Self-reconfigurable Control Architecture for Complex Robots

Jan Hartmann[1], Walter Stechele[2], and Erik Maehle[1]

[1]Institute for Computer Engineering, Universität zu Lübeck
Ratzeburger Alle 160, 23562 Lübeck

[2]Institute for Integrated Systems, Technische Universität München
Arcisstraße 21, 80290 München

{hartmann,maehle}@iti.uni-luebeck.de, walter.stechele@tum.de

Abstract: Advanced robot systems need to carry out increasingly complex task sets. However, they are typically optimized to a very restricted set of tasks and environments to solve demanding problems. This work will therefore propose a self-reconfigurable software and hardware architecture in order to enable the dynamic optimization of a robot system depending on the current situation, i.e. the current task, robot state, and environment. The proposed framework is based on organic computing principles and unsupervised machine learning techniques. It further uses dynamically reconfigurable Field Programmable Gate Arrays (FPGA) as hardware accelerators.

1 Introduction

In recent years, autonomous mobile robots have gained in practical importance. In industrial applications, they are, for example, used in search and rescue missions, logistics, or agriculture. In the consumer market, with various kinds of cleaning robots available, the goal for the near future will be the development of versatile personal service robots. To solve the increasingly complex tasks in both industrial and consumer markets, advances in sensor technology and algorithms generate ever more complex robot systems. Current state-of-the-art service robots, such as the PR2 from Willow Garage[1], already posses a multitude of sensors and actuators to solve typical tasks at home.

Experiences in a "robot butler" scenario were described in [BRJ+11] by analyzing the task of fetching a specific drink in detail. Even with such a restricted scenario, two main challenges in advanced mobile robotics become evident. First, several complex algorithms are utilized to accomplish a simple task, e.g. collision avoidance, identification of the specific drink to fetch, or face recognition. As there are several solutions to problems such as navigation or object detection, each with different strengths, here, the choice of an optimal set of algorithms for the given task and subsequently the choice of optimal parameters for these algorithms remain an open issue. Second, the realtime processing

---

[1] `http://www.willowgarage.com/pages/pr2/overview`

of the available sensor data, especially 3D-depth information and camera images, with general purpose CPUs is not possible, given the complexity of the utilized algorithms.

Ideally, a mobile service robot would dynamically and autonomously choose algorithms as well as parameters from a set of solutions based on the current situation, i.e. current task, state of the robot, and environment. Algorithms would be configured similarly in order to satisfy realtime constraints while providing high accuracy. This work will therefore present a unified architecture for the dynamic self-reconfiguration of robot systems. While simple physical reconfiguration and parameter reconfiguration have been investigated for walking machines in previous works, in this work the focus will explicitly lie on the complex robot systems of state-of-the-art mobile robots to satisfy such demands as described above.

The control unit of the target robot system is now assumed to consist of a multicore CPU and a dynamically reconfigurable Field Programmable Gate Array (FPGA) coprocessor, where the FPGA guarantees the realtime capability of the system via hardware acceleration and the CPU enables an easy migration of existing algorithms. Three different methods will be presented in the remainder of this work to ensure the optimal utilization of the control unit in different situations. First, the algorithm to solve a specific problem is automatically chosen. Second, the parameters of algorithms are automatically tuned to the current situation. Third, algorithms are automatically distributed to the CPU and FPGA to ensure the best runtime with the current set of algorithms. These methods are subsumed under the term *self-reconfiguration*.

The remainder of this paper will be structured as follows. First, the methods used in the approach presented in this work are shown in Secs. 2 and 3. Then, the target algorithm to be used as a demonstrator will be shown in Sec. 4. Future work will finally be discussed in Sec. 5.

2 Organic Robot Control Architecture (ORCA)

The Organic Robot Control Architecture (ORCA) forms the basis for a unified implementation of the methods for self-reconfiguration. ORCA is a modular, hierarchical architecture, based on the RCA-architecture [ALBD02], which is conductive to the emergence of self-x properties in a robot system [BMG$^+$11]. ORCA distinguishes between three main components. Basic Control Units (BCU) ensure the functionality of the robot under normal conditions. Organic Control Units (OCU) observe the BCUs and may change e.g. parameters of a BCU. Health Signals hierarchically describe the state of the robot and may detect anomalous behavior. The OCUs now enable the robot to react to changes in the environment or in the physical properties of the robot itself.

