The Carologistics RoboCup Logistics Team 2016

Tim Niemueller[1], Tobias Neumann[2], Christoph Henke[3], Sebastian Reuter[3], Alexander Ferrein[2], Sabina Jeschke[3], and Gerhard Lakemeyer[1]

[1] Knowledge-based Systems Group, RWTH Aachen University, Germany
[2] MASCOR Institute, FH Aachen University of Applied Sciences, Germany
[3] Institute Cluster IMA/ZLW & IfU, RWTH Aachen University, Germany

Abstract. The Carologistics team participates in the RoboCup Logistics League for the fith year. As a medium complex domain balancing implementation effort and significance the league requires the development and integration of various software components. We outline the approach with an emphasis on the modifications required for the new game originally introduced in 2015 in terms of custom hardware and software components. Additionally, we report on our efforts regarding the development of the league itself like releasing our full software stack for the second year in a row as open source software.

The team members in 2016 are Alexander Ferrein, Mostafa Gomaa, Christoph Henke, Daniel Künster, Nicolas Limpert, Matthias Löbach, Victor Mataré, Tobias Neumann, Tim Niemueller, Sebastian Reuter, Johannes Rothe, David Schmidt, Sebastian Schönitz, and Frederik Zwilling.

1 Introduction

The Carologistics RoboCup Team is a cooperation of the Knowledge-based Systems Group, the IMA/ZLW & IfU Institute Cluster (both RWTH Aachen University), and the MASCOR Institute (FH Aachen UoAS). The team was initiated in 2012. Doctoral, master, and bachelor students of all three partners participate in the project and bring in their specific strengths tackling the various aspects of the RoboCup Logistics League (RCLL): designing hardware modifications,

Fig. 1. Final round of the RoboCup Logistics League 2015 in Hefei, China. Team Carologistcs (laptop on top) and Team Solidus (pink parts) are competing against each other.

developing functional software components, system integration, and high-level control of a group of mobile robots. Our approach to the league's challenges is based on a distributed system where robots are individual autonomous agents that coordinate themselves by communicating information about the environment as well as their intended actions.

Our team has participated in RoboCup 2012–2015 and the RoboCup German Open (GO) 2013–2015. We were able to win the GO 2014 and 2015 as well as the RoboCup 2014 and 2015 (cf. Figure 1) in particular demonstrating flexible task coordination, robust collision avoidance and self-localization. We have publicly released our software stack used in 2014 and 2015 in particular including our high-level reasoning components for all stages of the game[4] [1].

In the following we will describe some of the challenges originating from the new game play in 2015. In Section 2 we give an overview of the Carologistics platform and the changes that were necessary to adapt to the new game play. We continue highlighting our behavior components in Section 4 and our continued involvement for advancing the RCLL as a whole as well as the RoboCup Industrial umbrella league in Section 5 before concluding in Section 6.

1.1 RoboCup Logistics League 2016

The goal is to maintain and optimize the material flow in a simplified Smart Factory scenario. Two competing groups of up to three robots each use a set of exclusive machines spread over a common playing field to produce and deliver products (cf. [2,3,4]).

In 2015, the RCLL has changed considerably by introducing actual processing machines based on the Modular Production System (MPS) platform by Festo Didactic [3] as shown in Figure 2. For more details on the new game play we refer to [5]. In 2016, the rules have only been changed slightly to allow teams to adapt and new teams to participate. The first RCLL Winter School organized by our team has vastly helped in this effort (cf. Section 5.3).

Fig. 2. Carologistics Robotino approaching a ring station MPS.

With a higher variety of work orders, time consumption for the individual task is an issue and efficient scheduling of the tasks a necessity to produce higher values goods (cf. Figure 3). The RCLL will introduce bar codes on the work pieces to be able to recognize them and award points during productio and not only on delivery.

[4] Software stack available at `https://www.fawkesrobotics.org/projects/rcll2015-release/`

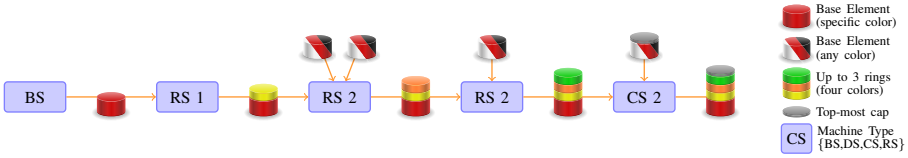

Fig. 3. Refinement steps for the production of a highest complexity product in the RCLL 2015 (legend on the right).

