A User-friendly Interface to Control ROS Robotic Platforms

Ilaria Tiddi[1,2], Emanuele Bastianelli[2], Gianluca Bardaro[3], Enrico Motta[2]

[1] Department of Computer Science, VU University Amsterdam, NL
ilaria.tiddi@vu.nl
[2] Knowledge Media Institute, The Open University, UK
{name.surname}@open.ac.uk
[3] Dip. di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy
gianluca.bardaro@polimi.it

Abstract. In this work we present a user interface assisting non-expert users to design complex robot behaviours, hence facilitating the deployment of robot-integrated applications. Due to the increasing number of robotic platforms available for commercial use, robots are being approached by users with different backgrounds, whose interests lie in the high-level capabilities of the platforms rather than their technical architecture. Our interface allows non-experts to use a robot as a development platform, i.e. giving it high-level commands (e.g. autonomous navigation, vision, natural language generation) by relying on a basic ontology of high-level capabilities mapped on the robot low-level capabilities (e.g. communication, synchronisation, drivers) exposed by the most common robotic middleware ROS. To show how our work can sensibly reduce the efforts required for having robots achieving basic tasks, we propose a live demonstration in which the ISWC audience will remotely program a robot to achieve different tasks without any previous knowledge of ROS.

Keywords: Knowledge Representation· Robots· Ontology Engineering

1 Introduction

The goal of our work is to facilitate the usability of robotic platforms to non-expert users, hence simplifying the time-consuming process of configuring and programming robotic platforms to integrate in real-world applications. In this demo, we present a user-friendly interface that we designed and implemented in order to allow users to make robots of different types and capabilities achieve basic tasks without having any previous knowledge of robot programming.

Our motivation stems from the growth of robotic platforms available for commercial use, which parallels to the increasing number of non-technical users that approach robots to exploit them as development platforms for their own purposes. For example, we are looking at integrating robots in a number of smart city applications, such as parking monitoring, building surveillance and garbage collection. Without the technical knowledge or the expertise of dedicated robot developers, we are limited to exploit the capabilities allowed by the commercial platforms (e.g. using a remote-controlled drone to record photos or videos),

while more advanced usages supported by those systems could be exploited (e.g. programming the drone to autonomously survey an area).

These limitations could be overcome by integrating a middleware between the robots and an application, such as the ROS framework (Robot Operating System[4]) promoted by the robotics community with the aim to relieve developers from the management of low-level components. From an end-user perspective, ROS remains a low-level technical platform directly above a robot's hardware components. This means that programming a robot to achieve high-level tasks is still a time-consuming process, requiring advanced knowledge of robot-specific ROS components, the ROS framework, and robotics in general.

Ontologies, on the other hand, have proven to be successful in facilitating use of complex systems, e.g. sensor networks, smart products and smart homes [1–3]. Starting from the idea that an ontological representation of robot capabilities could help abstracting from the low-level implementation of the robot, we developed in [4] an ontology-based system to derive high-level capabilities (such as autonomous navigation, vision, natural language understanding) from the ROS components (managing low-level capabilities such as communication, synchronisation, driver control). This paper then focuses on the presentation of the user interface and the live demonstration of the tool. During the demonstration, users will be remotely programming robots without previous expertise to achieve specific, purpose-made tasks in our department (the Knowledge Media Institute). We will show how using an ontological abstraction is not only beneficial to reduce learning times and efforts, but also to increase the interoperability of robotic platforms.

2 Exposing Robot Capabilities

The user interface, presented in Figure 1, is composed of three blocks: the Capability Panel (left block), the Construct Panel (right block) and the main Program Panel (middle block). All demo material, i.e. a video of the process, the ontology and code to perform the tasks, is available online[5].

Capability Panel. This panel offers the users the list of capabilities abstracted from a robotic platform to which the system is connected. We refer the reader to [4] for the in-depth description of the ontology-based system lying behind the interface. Here, we limit ourselves in mentioning that the system is based on a formal representation of the ROS architecture (in short, `:Node`s that exchange `:Message`s routed via specific `:Topic`s[6]), and that such communication exchanges mapped into specific robot `:Capability`(ies), organised into as a taxonomy whose highest classes are `:Sensing`, `:Movement` and `:RobotKnowledge`. Based on the ontology and the robot running on ROS, an Analyser module scans all active ROS components to determine the capabilities processed by the robot, based on the high-to-low-level capability mappings described in the knowledge

[4] http://www.ros.org/
[5] https://tinyurl.com/ybqg7om5
[6] http://wiki.ros.org/ROS/Concepts