The hexapod walking robot OSCAR was developed as a demonstrator for the ORCA architecture [MBG$^+$11]. With a distributed and organically inspired gait control system, OSCAR is able to walk robustly in rough terrain. It can detect anomalies and react accordingly to ensure that it accomplishes its task in the still best possible way, for example by adjusting the gate in rough terrain [AHHM12] or by amputating legs in case of hardware

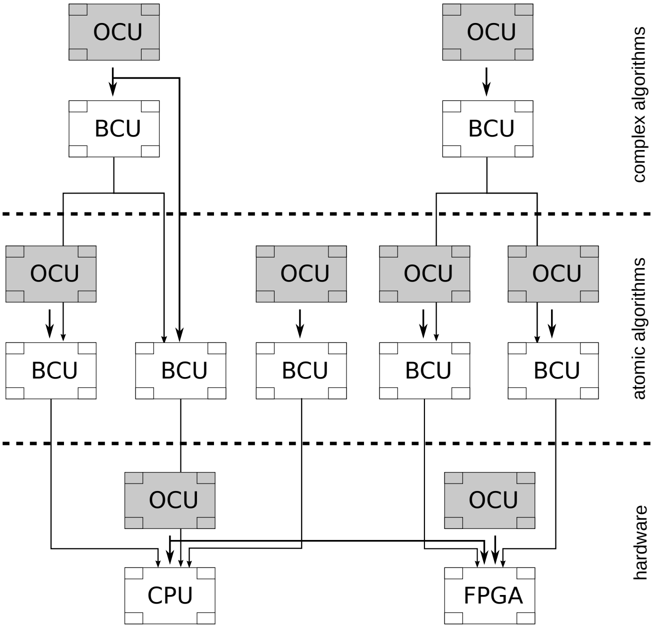

Figure 1: The Organic Robot Control Architecture (ORCA) as shown in a simplified form without sensor and actor paths.

failures [JM10] and then choosing a respective path [MM11]. The distributed network of BCUs and OCUs lead to an emergent walking pattern, similar to those that can be found in real insects.

In this work, the ORCA architecture will be expanded to accomplish similar effects on the self-reconfigurable target platform. The complex software system is therefore divided into simple *atomic algorithms*, as shown in Fig. 1. Each atomic algorithm is implemented as a BCU and may, in the expanded ORCA architecture, run either on the conventional CPU or the FPGA coprocessor. OCUs now control the self-reconfiguration of the robot system. They choose the BCUs (and thus algorithms) to be used, adjust BCU parameters, and determine which atomic algorithm is executed on the FPGA coprocessor. Universal criteria, such as accuracy and runtime, guide the distributed self-reconfiguration process to ensure globally optimal solutions for the purpose of a realtime capable, dynamically adjustable robot system.

3 OCU Implementation

Organic Control Units are implemented in a generic way by using Learning Classifier Systems (LCS, [Wil95]). LCS are a class of rule-based learning algorithms closely related to genetic algorithm and reinforcement learning where a set of rules is altered in order to achieve the best possible classifier. A rule consists of conditions, actions, and a fitness value. The conditions determine which rules apply in a given situation. If several rules match the current situation, a rule is selected randomly with a probability relative to the fitness value. The actions associated to the selected rule are then applied.

Learning is now performed in two different ways. On the one hand, the fitness value is updated based on a reward function, so that rules with associated actions that perform well are selected more frequently. This corresponds to reinforcement learning. On the other hand, rules are mutated using generic operators to find a good representation of the rule space.

Learning Classifier Systems were chosen as the means of OCU implementation for several reasons. They are powerful enough to model self-reconfiguration in such a complex system. At the same time, rules may provide a human-readable representation of the system which will help to construct different rule sets for various problems more easily.

For LCS-OCUs in the robotic system, conditions now reflect information about the robot's inner state, e.g. what kind of sensors are available, as well as information about the environment, e.g. the general kind of environment (indoor or outdoor, office or private home) or the time of day. Actions either determine which algorithms to use or change parameters of an algorithm. OCUs on the hardware level have the special task of distributing atomic algorithms to either the CPU or FPGA. A hardware implementation further governs the dynamic reconfiguration of the FPGA. The reward function now primarily determines the system behavior. Rewards can be given depending on the target system behavior, e.g. for high accuracy and low runtime.