2 The Carologistics Platform

The standard robot platform of this league is the Robotino by Festo Didactic [6]. The Robotino is developed for research and education and features omnidirectional locomotion, a gyroscope and webcam, infrared distance sensors, and bumpers. The teams may equip the robot with additional sensors and computation devices as well as a gripper device for product handling.

2.1 Hardware System

The robot system currently in use is based on the Robotino 3. The modified Robotino used by the Carologistics RoboCup team is shown in Figure 4 and features two additional webcams, a RealSense dept camera and a Sick laser range finder. The webcam on the top is used to recognize the signal lights, the one below the number to identify machine markers, and the dept camera below the gripper is used to recognize the conveyor belt. We use the Sick TiM571 laser scanner used for collision avoidance and self-localization. It has a scanning range of 25 m at a resolution of $1/3$ degrees. An additional laptop increases the computation power and allows for more elaborate methods for self-localization, computer vision, and navigation.

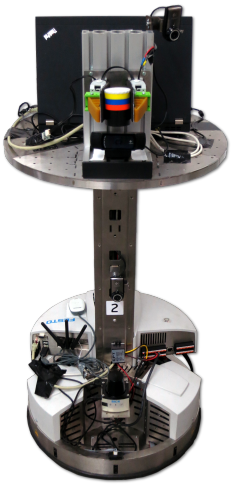

Fig. 4. Carologistics Robotino 2015/2016

Several parts were custom-made for our robot platform. Most notably, a custom-made gripper based on Festo fin-ray fingers and 3D-printed parts is used for product handling. The gripper is able to adjust for slight lateral and height offsets. The previously used servo motors have been replaced by stepper motors. These replacements were motoviated by increasing the positioning accuracy of the lateral axis. The motor is controlled with an additional Arduino board together with a motor shield. The acceleration of the motors follows an acceleration profile for smoothly increasing an decreasing the motor speed to avoid positioning errors. As no encoders are used a micro switch for initializing the lateral axis position is used.

2.2 Architecture and Middleware

The software system of the Carologistics robots combines two different middlewares, Fawkes [7] and ROS [8]. This allows us to use software components from both systems. The overall system, however, is integrated using Fawkes. Adapter plugins connect the systems, for example to use ROS' 3D visualization capabilities. The overall software structure is inspired by the three-layer architecture paradigm [9]. It consists of a deliberative layer for high-level reasoning, a reactive execution layer for breaking down high-level commands and monitoring their execution, and a feedback control layer for hardware access and functional components. The lowest layer is described in Section 3. The upper two layers are detailed in Section 4. The communication between single components – implemented as *plugins* – is realized by a hybrid blackboard and messaging approach [7]. This allows for information exchange between arbitrary components. As shown in Figure 5, information is written to or read from *interfaces*, each carrying certain information, e.g. sensor data or motor control, but also more abstract information like the position of an object. The information flow is somewhat restricted – by design – in so far as only one component can write to an interface. Reading, however, is possible for an arbitrary number of components. This approach has proven to avoid race conditions when for example different components try to instruct another component at the same time. The principle is that the interface is used by a component to provide state information. Instructions and commands are sent as messages. Then, multiple conflicting commands can be detected or they can be executed in sequence or in parallel, depending on the nature of the commands.

3 Advances to Functional Software Components

A plethora of different software components is required for a multi-robot system. Here, we discuss some components and advances of particular relevance to the game as played in 2016.

3.1 Basic Components

The lowest layer in our architecture which contains functional modules and hardware drivers. All functional components are implemented in Fawkes. For this

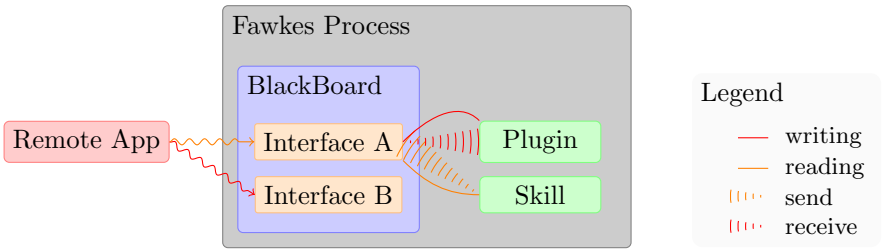

Fig. 5. Components communicate state data via interfaces stored in the blackboard. Commands and instructions are send as messages. Communication is universally shared among functional plugins and behavioral components.

year, we have developed a module for direct communication with the Robotino microcontroller, fully bypassing and eliminating the need for OpenRobotino. A major issue was that OpenRobotino has no concept of time and the age of sensor and odometry data could not be determined once it arrived in our system. Furthermore, a new velocity and acceleration controller has been implemented resulting in smoother driving.

