Industrial Robot Skills

Stenmark, Maj

Published in:
Frontiers in Artificial Intelligence and Applications

DOI:
10.3233/978-1-61499-330-8-295

2013

Industrial Robot Skills

Maj STENMARK [1]
Department of Computer Science, Lund University, Lund, Sweden

Abstract. When robots are working in dynamic environments, close to humans lacking extensive knowledge of robotics, there is a strong need to simplify the user interaction and make the system execute as autonomously as possible. For industrial robots working side-by-side with humans in manufacturing industry, AI systems are necessary to lower the demand on programming time and expertise. One central concept in knowledge modeling for robots is action representation. In this paper, we describe our representation of robot *skills*. The skills have resource requirements, logical and procedural information from which executable code can be generated.

Keywords. knowledge representation, robot skills, industrial robotics, assembly

1. Introduction

In manufacturing industry, robots can still not compete with human labour in low-cost countries. The dynamic environments, the fast changing product lines and the interaction with human coworkers pose problems when trying to replace humans with robots in the factories. One way to overcome these challenges is to add more AI into the systems to make the robots understand the humans better and act more autonomously when carrying out the tasks. Our approach is to add knowledge to the robot system. One central concept in the knowledge modeling is action representation. We refer to actions and robot capabilities as *skills*. A skill is a semantically described state machine consisting of robot motions and end-effector and sensor actions, e.g., gripper movements or locating an object. The skills can be hierarchically composed into more complex state machines and sequenced into a task.

The purpose of the skills is to abstract away from the details of the robot control, allowing non-skilled users to instruct the robots using preprogrammed skills from online libraries, or by extracting the skill representations from user demonstrations. Thus the skills have to be platform independent and generate code that executes robustly. We work primarily with assembly tasks for industrial robots. A typical task is a two-armed assembly of an emergency-stop box or a cell phone. The skills involved require force and torque control for screwing and fitting pieces together.

The objective of my thesis is to create a platform independent skill representation that can be used to create robust programs in dynamic environments. We have identified five requirements on the skill representation: 1) the representations must be possible to extract automatically from other specifications, such as domain specific languages or

[1] E-mail: maj.stenmark@cs.lth.se.

demonstrations, 2) it must be easy to program the robot using the skills, 3) it should be possible to reason and plan using the skills, 4) it must be possible to generate executable code from the specifications for several different platforms and 5) the execution should be robust, thus it should be possible to generate non-nominal behavior such as error handling procedures.

In this paper, we discuss a skill representation fulfilling requirement 1 - 4 and how to extend the representation to also facilitate error handling procedures.

2. System Overview

The system [1] consists of three main components. The central component in the system is a **knowledge base**, which is an online server containing data repositories and ontologies. It also provides computing and reasoning services for programming support, task validation and scheduling [2]. The **Engineering System** is a graphical programming IDE and simulation environment that uses the ontologies provided by the knowledge base to model the workspace objects and downloads skills and tasks from the online libraries. The objects are visualized using CAD models, and the user can attach local coordinate frames to the objects. Constraints between these coordinate frames are used to express the low level control which is evaluated in the **Execution Environment**. The execution environment generates code for the native robot controller and the sensor-based control.

3. Robot Skills

The skill representation incorporates logical and procedural knowledge. The current skill description contains the following:

Resource requirements The skill lists the required robot type, sensors and end-effectors as well as objects in the scene e.g., *two-armed robot*, *force sensor*, *fixture*, *suction gripper* and *two finger gripper*.

Pre- and postconditions The pre- and postconditions are used in action planning and validity checks. Preconditions can express required positions of the objects, such as *Object2 attached to gripper and Object2 located above Oject1*, where *Object1* and *Object2* are instances of work piece types in the ontology. Postconditions express the purpose of the skill: *Frame2 is aligned to Frame1 and Object1 is assembled to Object2* where *Frame1* and *Frame2* are instances of frames belonging to *Object1* and *Object2* respectively.

Input and output parameters The parameters to the skill are described as objects and frames for relative positions or physical properties such as velocity.

Procedural information There are several types of motions and actions in a skill: simple blind motions, gripper or sensor actions and sensor-controlled motions. The blind motions have velocity and positions (relations to frames on objects) as input parameters. The gripper and sensor actions are expressed using the ontology description of the resource, e.g., a gripper can open and close, a camera can output a position of an object etcetera. The sensor-controlled motions are expressed using a set of constraints. Given a frame, they have one search axis, translational

or rotational, a stop condition and additional parameters such as speed and what impedance controller to use. The stop condition is typically a force or torque threshold and/or a timeout. The other axes can have constraints as well, those must be fulfilled during the execution. E.g., *Move along the x-axis in 50 mm/s until a force of 5 N is measured along the axis. At the same time, press down along the y-axis with 2 N.*

Optional information The skills can have optional labels such as human readable descriptions, English names for natural language programming, cycle times for scheduling purposes, log data etcetera.

