The Carologistics RoboCup Logistics Team 2018

Till Hofmann[1], Nicolas Limpert[2], Victor Mataré[2], Sebastian Schönitz[3], Tim Niemueller[1], Alexander Ferrein[2], and Gerhard Lakemeyer[1]

[1] Knowledge-Based Systems Group, RWTH Aachen University, Germany
[2] MASCOR Institute, FH Aachen University of Applied Sciences, Germany
[3] Cybernetics Lab IMA & IfU , RWTH Aachen University, Germany

Abstract. The Carologistics team participates in the RoboCup Logistics League for the seventh year. The RCLL requires precise vision, manipulation and path planning, as well as complex high-level decision making and multi-robot coordination. We outline our approach with an emphasis on recent modifications to those components.

The team members in 2018 are David Bosen, Christoph Gollok, Mostafa Gomaa, Daniel Habering, Till Hofmann, Nicolas Limpert, Sebastian Schönitz, Morian Sonnet, Carsten Stoffels, and Tarik Viehmann.

This paper is based on the last year's team description [1].

1 Introduction

The Carologistics RoboCup Team is a cooperation of the Knowledge-Based Systems Group, the Cybernetics Lab IMA & IfU (both RWTH Aachen University), and the MASCOR Institute (FH Aachen University of Applied Sciences). 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, 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 2012–2017 and the RoboCup German Open (GO) 2013–2018. We were able to win the GO 2014-2018 as well as the RoboCup 2014 thru 2017 demonstrating flexible task coordination, robust collision avoidance and self-localization through an easily maintainable and extensible framework architecture. We have publicly released our software stack in 2014, 2015 and 2016[4] [2].

In the following we will describe some of the challenges of the RCLL with a focus on the changes introduced in 2018. In Section 2 we give an overview of the Carologistics platform and describe how we adapted the platform in the past

[4] Software stack available at `https://www.fawkesrobotics.org/projects/rcll2016-release/`

year. We continue describing the changes to behavior components in Section 4 and our continued involvement for advancing the RCLL before concluding in Section 6.

1.1 RoboCup Logistics League 2018

As in previous years, 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. [4,5,6]). After the league switched from purely symbolic production to Festo's physical Modular Production System (MPS) in 2015 [5], the rules and field layout have been incrementally refined to focus on challenges that are relevant to the Industry 4.0 movement [7].

For 2018, only minor changes have been brought to the league to allow the teams to stabilize and incrementally improve their performance. After the Storage Station was added in 2017, we expect to use the storage station to fetch a pre-produced C_0 product, which allows the fulfillment of a product with less effort, but also with lower reward.

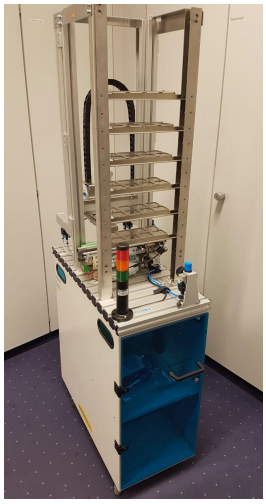

Fig. 1. The Storage Station [3]

2 The Carologistics Platform

The standard robot platform of this league is the Robotino by Festo Didactic [8]. The Robotino is developed for research and education and features omni-directional 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

Our current robot system is based on the Robotino 3. The modified Robotino used by the Carologistics RoboCup team is shown in Figure 2 and features an additional webcam to identify machine markers, a RealSense depth camera to recognize the conveyor belt, and two Sick laser range finders. We use a forward facing Sick TiM571 and a tilted backwards facing Sick TiM551 laser scanner for collision avoidance and self-localization. The TiM571 has a scanning range of 25 m (10 m for the TiM551) at a resolution of 1/3 degrees (1 degree for the TiM551).

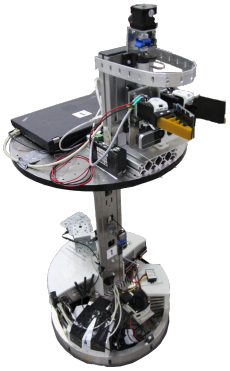

Fig. 2. Carologistics Robotino 2016 [9].

An additional laptop increases the computation power and allows for more elaborate methods for self-localization, computer vision, and reasoning.

Several parts were custom-made for our robot platform. As described in detail in the next section, we use a custom-made gripper and 3D-printed parts for product handling.