3.2 MPS detection and approaching

The MPS machines are detected in two ways. Firstly with the tag placed on the machines, but also using a line fitting algorithm on the laser data. The approach the MPS during a game, the tag detection is used to validate the correct machine and for a first rough alignment. In a second step the laser lines are used for an more precised alignment especially regards the rotation. During the exploration phase both methods are used concurrently while searching for machines.

3.3 Light Signal Vision

A multi-modal perception component for robust detection of the light signal state on the field has been developed specifically for this domain [10]. It limits the search within the image by means of the detected position of the machine. This provides us with a higher robustness towards ambiguous backgrounds, for example colored T-shirts in the audience. Even if the machine cannot be detected, the vision features graceful degradation by using a geometric search heuristic to identify the signal, loosing some of the robustness towards the mentioned disturbances.

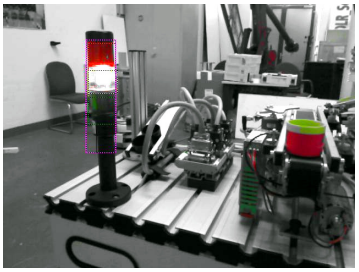

Fig. 6. Vision-based light-signal detection during production (post-processed for legibility).

3.4 Conveyor Belt Detection

The conveyor belts are rather narrow compared to the products thus require a precise handling. The tolerable error margin is in the range of about 3 mm. The marker on a machine allows to determine the lateral offset from the gripper to the conveyor belt. It gives a 3D pose of the marker with respect to the camera and thus the robot. However, this requires a precise calibration of the conveyor belt with respect to the marker. While ideally this would be the same for each machine, in practice there is an offset which

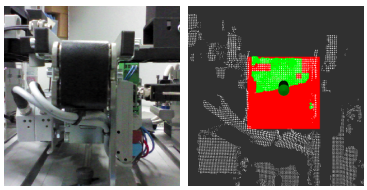

Fig. 7. Depth based conveyor belt detection. Left RGB picture, right point cloud with detected conveyor belt and its normal.

would need to be calibrated per station. Therefore we are using the approach described in 3.2 for a pre-alignment which is then improved with our depth based conveyor detection, where a point cloud from an Intel RealSens F200 camera is used to detect the conveyor. This is done by pruning the point cloud towards our region of interest by fusing the initial guess of the belt gathered by the machine position detected with the laser scanner. Afterwards a plane search is done to detect the precise pose of the front-plane of the conveyor belt and its normal.

3.5 Barcode Detection

Due to recent changes in the the RCLL, products can now be identified and tracked by means of a barcode. This allows for identifying each with an unique identifier. This enables the refbox to award points for intermediate production steps. We have implemented a detection component to be run on our robot. It allows to be more robust towards handling failures and to verify that the shared world model is consistent. Our implementation is based on the ZBar[5] OpenCV computer vision libraries. The detection is promising and we are working towards integration into our reasoning system. We will evaluate the system during the competition in Leipzig.

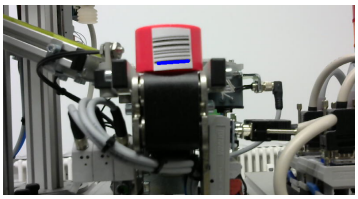

Fig. 8. Vision-based barcode detection (training images).

4 High-level Decision Making and Task Coordination

The behavior generating components are separated into three layers, as depicted in Figure 9: the low-level processing for perception and actuation, a mid-level reactive layer, and a high-level reasoning layer. The layers are combined following an adapted hybrid deliberative-reactive coordination paradigm.

