A System for High-Level Task Specification Using Complex Sensor-based Skills

Maj Stenmark
Department of Computer Science
Lund University, Sweden
Email: maj.stenmark@cs.lth.se

Andreas Stolt
Department of Automatic Control
Lund University, Sweden
Email: andreas.stolt@control.lth.se

***Abstract*—We describe our system for task programming using sensor-based skills and the ongoing work on transforming the high-level specifications of objects and frames into constraints for the motion control.**

I. Introduction

The goal of our research is to semantically describe and reason upon complex sensor-based robot tasks in order to simplify the programming of industrial robots. We are focusing on manipulation tasks for small part assembly, see Fig. 1. A task is realized by a sequence of *skills*, which are semantically annotated state machines. On the lowest level, the skills are specified by simple motion constraints. In order to reuse and reason about the skills, they are uploaded to a knowledge base, named Knowledge Integration Framework (KIF) [3]. KIF contains ontologies representing skill types, objects such as robots, workpieces and sensors, and physical properties. It also provides services that are used e.g. by a graphical task specification tool (see Section III). In this programming environment, the user specifies the desired subgoals of the task, possibly by natural language, and these specifications are translated into robot control programs. Our ongoing research bridges the gap between high-level task specifications and efficient motion control (see Section IV).

II. Sensor controlled skills

The framework used for low-level motion control is based on iTaSC [6]. The interface to the robot [4, 5] admits sending position references and velocity feedforward to the joint servo control loops. An external force/torque sensor is used to give the robot capability to perform force control.

A motion is specified by constraining outputs from a kinematic chain, this can be positions, velocities or forces. The kinematic chain is a specification of the relation between the task variables and the robot, and it is represented as a list of simple transformations. All constraints are handled at the velocity level, which means that position and force constraints are handled by the use of controllers that gives a velocity that is the actual constraint. The implementation has been performed with Matlab/Simulink, and real-time executable code has been generated with the Real-Time Workshop toolbox.

A skill consists of a number of simple motions, and it is specified as a state machine. Each state defines kinematic chains and a specification of the constraints, i.e., controller parameters and setpoints. Transition conditions in the state machine consist, e.g., of force/torque and position thresholds. The software used for the implementation of the state machine is JGrafchart [8].

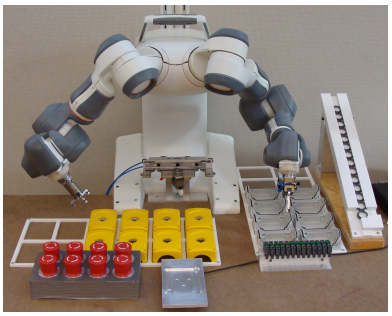

Fig. 1. A robot cell for assembling emergency stop buttons.

III. High-level programming

The high level programming interface is implemented as a plugin to ABB RobotStudio [1]. When creating the station, the objects such as the robot, workpieces, sensors, trays and fixtures, can be generated in the station or downloaded from KIF. The ontologies stored in KIF describe the objects. Each object lists its own properties, e.g. weight and dimensions. A physical object is characterized by its *object frame* and a number of *feature frames* related to the object frame. Geometrical constraints are expressed as relations between feature frames. These frames are later used to create the kinematic chains.

On the highest level, the task is represented as an assembly graph, which is a partially ordered tree of assembly operations [11]. Each assembly operation specifies the desired geometrical relations of the involved objects and the skill type. The assembly operations are subgoals, and the root node represents the final goal of the task.

Before the task specification is translated into executable code, it is subjected to a number of checks verifying consistency, correctness and completeness. As an example, a KIF service checks that the station fulfills the device requirements of each used skill. It loads the station description with all its objects and the assembly graph. In the ontology, each skill description contains device requirements, pre- and postconditions. Initially, the service matches the device requirements

to the objects in the station. If all device requirements are fulfilled, the service will continue by suggesting a sequence of skills so that no preconditions are violated.

To concretize, in a task where an emergency stop button box is assembled, see Fig. 1, one assembly operation consists of snapping a switch into the bottom of the box. This skill has the type SnapFit and requires 1) the box to be fixed, 2) the force to be measured in positive z-direction of the fixed frame, and 3) the switch to be located above the box when starting the motion. When the service evaluates this skill for a particular station, it will match the objects in the station to the requirements, e.g. verify that a force sensor and fixture exist and that the sensor is either aligned with the fixture or mounted on the gripper.