4 Demonstration Algorithm and Initial Results

The approach to self-reconfigurable control architectures will be demonstrated using the Visual Simultaneous Localization and Mapping (VSLAM) problem [EHE$^+$12, SDMK11, KHFM13]. VSLAM describes the estimation of the robot position in an unknown environment while constructing a representation of the environment (map) using cameras as the main sensors. VSLAM is a central problem of mobile robots and was chosen as demonstrator, on the one hand, because many different, typically computationally expensive algorithms are used to solve the VSLAM problem while, on the other hand, results very much depend on the environment and algorithms typically have to be manually tuned to provide good results.

An example application of the presented self-reconfiguration approach to a typical VSLAM algorithm is shown in Fig. 2. The algorithm is factored into three main modules. A SLAM backend manages and optimizes the map, the odometry incrementally estimates the robot movement, and the registration corrects the accumulating odometry error by matching current sensor information to the map.

OCUs observe the odometry and registration on the algorithm level and the CPU and FPGA on the hardware level. LCS rule conditions may now specifically reflect the status information that affects the VSLAM algorithm, e.g. the robot movement speed, or visual

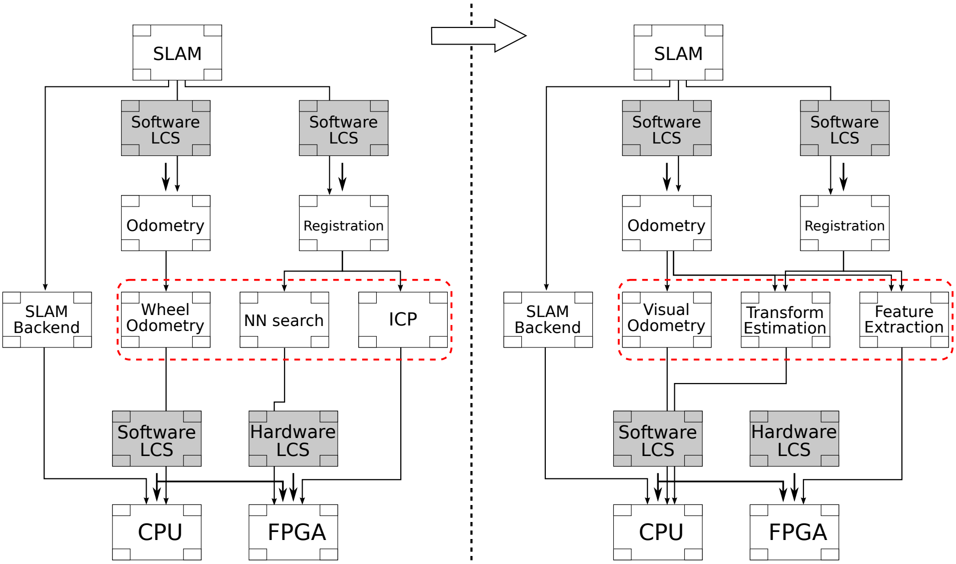

Figure 2: Illustration of the self-reconfiguration for a typical VSLAM algorithm.

cues about the environment, e.g. amount of texture and descriptiveness. Actions reconfigure the VSLAM algorithm. The odometry may, for example, be implemented using rotary wheel encoders and the registration using the Iterative Closest Point (ICP) algorithm. If the environment changes, e.g. the robot moves in rough terrain and the wheel odometry becomes inaccurate, the OCUs may force the BCUs to switch to a visual odometry estimation using image feature extraction techniques. These may then be additionally used as a means of registration and accelerated on the FPGA to ensure the realtime capability. Parameters, e.g. the number of features, are similarly adjusted. Rewards are given based on the total runtime for a VSLAM update step and a measure of accuracy, e.g. how well the registration matches the map.

Initial experiments using a similar VSLAM system have been presented in [HSM12] where only the software reconfiguration was evaluated. Even here, a similar accuracy to a manually optimized configuration could be achieved while significantly decreasing the computational load.

5 Summary and Future Work

In this paper, a self-reconfigurable software and hardware architecture for mobile robots was introduced to adapt the algorithmic solution to complex tasks to the current robot state and environment. The Organic Robot Control Architecture (ORCA) forms the basis of the self-reconfigurable system, in which Basic Control Units (BCU), which ensure the basic functionality, are reconfigured by Organic Control Units (OCU), which are implemented using Learning Classifier Systems (LCS) and may choose suitable algorithms and adjust the parameters of these algorithms. Low level (atomic) algorithms are then similarly distributed to the compute platform which is assumed to consist of a conventional CPU and a Field Programmable Gate Array (FPGA) coprocessor.