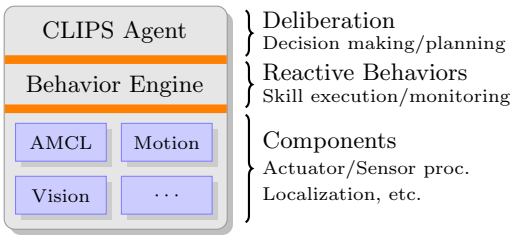

Fig. 9. Behavior Layer Separation

The robot group needs to cooperate on its tasks, that is, the robots communicate information about their current intentions, acquire exclusive control over resources like machines, and share their beliefs about the current state of the environment. Currently, we employ a distributed, local-scope, and incremental reasoning approach [5]. This means that each robot determines only its own action (local scope) to perform next (incremental) and coordinates with the others through communication (distributed), as opposed to a central instance which plans globally for all robots at the same time or for multi-step plans.

[5] `http://zbar.sourceforge.net/`

In the following we describe the reactive and deliberative layers of the behavior components. For computational and energy efficiency, the behavior components need also to coordinate activation of the lower level components.

4.1 Lua-based Behavior Engine

In previous work we have developed the Lua-based Behavior Engine (BE) [11]. It serves as the reactive layer to interface between the low- and high-level systems. The BE is based on hybrid state machines (HSM). They can be depicted as a directed graph with nodes representing states for action execution, and/or monitoring of actuation, perception, and internal state. Edges denote jump conditions implemented as Boolean functions. For the active state of a state machine, all outgoing conditions are evaluated, typically at about 15 Hz. If a condition fires, the active state is changed to the target node of the edge. A table of variables holds information like the world model, for example storing numeric values for object positions. It remedies typical problems of state machines like fast growing number of states or variable data passing from one state to another. Skills are implemented using the light-weight, extensible scripting language Lua.

4.2 Reasoning and Planning

The problem at hand with its intertwined world model updating and execution naturally lends itself to a representation as a fact base with update rules for triggering behavior for certain beliefs. We have chosen the CLIPS rules engine [12], because using incremental reasoning the robot can take the next best action at any point in time whenever the robot is idle. This avoids costly re-planning (as with approaches using classical planners) and it allows us to cope with incomplete knowledge about the world. Additionally, it is computationally inexpensive. More details about the general agent design and the CLIPS engine are in [13].

The agent for 2016 is based on the continued development effort of our CLIPS-based agent [4]. We have finalized generic world model synchronization capabilities that allow to mark specific facts in the fact base to be synchronized with other robots. A central robot dynamically determined through leader election is responsible for generating a consistent few and distributing it to all robots.

We have evaluated several different possibilities for the implementation of agent programs in the RCLL [13] and are making efforts towards a centralized global planning system.

4.3 Multi-robot Simulation in Gazebo

The character of the RCLL game emphasizes research and application of methods for efficient planning, scheduling, and reasoning on the optimal work order of production processes handled by a group of robots. An aspect that distinctly separates this league from others is that the environment itself acts as an agent by posting orders and controlling the machines' reactions. This is what we call

environment agency. Naturally, dynamic scenarios for autonomous mobile robots are complex challenges in general, and in particular if multiple competing agents are involved. In the RCLL, the large playing field and material costs are prohibitive for teams to set up a complete scenario for testing, let alone to have two teams of robots. Additionally, members of related communities like planning and reasoning might not want to deal with the full software and system complexity. Still they often welcome relevant scenarios to test and present their research. Therefore, we have created an *open simulation environment* [14,15] based on Gazebo.[6]

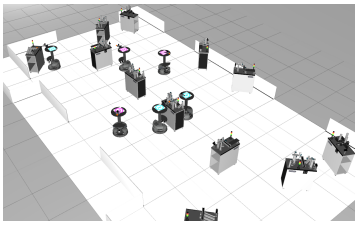

Fig. 10. Simulation of the RCLL 2015 with MPS stations.

5 League Advancements and Continued Involvement

We have been active members of the Technical, Organizational, and Executive Committees and proposed various ground-breaking changes for the league like merging the two playing fields or using physical processing machines in 2015 [2,3]. Additionally we introduced and currently maintain the autonomous referee box for the competition and develop the open simulation environment described above. We have also been a driving factor in the establishment of the RoboCup Industrial umbrella league [3]. It serves to coordinate and bring closer the efforts of industrially inspired RoboCup leagues. The first steps are the unification to a common referee box system (Section 5.1) and the introduction of a cross-over challenge (Section 5.4).

