an underspecified, hardware independent description of the robot's tasks. In order to execute the task on a specific workcell, the tasks need to be fully specified and mapped to executable actions for the robot. As an example, the tool for grasping an assembly object is an optional parameter in the task specification interface. The appropriate tool can be


Tasks

pickmechanicaltree
Pick mechanicaltree and place it onto the table

assemblelowerbearing
Assemble bearinglower with mechanicaltree

centeringpipe
Grasp & Release mechanicalpipe

centeringbearingupper
Grasp & Release bearingupper

assemblebearingintopipe
Assemble bearingupper with mechanicalpipe

assembleallpart
Assemble undefined with undefined

Parameters

objectToPick
Object for Assemble ...

pickTool
Tool for Assemble P...

objectToPlaceOn
Object for Assemble ...

placeTool
Tool for Assemble P...

endPose
Constraint for Assem...

Fig. 9. Snippet of the relevant subpart of the teaching interface that acts as a frontend to the semantic process descriptions

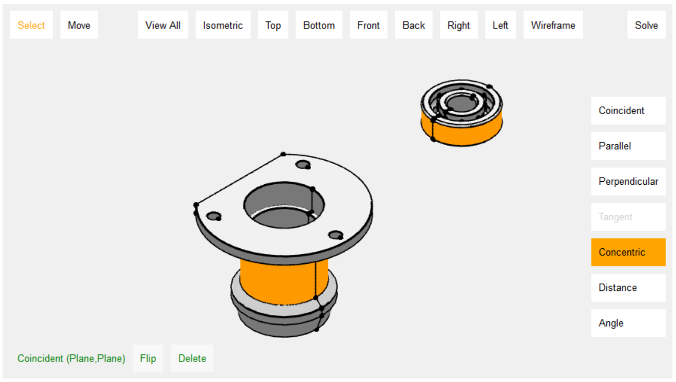


Fig. 10. Mode of the intuitive teaching interface for defining geometric constraints between two objects. As the two objects are described in a boundary representation, each of the two highlighted cylindrical surfaces can be selected with a single click. Geometric constraints can be chosen from the list on the right. A *PlanePlaneCoincidenceConstraint* (Fig. 6) has already been specified and is visualized in the bottom left corner of the GUI.

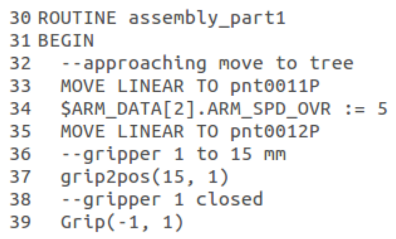
automatically selected by matching the set of available tools in the workcell to the ones suitable for manipulating the object.

Once completely specified, the assembly task is mapped to a set of primitive actions (e.g., move, open/close gripper commands) that are offered by the robot controller interface. The system can reason if this mapping is even possible for a specific task on a specific workcell and provide the user with appropriate feedback. A world model component maintains the state of all involved entities in the world, including instances of objects, tools, robots, etc. In our experimental setup, poses of detected objects in the workcell are obtained from the vision system and updated in the world model.

Once the task execution begins, monitoring can be done at different levels of abstraction. The individual actions (e.g., open/close gripper) might fail, or the task as a whole might not be possible due to unmet pre-conditions (e.g., assembly object missing). In either case, the error message that is generated contains semantically meaningful information for the user.

IX. Evaluation

In order to evaluate our teaching concept, we carried out a pre-study with one test person. The robotic task used in this evaluation is the partial assembly of a gearbox (Fig. 5) using an



```
ROUTINE assembly_part1
BEGIN
  --approaching move to tree
  MOVE LINEAR TO pnt0011P
  $ARM_DATA[2].ARM_SPD_OVR := 5
  MOVE LINEAR TO pnt0012P
  --gripper 1 to 15 mm
  grip2pos(15, 1)
  --gripper 1 closed
  Grip(-1, 1)
```

Fig. 11. Snippet of a robot program created using the teach pendant, which is hard to understand especially without the programmer's comments

industrial dual-arm robot system. The gearbox comprises four parts which have to be assembled in three steps (Fig. 5(b)). Each assembly step requires sub-millimeter precision, making it an ideal use-case for a precise industrial robot.

The subject first followed the classical approach of programming the task in a robot programming language using a combination of teach pendant and PC. The second attempt was made on a touch screen showing the graphical user interface of our cognitive teaching framework. This comparison is biased against our system as the test person has several years of experience in using the teach pendant and robot programming language, but only received a 10-minute introduction to the intuitive framework right before he had to use it.

A video demonstrating our intuitive interface and the results of this comparison can be found online at http://youtu.be/B1Qu8Mt3WtQ.

A. Classical Teaching

The program for performing this robotic task is fairly complex, involving numerous robot movement commands and synchronization between the two robot arms (Fig. 11). Hence, the skeleton for the program was created on a PC and the specific robot poses for each movement were obtained by jogging the robot using a teach pendant (Fig. 1(a)). Given the precision required by the assembly task, the parts needed to be placed into fixtures and the robot poses fine-tuned accordingly.

B. Intuitive Teaching

The process programmed using the intuitive teaching interface is shown in Fig. 9. It consists of six steps, of which three are assembly tasks parametrized using the BREP mating interface. We employ a simple CAD model-based vision system using a Kinect-like camera placed on top of the working table (at a distance of ˜1 m) to recognize objects, providing a positioning accuracy of ˜1 cm. There are two centering steps that use the imprecise object poses obtained from the vision system and center the parts between the gripper fingers of a parallel gripper, thereby reducing their pose uncertainties.

