SkiROS

A four tiered architecture for task-level programming of industrial mobile manipulators

Rovida, Francesco; Schou, Casper; Andersen, Rasmus Skovgaard; Damgaard, Jens Skov; Chrysostomou, Dimitrios; Bøgh, Simon; Pedersen, Mikkel Rath; Grossmann, Bjarne; Madsen, Ole; Krüger, Volker

Publication date:
2014

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Rovida, F., Schou, C., Andersen, R. S., Damgaard, J. S., Chrysostomou, D., Bøgh, S., Pedersen, M. R., Grossmann, B., Madsen, O., & Krüger, V. (2014). *SkiROS: A four tiered architecture for task-level programming of industrial mobile manipulators*. Paper presented at International Workshop on Intelligent Robot Assistants, Padova, Italy.

SkiROS: A four tiered architecture for task-level programming of industrial mobile manipulators

Francesco Rovida[1], Casper Schou[2], Rasmus Skovgaard Andersen[2],
Jens Skov Damgaard[2], Dimitrios Chrysostomou[2], Simon Bøgh[2],
Mikkel Rath Pedersen[1], Bjarne Großmann[1], Ole Madsen[2], and Volker Krüger[1]

[1] Robotics, Vision and Machine Intelligence Lab,
Aalborg University of Copenhagen
AC Meyers Vaenge 15, DK-2450 Copenhagen
[2] Robotics & Automation Group,
Department of Mechanical and Manufacturing Engineering,
Fibigerstraede 16, DK-9220 Aalborg East

Abstract. During the last decades, the methods for intuitive task level programming of robots have become a fundamental point of interest for industrial application. The paper in hand presents *SkiROS* (Skill-based Robot Operating System) a novel software architecture based on the skills paradigm. The skill paradigm has already been used and tested within the FP7 project TAPAS, and we are going to use it in several new FP7 projects (CARLOS, STAMINA, ACAT). It facilitates task-level programming of mobile manipulators by providing the robot with a set of movement primitives, skills and tasks. This hierarchy brings many advantages, where the most relevant is the separation of control in the layers of hardware abstraction(proxy), multi-sensory control(primitive), object-level abstraction (skill) and planning (task). The definition and the clear division in different abstraction levels allows the implementation of a flexible, highly modular system for the development of cognitive robot tasks.

Keywords: Mobile manipulation, Software architecture, Task-level programming

1 Project idea and objectives

The task-level programming is a way of programming a robot on an abstract level. This promises to decrease the amount of competences necessary to deal with complex robots and it is a powerful way to speed up the programming process and the code reuse. This is realized creating an abstraction which hide the complexity of the lower layers and allows to focus on the task itself. Task-level programming specifies what the robot should do in terms of actions on the objects involved in the task and not necessarily to the full extent how it should be achieved.

In earlier papers [1, 4] we defined tasks as a sequence of skills where the skills provide the atomic programming blocks that a) unify sensing and robot action for interaction with the environment [4] and b) are motivated by human intuition [1]. The skills are comparable to the manipulation primitive nets, described best in [2]. These nets are sequences of manipulation primitives, with simple decisions based on the outcome of each single manipulation primitive in the net. In these nets each manipulation primitive needs its own explicitly specified parameters, e.g. a specic force or velocity in a specic direction. This , however, makes this particular implementation unsuitable for robotics novices in a factory hall. Instead, our skills propose a method for specifying only high-level parameters, while low-level parameter for the lower level primitives are mostly inferred through reasoning. This also brings the skill names more close to human intuition, e.g., "pick" and "place".

The research on tiered robotic architectures started in '80 [3], and still no standard exists. A major research effort to form a generalized robot middleware is the Robotic Operating System (ROS) [6] where there are some projects in the similar direction as *SkiROS*. Two examples are *moveIT* [7] and *ROSco* [5]. The former is focused on industrial manipulators and embed some good tools, like collision-detection, arm motion planning and execution, etc. *ROSco* is focused on home environment robots, it is based on "behaviors", which are Hierarchical Finite State Machine (HFSM) composed of generic, parameterized building blocks. [5]

SkiROS aims to create a system capable of extending the capabilities of moveIT with high level ones similar to those realized within *ROSco*. In that sense, the final objective of *SkiROS* is to give us a flexible platform, integrating the state of the art in the task-level programming. It will be used as a tool to make research and aims to go beyond the state of the art in terms of robustness, sustainability and execution speed.