5.1 RCLL Referee Box and MPS Stations

The Carologistics team has developed the autonomous referee box (refbox) for the RCLL which was deployed in 2013 [2]. It strives for full autonomy on the

[6] More information, media, the software itself, and documentation are available at `http://www.fawkesrobotics.org/projects/llsf-sim/`

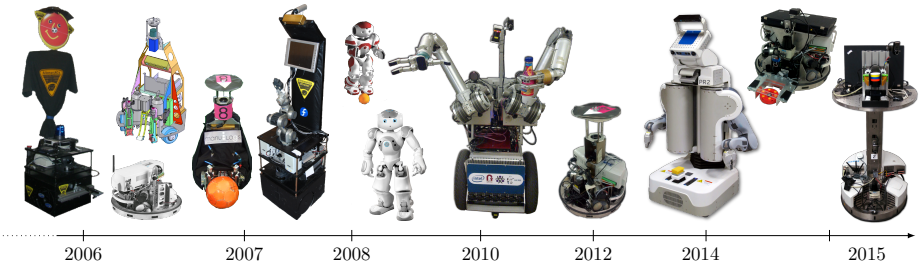

Fig. 11. Robots running (parts of) Fawkes which were or are used for the development of the framework and its components.

game controller, i.e., it tracks and monitors machine states, creates (randomized) game scenarios, handles communication with the robots, and interacts with a human referee. In 2014 the refbox has been adapted to the merged fields and two opposing teams on the field at the same time. We have also implemented a basic encryption scheme for secured communications.

In 2016, tracking of workpieces through barcode scanners on all stations must be implemented. Furthermore, there are efforts to create a general RoboCup Industrial Referee Box[7] based on the RCLL refbox. This would then be used in the RCLL as well as the RoboCup Industrial @Work League.

5.2 Public Release of Full Software Stack

Over the past nine years, we have developed the *Fawkes Robot Software Framework* [7] as a robust foundation to deal with the challenges of robotics applications in general, and in the context of RoboCup in particular. It has been developed and used in the Middle-Size [16] and Standard Platform [17] soccer leagues, the RoboCup@Home [18,19] service robot league, and now in the *RoboCup Logistics League* [15] as also shown in Figure 12.

The Carologistics are the first team in the RCLL to publicly release their software stack. Teams in other leagues have made similar releases before. What makes ours unique is that it provides a complete and *ready-to-run package with the full software* (and some additions and fixes) that we used in the competition in 2015. This in particular *includes* the complete *task-level executive* component, that is the strategic decision making and behavior generating software. This component was typically held back or only released in small parts in previous software releases by other teams (for any league).

[7] Project website: `https://github.com/robocup-industrial/rci-refbox`

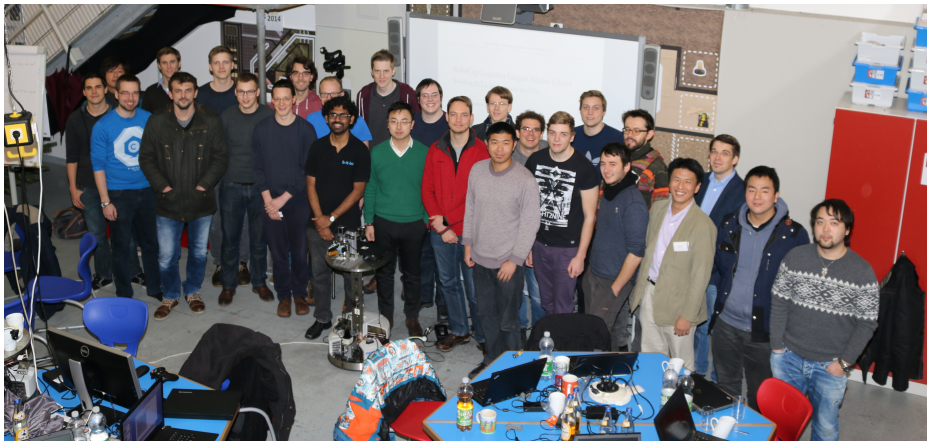

Fig. 12. Participants and the Carologistics at the RCLL winter school in 2015.

5.3 RoboCup Logistics League Winter School

In December 2015, the Carologistcs Team organized the week-long RoboCup Logistics League Winter School in Aachen. Within these days participants from Asia and Europe were introduced to the RoboCup Logistics League and the relevant components of the Fawkes software framwork. The winter school was structured by theoretical session where participants were members of the Carologistics presented topics like perception, navigation, simulation, and behavior design. Aftwards hands-on sessions with the Fawkes software framework deepened the theoretical sessions and were applied in simulation and in the real environment. This has been made possible through the generous support of Festo Didactic SE and a RoboCup Federation grant. Festo provided the full playing field for the winter school. Videos of all presentations and further information is available at `https://www.carologistics.org/winter-school-2015/`.