C. Pre-Study Results

The experiments have been carried out in the same industrial dual-arm robot workcell (Fig. 1). They investigated the times required to program the assembly process from scratch, to adjust the object poses in the existing program, and to adjust the robots' approach poses for the objects.

TABLE I

REQUIRED TIMES FOR ACCOMPLISHING VARIOUS PROGRAMMING TASKS FOR THE CLASSICAL AND THE INTUITIVE APPROACH.

Task	Classical	Intuitive	
	time in min	time in min	saved time in %
Full assembly	48	8	83
Adjust object poses	23	0	100
Adjust approach poses	5	2	60

Comparing the times required to teach the full assembly using the two different methods shows a time-saving of 83 % for the intuitive teaching approach (Table I). Updating the pre-existing robot program to reflect changes in the work pieces' positions took 23 min for the classical teaching approach, whereas the intuitive teaching framework's vision system naturally coped with the changes automatically. Adjusting the approach poses for the work pieces had been accomplished in 5 min on a teach pendant. Using the intuitive teaching interface it required 2 min.

While the programmer decided to use precise positioning by jogging the robot using the classical approach, this option was not available for our intuitive approach. With a more precise vision system, the additional centering tasks would not be necessary. This centering technique could have also been used for the classical approach. However, as can be seen from the second comparison (Table I), fine tuning object poses takes much more time than programming centering tasks.

X. Conclusion

The pre-study indicates the potential of semantically meaningful and object-centric robot programming, as compared to classical methods. We demonstrated that domain-specific interfaces in a cognitive system with domain knowledge eases the tedious task of programming a robot to a large extent. By programming at the object level, the low-level details pertaining to robot execution did not have to be programmed. The resulting process plan was more concise, readable, and re-useable. The communication between the operator and the robot system could be lifted on an abstract level, relying on previously available knowledge on both sides. This enables a new operator to use the system with minimal training.

The preliminary results from our pre-study are very promising and we plan to perform additional evaluations and full-scale user studies in the future, which will provide a more holistic assessment of the proposed robot teaching approach.

Acknowledgements

We would like to thank Comau Robotics for providing the industrial robot workcell used in our experiments and Alfio Minissale for his support and participation in the evaluation.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 287787 in the project SMErobotics.

References

[1] A.P. Ambler and R.J. Popplestone. Inferring the positions of bodies from specified spatial relationships. *Artificial Intelligence*, 6(2):157–174, June 1975.

[2] Stephen Balakirsky and Andrew Price. Implementation of an ontology for industrial robotics. *Standards for Knowledge Representation in Robotics*, pages 10–15, 2014.

[3] Muhammad Baqar Raza and Robert Harrison. Design, development & implementation of ontological knowledge based system for automotive assembly lines. *International Journal of Data Mining & Knowledge Management Process*, 1(5):21–40, 2011.

[4] Simon Bøgh, Oluf Skov Nielsen, Mikkel Rath Pedersen, Volker Krüger, and Ole Madsen. Does your robot have skills? *Proceedings of the International Symposium on Robotics*, page 6, 2012.

[5] D. F. Corner, A. P. Anbler, and R. J. Popplestone. Reasoning about the spatial relationships derived from a RAPT program for describing assembly by robot. pages 842–844, August 1983.

[6] Tinne De Laet, Steven Bellens, Ruben Smits, Erwin Aertbeliën, Herman Bruyninckx, and Joris De Schutter. Geometric relations between rigid bodies (part 1): Semantics for standardization. *IEEE Robotics Automation Magazine*, 20(1):84–93, March 2013.

[7] Rüdiger Dillmann. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47(2-3):109–116, June 2004.

[8] Masakazu Ejiri, Takeshi Uno, Haruo Yoda, Tatsuo Goto, and Kiyoo Takeyasu. A prototype intelligent robot that assembles objects from plan drawings. *IEEE Transactions on Computers*, C-21(2):161–170, 1972.

[9] P. Gu and X. Yan. CAD-directed automatic assembly sequence planning. *International Journal of Production Research*, 33(11):3069–3100, 1995.

[10] Monica N. Nicolescu and Maja J. Mataric. Natural methods for robot task learning. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, page 241, 2003.

[11] Bartholomew O. Nnaji. *Theory of automatic robot assembly and programming*. Chapman & Hall, London, 1st edition, 1993.

[12] Zengxi Pan, Joseph Polden, Nathan Larkin, Stephen van Duin, and John Norrish. Recent progress on programming methods for industrial robots. *Proceedings of the International Symposium on Robotics*, 28:619–626, 2010.

[13] Gerardo Pardo-Castellote. *Experiments in the integration and control of an intelligent manufacturing workcell*. Phd thesis, Stanford University, September 1995.

[14] Mikkel Rath Pedersen, Lazaros Nalpantidis, Aaron Bobick, and Volker Krüger. On the integration of hardware-abstracted robot skills for use in industrial scenarios. In *Proceedings of the IEEE/RSJ International Conference on Robots and Systems, Workshop on Cognitive Robotics Systems: Replicating Human Actions and Activities*, 2013.

[15] Maj Stenmark. *Instructing Industrial Robots Using High-Level Task Descriptions*. Licentiate thesis, Lund University, 2015.

[16] Moritz Tenorth and Michael Beetz. KnowRob – a knowledge processing infrastructure for cognition-enabled robots. *International Journal of Robotics Research*, 32(5):566 – 590, April 2013.

[17] Moritz Tenorth, Alexander Perzylo, Reinhard Lafrenz, and Michael Beetz. The RoboEarth language: Representing and exchanging knowledge about actions, objects and environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1284–1289, St. Paul, MN, USA, May 2012.

[18] Andrea L. Thomaz and Cynthia Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172 (6-7):716–737, April 2008.