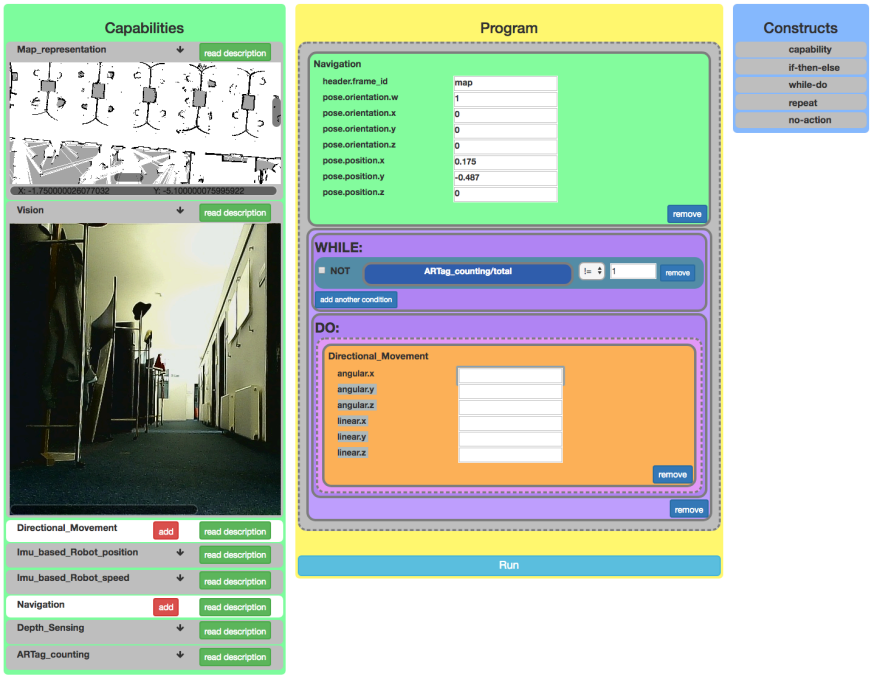

Fig. 1: The interface exposes the capabilities that are available to the robot in use (left), the blocks that can be used to instruct it (right), and the main program defined by the user (middle).

base. The retrieved list of capabilities is then shown to the user, for he/she to create an ad-hoc program. Note that capabilities come in two modalities: *read* capabilities, giving information about the robot (e.g. the *Vision* capability offering data from a camera), and *write* capabilities that expect some inputs (e.g. *Navigation* expects a position in space).

Construct Panel. The Construct Panel shows the user the set of programming constructs that he/she can choose to create the program to instruct the robot with. A program can be created as an imperative programming language, in which the atomic blocks are invocations of the available robot capabilities on the left panel (using the *add* button), and a set of conditional operators, such as *if-then-else*, *while-do* and the *repeat* statement. An additional *no-operation* statement can be employed to perform empty operations. The parameters of a capability can be used in the conditions, so to exploit any robot output to drive the program flow (e.g. moving forward until an object is detected).

Program Panel. The Program Panel in the middle of Figure 1 shows the user the program he/she has created through constructs and capabilities of the respective panels. Once the user is satisfied with the set of commands, they are sent to the robot through a Dynamic module. If a read capability is requested, the Dynamic

module relays to the interface the messages received from the robot. If a write capability is requested, the Dynamic module reads the parameters from the interface, structures them in a message and then sends them to the robot, which consequently performs the operations.

3 Planned Demonstration

We plan a live demonstration where the audience will be able to remotely instruct a ground wheeled robot operating in an office environment (the Knowledge Media Institute) to solve a number of high-level tasks using the presented interface, i.e. creating a program allowing the robot to achieve a specific task. We plan tasks to be of increasing difficulty for the audience to get familiar with the interface, namely:

1. implementing a *single movement* behaviour, e.g. "go to Enrico Motta's room";
2. implementing a *complex behaviour* as a sequence of commands, e.g. "navigate to Enrico Motta's room after going to the Podium (the seminar room)";
3. implementing a sequence of actions with a *termination condition*, e.g. "patrol three rooms of KMi";
4. implementing an *object detection* detection behaviour, e.g. "patrol KMi rooms until you find Enrico Motta".

The goal of the demonstration will be to show users without previous robotics expertise that efforts and time required to program robots can be sensibly reduced through relying on ontological knowledge, hence making a step towards facilitating the development of applications in which robots are integrated. If time and conditions allow, we also plan to collect quantitative and qualitative data about the audience's performance (e.g. time taken to perform an action or robot precision) in view of a larger and thorough evaluation of our tool.

In the future, we will be offering a set of fine-grained APIs for a more high-level communication with the robot, following the structure of the user interface presented in the paper.

References

1. Chen, L., Nugent, C., Mulvenna, M., Finlay, D., Hong, X.: Semantic smart homes: towards knowledge rich assisted living environments. Intelligent Patient Management pp. 279–296 (2009)
2. Corcho, O., García-Castro, R.: Five challenges for the semantic sensor web. Semantic Web **1**(1, 2), 121–125 (2010)
3. Sabou, M., Kantorovitch, J., Nikolov, A., Tokmakoff, A., Zhou, X., Motta, E.: Position paper on realizing smart products: Challenges for semantic web technologies. In: Proceedings of the 2nd International Conference on Semantic Sensor Networks-Volume 522. pp. 135–147. CEUR-WS. org (2009)
4. Tiddi, I., Bastianelli, E., Bardaro, G., d'Aquin, M., Motta, E.: An ontology-based approach to improve the accessibility of ros-based robotic systems. In: Proceedings of the Knowledge Capture Conference. p. 13. ACM (2017)