5.4 RoboCup Industrial Cross-over Challenge

As a first step for closer cooperation for the industry-inspired leagues under the RoboCup Industrial umbrella, together with stakeholders from the @Work league we have initiated a crossover challenge [20]. It describes several milestones towards closer cooperation. The task for the first year is depicted in Figure 13.

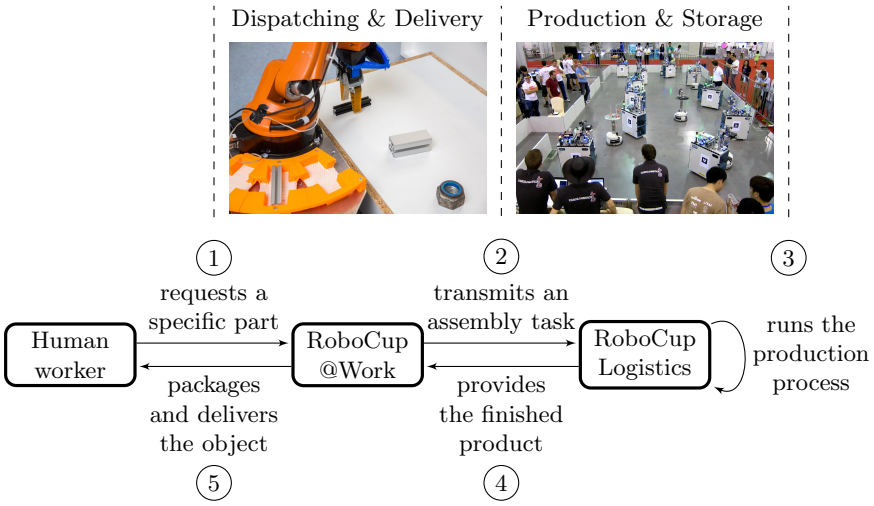

Fig. 13. Workflow of the cross-over scenario between @Work and the RCLL [20].

5.5 Planning Competition for Logistics Robots in Simulation

As an outcome of the presentation of the RCLL at the workshop on Planning in Robotics at the International Conference on Automated Planning and Scheduling (ICAPS) in 2015, a planning competition in simulation is being prepared.

At ICAPS 2016, a tutorial will be held to present the idea, gather feedback, and kickstart interested teams [21]. The particular challenges are to efficiently plan in short time with dynamic orders and to provide an effective executive to execute multi-robot plans. The competition will be based on the simulation developed by the Carologistics team. The idea is to foster collaboration and exchange among the planning and robotics communities. Further information is available at `http://www.robocup-logistics.org/sim-comp`.

6 Conclusion

In 2016, we have in further adapted to the new game. We upgraded our custom hardware gripper based on the feedback of the 2015 season, and further adapted and extended the behavior and functional components. We have also continued our contributions to the league as a whole through active participation in the league's committees, publishing papers about the RCLL, and initiating a crossover challenge under the RoboCup Industrial umbrella. The development of the simulation we initiated has been transferred to a public project where other teams have joined the effort and it is used in a spin-off simulation competition. Most notably, however, have we released the complete software stack including all components and configurations as a ready-to-run package.

The website of the Carologistics RoboCup Team with further information and media can be found at `http://www.carologistics.org`.

Acknowledgments. We gratefully acknowledge the financial support of RWTH Aachen University and FH Aachen University of Applied Sciences.
We thank Sick AG and Adolf Hast GmbH & Co. KG for sponsoring our efforts by providing hardware and manufacturing support.
F. Zwilling and T. Niemueller were supported by the German National Science Foundation (DFG) research unit *FOR 1513* on Hybrid Reasoning for Intelligent Systems (`http://www.hybrid-reasoning.org`).