The main structure of *SkiROS* is illustrated in Figure 1. It is divided into 4 layers, each one with separate interfaces for specific usage and development. This structure allows the development of key features and algorithms as independent

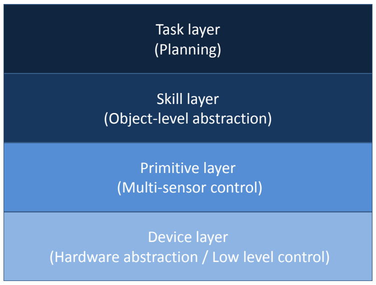

Fig. 1. The proposed layers in *SkiROS*, with the related concerns. The *Device Layer* realizes the hardware abstraction, the *Primitive Layer* merges the hardware control with sensors feedback. The *Skill Layer* realizes a concatenation of primitive device properties, based only on semantic information. Finally, the *Task Layer* plans the skill sequence to achieve the final goal.

plugins in a modular way without interfering with the rest of the core system. Thus, the sharing, reuse and collaborative development of further plugins will be an straightforward task.

2 ROS benefit for the project

SkiROS targets an ambitious goal which shares some of its main functionalities with ROS. In particular:

- ROS interprocess communication structure (topics, services and actions), a fundamental feature to modularize the system.
- ROS pluginlib to easily insert new functionalities in the system layers meanwhile decouple them from the system itself. Note that the use of pluginlib bind us, for now, to code entirely in C++.
- ROS drivers to interact with sensors.
- URDF/SRDF standard to describe the robot structure.
- Rviz and Gazebo used to visualize and/or simulate robot environments.
- ROS moveIT package offering many fundamental functionalities in the pick-place pipeline.
- ROS-industrial package offers drivers for industrial arms and grippers.

3 Contribution to ROS-I

The contribution of *SkiROS* to ROS-industrial is a trailblazing investigation of the first ROS architecture capable to allow easy and independent implementation/use of functionalities on different levels of abstraction, from hardware control to complex tasks. In particular it will:

- Bring forward the hardware abstraction standard already started for arms, but not for other devices.
- Investigate a primitive standard, where can be realized different constrained and sensor-based control strategies and sensor data analysis.
- Investigate skill standard to realize a reliable concatenation of primitives using semantic information.
- Investigate a task-planning system.

The system is going to extensively used and validated into real industrial environment in terms of robustness , sustainability and intuitiveness within at least three European projects, *CARLOS, STAMINA* and *ACAT*.

4 Use cases

SkiROS will target numerous use-cases in several projects. Some of them identified within the *STAMINA* project are presented below:

- Use-Case 1: De-palletizing, pick objects placed with a rough order in a container.
- Use-Case 2: Bin-picking, pick objects randomly placed into a container.
- Use-Case 3: Kitting, collecting the components into a joint kit.

The use-cases require the robot to fetch the right parts from different types storage, pallets and containers, and place them into the kits. This should be accompanied with a *built-in* quality check of the parts. This require skills such as bin-picking, navigation, inspection and placing. Based on the use-case specs provided the robot should be able to find out autonomously how it can solve the task, ideally without the need of additional programming.
STAMINA address the call ICT-2013.2.2 ”Robotics use cases & Accompanying measures project” and it is a 3 years length project.

5 Conclusion and future works

In this paper we briefly presented our task-level programming paradigm, based on skills. The skill based system has already been used and tested extensively in the context of industrial settings of the EU FP7 Project TAPAS [4]. It was a robust system, which simplified the task planning of mobile manipulators and while its general performance can be considered adequate, there is plenty of room for improvements. As a consequence, we have now started developing a reference implementation under the name *SkiROS*, a robot software architecture tailored for the specific needs of industrial projects, which implements the skill paradigm with a modular and flexible way. Its main characteristic is the realization of complex robotic applications including multi-robot cooperation and integration of sensors in a plethora of robot operations. It will allow to develop, compile, share and use code easier, on every layer of abstraction.

References

1. Simon Bø gh, Oluf Skov Nielsen, Mikkel Rath Pedersen, Volker Krüger, and Ole. Madsen. Does your robot have skills? *The 43rd Intl. Symp*, 2012.
2. Bernd Finkemeyer, Torsten Kröger, and Friedrich M. Wahl. Executing assembly tasks specified by manipulation primitive nets. *Advanced Robotics*, 19(5):591–611, January 2005.
3. David Kortenkamp and Reid Simmons. Robotic systems architectures and programming. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 187–206. Springer Berlin Heidelberg, 2008.
4. Ole Madsen, Simon Bø gh, Casper Schou, Rasmus Skovgaard Andersen, Jens Skov Damgaard, Mikkel Rath Pedersen, and Volker Krüger. Integration of two autonomous mobile manipulators in a real - world industrial setting. 2013.
5. Georgia Institute of Technology. Ros commander (rosco): Behavior creation for home robots, [Online]. https://www.hsi.gatech.edu/hrl/project_rosco.shtml.
6. Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.
7. Ioan A. Sucan and Sachin Chitta. Moveit! [available] http://moveit.ros.org, 2013.