The main goal of the self-reconfigurable robot architecture is to dynamically react to the current situation, i.e. robot state and environment, in order to achieve the best possible results according to user-defined criteria, e.g. low runtime and high accuracy. First results were obtained, in software only, for the Visual Simultaneous Localization and Mapping (VSLAM) Problem for which high accuracy could be achieved while significantly reducing the algorithm runtime.

The work on a complete self-reconfigurable architecture is at the very beginning, thus much work needs still to be done. The steps will be the transparent integration into the Robot Operating System (ROS)[2] which is currently the most popular robot framework. Here, the self-reconfiguration will be based on existing interfaces for the manual reconfiguration to an easy integration into existing projects. BCUs and OCUs will be implemented as ROS nodes, i.e. distinct processes and communicate via messages. We will further investigate the automatic initialization and the offline application of self-reconfiguration with ground truth data for an optimal initialization of LCS rule sets.

Finally, the VSLAM demonstration algorithm will be transferred to the target CPU/FPGA hybrid system for which e.g. the Xilinx Zync SoC[3] may be used. Computationally complex parts of the VSLAM algorithm will be provided with hardware acceleration on the FPGA to ensure the realtime capability.

6 Acknowledgments

This work was funded in part by the German Research Foundation (DFG) within priority program 1183 under grant reference MA 1412/8-2.

References

[AHHM12] A. Al-Homsy, J. Hartmann, and E. Maehle. Sandy Ground Detection for Hexapod Robots Based on Organic Computing Principles and Somatosensory Feedback. In *Proc. of the IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, pages 43–48, 2012.

[ALBD02] Jan Albiez, Tobias Luksch, Karsten Berns, and Rüdiger Dillmann. An activation based behaviour control architecture for walking machines. In *Proc. of the Seventh International Conference on Simulation of Adaptive Behavior (ICSAB)*, ICSAB, pages 118–126, 2002.

---

[2] `http://www.ros.org/wiki`

[3] `http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/index.htm`

[BMG$^+$11] W. Brockmann, E. Maehle, K.-E. Grosspietsch, N. Rosemann, and B. Jakimovski. ORCA: An Organic Robot Control Architecture. In C. Müller-Schloer, H. Schmeck, and T. Ungerer, editors, *Organic Computing - A Paradigm Shilft for Complex Systems*. Birkhuser-Springer, 2011.

[BRJ$^+$11] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mösenlechner, W. Meeussen, and S. Holzer. Towards Autonomous Robotic Butlers: Lessons Learned with the PR2. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5568–5575, 2011.

[EHE$^+$12] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An Evaluation of the RGB-D SLAM System. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

[HSM12] Jan Hartmann, Walter Stechele, and Erik Maehle. Self-Adaptation for Mobile Robot Algorithms Using Organic Computing Principles. In *International Conference on Architecture of Computing Systems (ARCS)*, pages 232–243, Prague, 2012.

[JM10] B. Jakimovski and E. Maehle. In situ self-reconfiguration of hexapod robot OSCAR using biologically inspired approaches. In Behnam Miripour, editor, *Climbing and Walking Robots*. InTech, 2010.

[KHFM13] Jan Helge Klüssendorff, Jan Hartmann, Dariush Forouher, and Erik Maehle. Visual Graph-Based SLAM and Visual Odometry using an RGB-D Camera. In *To Appear in Proc. of the 9th International Workshop on Robot Motion and Control (RoMoCo)*, 2013.

[MBG$^+$11] E. Maehle, W. Brockmann, K.-E. Grosspietsch, A. El Sayed Auf, B. Jakimovski, S. Krannich, M. Litza, R. Maas, and A. Al-Homsy. Application of the Organic Robot Control Architecture ORCA to the Six-legged Walking Robot OSCAR. In C. Mller-Schloer, H. Schmeck, and T. Ungerer, editors, *Organic Computing - A Paradigm Shilft for Complex Systems*. Birkhuser-Springer, 2011.

[MM11] R. Maas and E. Maehle. Fault Tolerant and Adaptive Path Planning for Mobile Robots Based on Health Signals. In *International Conference on Architecture of Computing Systems (ARCS), Workshop on Dependability and Fault-Tolerance*, pages 58–63, 2011.

[SDMK11] Hauke Strasdat, Andrew J. Davison, J. M. M. Montiel, and Kurt Konolige. Double window optimisation for constant time visual SLAM. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pages 2352–2359, 2011.

[Wil95] S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.