References

1. Niemueller, T., Reuter, S., Ferrein, A.: Fawkes for the robocup logistics league. In: RoboCup Symposium 2015 – Development Track. (2015) to appear.
2. Niemueller, T., Ewert, D., Reuter, S., Ferrein, A., Jeschke, S., Lakemeyer, G.: RoboCup Logistics League Sponsored by Festo: A Competitive Factory Automation Benchmark. In: RoboCup Symposium 2013. (2013)
3. Niemueller, T., Lakemeyer, G., Ferrein, A., Reuter, S., Ewert, D., Jeschke, S., Pensky, D., Karras, U.: Proposal for Advancements to the LLSF in 2014 and beyond. In: ICAR – 1st Workshop on Developments in RoboCup Leagues. (2013)
4. Niemueller, T., Lakemeyer, G., Ferrein, A.: Incremental Task-level Reasoning in a Competitive Factory Automation Scenario. In: Proc. of AAAI Spring Symposium 2013 - Designing Intelligent Robots: Reintegrating AI. (2013)
5. Niemueller, T., Lakemeyer, G., Ferrein, A.: The robocup logistics league as a benchmark for planning in robotics. In: 25th International Conference on Automated Planning and Scheduling (ICAPS) – Workshop on Planning in Robotics. (2015) to appear.

6. Karras, U., Pensky, D., Rojas, O.: Mobile Robotics in Education and Research of Logistics. In: IROS 2011 – Workshop on Metrics and Methodologies for Autonomous Robot Teams in Logistics. (2011)
7. Niemueller, T., Ferrein, A., Beck, D., Lakemeyer, G.: Design Principles of the Component-Based Robot Software Framework Fawkes. In: Int. Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR). (2010)
8. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source Robot Operating System. In: ICRA Workshop on Open Source Software. (2009)
9. Gat, E.: Three-layer architectures. In Kortenkamp, D., Bonasso, R.P., Murphy, R., eds.: Artificial Intelligence and Mobile Robots. MIT Press (1998) 195–210
10. Mataré, V., Niemueller, T., Lakemeyer, G.: Robust multi-modal detection of industrial signal light towers. In: RoboCup Symposium. (2016)
11. Niemueller, T., Ferrein, A., Lakemeyer, G.: A Lua-based Behavior Engine for Controlling the Humanoid Robot Nao. In: RoboCup Symposium 2009. (2009)
12. Wygant, R.M.: CLIPS: A powerful development and delivery expert system tool. Computers & Industrial Engineering **17**(1–4) (1989)
13. Niemueller, T., Zwilling, F., Lakemeyer, G., Löbach, M., Reuter, S., Jeschke, S., Ferrein, A.: Cyber-Physical System Intelligence – Knowledge-Based Mobile Robot Autonomy in an Industrial Scenario. In: Industrial Internet of Things: Cybermanufacturing Systems. Springer (2016) (to appear).
14. Zwilling, F., Niemueller, T., Lakemeyer, G.: Simulation for the RoboCup Logistics League with Real-World Environment Agency and Multi-level Abstraction. In: RoboCup Symposium. (2014)
15. Niemueller, T., Reuter, S., Ewert, D., Ferrein, A., Jeschke, S., Lakemeyer, G.: Decisive Factors for the Success of the Carologistics RoboCup Team in the RoboCup Logistics League 2014. In: RoboCup Symposium – Champion Teams Track. (2014)
16. Beck, D., Niemueller, T.: AllemaniACs 2009 Team Description. Technical report, Knowledge-based Systems Group, RWTH Aachen University (2009)
17. Ferrein, A., Steinbauer, G., McPhillips, G., Niemueller, T., Potgieter, A.: Team ZaDeAt 2009 – Team Report. Technical report, RWTH Aachen Univ., Graz Univ. of Technology, and Univ. of Cape Town (2009)
18. Schiffer, S., Lakemeyer, G.: AllemaniACs Team Description RoboCup@Home. TDP, Knowledge-based Systems Group, RWTH Aachen University (2011)
19. Ferrein, A., Niemueller, T., Schiffer, S., and, G.L.: Lessons Learnt from Developing the Embodied AI Platform Caesar for Domestic Service Robotics. In: AAAI Spring Symposium 2013 - Designing Intelligent Robots: Reintegrating AI. (2013)
20. Zug, S., Niemueller, T., Hochgeschwender, N., Seidensticker, K., Seidel, M., Friedrich, T., Neumann, T., Karras, U., Kraetzschmar, G., Ferrein, A.: An Integration Challenge to Bridge the Gap among Industry-inspired RoboCup Leagues. In: RoboCup Symposium. (2016)
21. Niemueller, T., Karpas, E., Vaquero, T., Timmons, E.: Planning Competition for Logistics Robots in Simulation. In: WS on Planning and Robotics (PlanRob) at Int. Conf. on Automated Planning and Scheduling (ICAPS), London, UK (June 2016)