Logical disjunction and conjunction can be used to express the resource requirements, the pre- and postconditions and the procedural constraints.

These skill representations are enough to program the robot using high-level natural language programming, make action planning and scheduling services and generate executable code.

4. Ongoing Work

To verify that the skill representation can fulfill the first requirement, we want to extract the conditions from natural language. We already have a general purpose natural language interface [4,5], however, we only use it for sequence creation using skills from the library and not to create new skills from scratch. To limit the modeling effort when creating skills, it is valuable to be able to parse English specifications directly.

The final step is to generate error handling policies for the task. The error handling policies can be both on skill and task level. Good error handling policies require knowledge and reasoning such as geometrical reasoning and the saving and classifying of nominal sensor data and cycle times. On a logical level, the challenge is to automatically identify error states and create a plan to go from these states to the goal. In some cases these error handling policies can be generated offline. In other situations, the error handling must be handled online. One approach we are investigating is to let the robot simulate different action sequences online to autonomously find solutions.

5. Related work

Task representation in robotics have a long history [6], starting out using logical approaches but now going towards architectures with a high-level reasoning layer, a low-level control layer and a synchronization layer in the middle. The problem of interfacing between high-level declarative task specifications and primitive motions is still an open question, addressed by several different languages and grammars [7,8,9].

Depending on the application domain and abstraction level, the task description varies greatly. A popular top-bottom approach is the composition of semantically described skills into tasks [10,11]. On the execution level, the program can be expressed as sequential function charts (SFCs) [12], iTaSC constraint specifications [3] or vendor specific code. For error handling, there are approaches for extending the executable state machine with error cases [13] or use online reasoning systems [14].

Acknowledgments

The research leading to these results has received partial funding from the European Union's seventh framework program (FP7/2007-2013) under grant agreements No. 230902 (project ROSETTA) and No. 285380 (project PRACE).

References

[1] Jacek Malec, Klas Nilsson, and Herman Bruyninckx. *Describing assembly tasks in a declarative way.* In ICRA 2013 WS on Semantics, Identification and Control of Robot-Human-Environment Interaction, 2013.

[2] Maj Stenmark and Jacek Malec. *Knowledge-Based Indutrial Robotics*. To appear in Proc. Scandianvian AI Conference 2013.

[3] Joris De Schutter, Tinne De Laet, Johan Rutgeerts, Wilm Decré, Ruben Smits, Erwin Aertbeliën, Kasper Claes, and Herman Bruyninckx. *Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty*. *The International Journal of Robotics Research*, 26(5):433–455, 2007.

[4] Anders Björkelund, Bernd Bohnet, Love Hafdell and Pierre Nugues, *A high-performance syntactic and semantic dependency parser*, Proc. of the 23rd International Conference on Computational Linguistics: Demonstrations. Association for Computational Linguistics, 2010.

[5] Maj Stenmark and Pierre Nugues, *Natural Language Programming of Industrial Robots*, To appear in Proc. International Symposium of Robotics 2013, Seoul, South Korea, 2013.

[6] Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann, 1985.

[7] Ulrike Thomas, Gerd Hirzinger, Bernhard Rumpe, Christoph Schulze and Andreas Wortmann. *A New Skill Based Robot Programming Language Using UML/P Statecharts*. In ICRA 2013.

[8] Jacob Huckaby and Henrik I. Christensen. *A Taxonomic Framework for Task Modeling and Knowledge Transfer in Manufacturing Robotics*. AAAI Technical Report WS, 2012.

[9] Neil Dantam and Mike Stilman. *The Motion Grammar Calculus for Context-Free Hybrid Systems*. American Control Conference, 2012.

[10] Torsten Kröger, Bernd Finkemeyer, and Friedrich M.Wahl *Manipulation Primitives – A Universal Interface between Sensor-Based Motion Control and Robot Programming*. Robotic Systems for Handling and Assembly, STAR 67, pp. 293313.Springer 2010.

[11] Daniel Di Marco, Moritz Tenorth, Kai Häussermann, Oliver Zweigle, Paul Levi. *RoboEarth Action Recipe Execution*. Frontiers of Intelligent Autonomous Systems, pp. 117-126, Springer, 2013.

[12] IEC. IEC 61131-3: Programmable controllers – part 3: Programming languages. Technical report, International Electrotechnical Commission, 2003.

[13] Kei Okada, Yohei Kakiuchi, Haseru Azuma, Hiroyuki Mikita, Kazuto Murase, Masayuki Inaba *Task Compiler : Transferring High-level Task Description. to Behavior State Machine with Failure Recovery Mechanism.* In ICRA 2013.

[14] Michael Beetz, Lorenz Mösenlechner and Moritz Tenorth. *CRAM A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments*. In Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems October 18-22, 2010, Taipei, Taiwan