When analyzing the sequence, the preconditions of a skill have to match the postconditions of the previous skills. A service suggests skills that fulfill missing conditions if such skills exist. In particular picking, placing and moving objects are simple to add. However, we do not focus on optimizing the task execution, but only on guaranteeing its executability.

Another, even more high-level functionality, is the natural language support. The user describes the task using natural language sentences. They are parsed and semantically analyzed [2] in order to identify skills and the objects to be manipulated. These skills are combined into a task sequence.

IV. Ongoing and future work

We are currently working on generating the control programs from the specifications. Currently motions are specified using primitive motions like translate or rotate, expressed using geometrical or velocity constraints between feature frames. We are extending the constraint vocabulary to include sensor conditions, such as force and time.

Given such a specification, we can extract the kinematic chain of the task by taking the coordinate systems of the involved feature frames and use them to form a closed kinematic chain. This chain is the list of simple transformations that is used to compute the low-level motion control. Together with the motion parameters, this forms one state in the skill state machine. A complete skill consists of a sequence of such motions.

The generated state machine describes the nominal behavior of the skill. If needed, the resulting state machine can be modified in the state machine editor [8]. In the future, the system will, at least semi-automatically, be able to extend the state machine with typical error handling procedures. We consider error handling one of the important aspects regarding the usability of our system.

V. Related Work

Huckaby and Christensen [7] has created a taxonomy for task modeling using skill primitives and control skills. Reusable skill or manipulation primitives are a convenient way of hiding the detailed control structures [10]. Kresse and Beetz [9] describe an approach to map high-level constraints to control parameters in order to flip a pancake.

Acknowledgments

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

References

[1] ABB RobotStudio. http://www.abb.com/product/seitp327/78fb236cae7e605dc1256f1e002a892c.aspx, 2013.

[2] A. Björkelund, B. Bohnet, L. Hafdell, and P. Nugues. A high-performance syntactic and semantic dependency parser. In *Proc. COLING '10*, pages 33–36. ACL, 2010.

[3] A. Björkelund, L. Edström, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. G. Robertz, D. Storkle, A. Blomdell, R. Johansson, and et al. On the integration of skilled robot motions for productivity in manufacturing. In *Proc. IEEE ISAM 2011*, pages 1–9, 2011.

[4] A. Blomdell, G. Bolmsjö, T. Brogårdh, P. Cederberg, M. Isaksson, R. Johansson, M. Haage, K. Nilsson, M. Olsson, T. Olsson, A. Robertsson, and J. Wang. Extending an Industrial Robot Controller–Implementation and Applications of a Fast Open Sensor Interface. *IEEE Robotics & Automation Magazine*, 12(3):85–94, 2005.

[5] A. Blomdell, I. Dressler, K. Nilsson, and A. Robertsson. Flexible Application Development and High-performance Motion Control Based on External Sensing and Reconfiguration of ABB Industrial Robot Controllers. In *Proc. ICRA 2010 Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications*, pages 62–66, 2010.

[6] De Schutter, J. and De Laet, T. and Rutgeerts, J. and Decré, W. and Smits, R. and Aertbeliën, E. and Claes, K. and Bruyninckx, H. Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *Int. J. Robotics Research*, 26(5):433–455, 2007.

[7] J. Huckaby and H. I. Christensen. A Taxonomic Framework for Task Modeling and Knowledge Transfer in Manufacturing Robotics. In *Proc. 26th AAAI Cognitive Robotics Workshop*, pages 94–101, 2012.

[8] JGrafchart. http://www.control.lth.se/Research/tools/grafchart.html, 2012.

[9] I. Kresse and M. Beetz. Movement-aware Action Control Integrating Symbolic and Control-theoretic Action Execution. In *Proc. ICRA 2012*, pages 3245–3251, 2012.

[10] 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*, pages 293–313. Springer, 2010.

[11] J. Malec, K. Nilsson, and H. Bruyninckx. Describing assembly tasks in a declarative way. In *ICRA 2013 WS Semantics, Identification and Control of Robot-Human-Environment Interaction*, 2013. Accepted.