2.2 Mechanical Adaptations

This year, we are working on a new gripping system that is shown in Figure 3. In the past we had to move the whole robot to correct a lateral offset between the gripper and the conveyor belt. This approach has at least two downsides. Firstly, we can only consider motion relative to the last known pose calculated by the odometry, as a global localization is unreliable right in front of the MPS. Secondly, the motors of the Robotino only allow motions with a certain speed threshold. This prevents to perform small corrections of the gripper pose within millimeters, so we had to work with rather large tolerances. To overcome these limitations we decided to add two additional axes to the gripper that allow motion in 3 dimensions. The linear axes are driven by stepper motors which allow an accuracy in the range of a tenth of a millimeter. In addition to the 3 stepper motors for the linear axes another one is added to open and close the gripper fingers.

To keep costs as low as possible, the controller of the stepper motors consists of an Arduino Uno with an appropriate CNC milling electronics board. This Arduino shield handles up to 4 stepper motors. Particularly it has 2 pins for each stepper, one for the motion direction and another to trigger a step. The resulting force that the steppers have to apply is very low. However, to make sure that we do not lose steps while moving we apply an accelerated motion for each stepper motor.

We also replaced the Festo flex fingers by stiff, 3D-printed fingers. The advantage of the new fingers is mainly that they are not as wide, which reduced the risk of colliding with the machine output sensors, a common problem of the previous years.

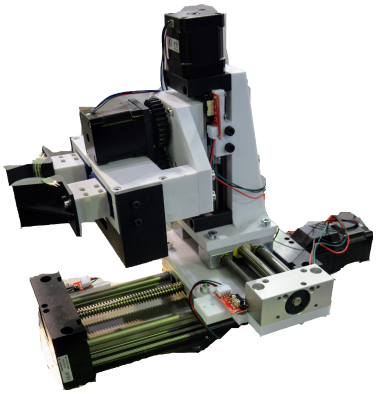

Fig. 3. New gripper with 3 linear axes and 3d printed fingers

2.3 Architecture and Middleware

The software system of the Carologistics robots combines two different middlewares, Fawkes [10] and ROS [11]. This allows us to use software components

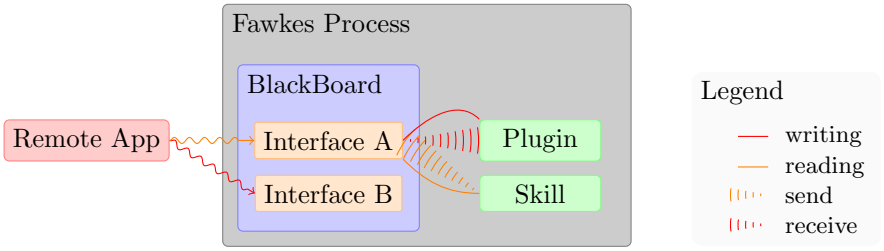

Fig. 4. Components communicate state data via interfaces stored in the blackboard. Commands and instructions are send as messages. Communication is universally shared among functional plugins and behavioral components [10].

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 [12]. 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 [10]. This allows for information exchange between arbitrary components. As shown in Figure 4, 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. In this section, we focus on changes for this year's competition, namely improvements on the MPS detection, a new path planning module and a revised conveyor belt detection.

3.1 MPS Detection and Approaching

We use a combination of tag detection and line fitting on the laser data to detect and approach an MPS. In a first step, the tag on the machine is used to validate whether the robot is approaching the correct machine and to roughly align the

robot to the machine. As the tag vision only gives an imprecise position and especially rotation of the detected tag, we perform a second alignment step by searching the laser data for a suitable line, which is then used for a more precise alignment to the machine.

Markerless Machine Detection As in 2016 and 2017, this year's technical challenges will include recognition of the machine type without the use of AR tags. To do so, we trained an artificial neural network (ANN) with several RGB-D pictures recorded for each machine during the Robocup 2017. During machine detection a picture is fed to the ANN, which then calculates a similarity measure for each possible machine type.

Since some machine types offer very few differentiating features, two thresholds are defined. The first specifies a minimum probability for the best match, while the second specifies a maximum probability for all other possible matches. To reach sufficient certainty for a successful report, both thresholds have to be satisfied for multiple perspectives on a single machine.

3.2 Path Planning

With the change to the ROS Navigation Stack (navstack) in 2017 we got a flexible navigation system for path planning. In general, the procedure of the navstack is as follows: After receiving a desired goal, the global planner tries to find a path with an A*-Search algorithm [13]. The graph for the least-cost search problem is represented as an occupancy grid (occ-grid). If the global planner successfully finds a path it is sent to a controller which has the task to calculate motion commands to bring the robot to the goal. As proposed last year we still make use of the TEB Local Planner. The calculated global path is optimal with respect to path length. However, the generated paths only consider static obstacles. The occ-grid representation in the navstack is done with the *costmap_2d* package [14]. Updates of the occ-grid represent the static map, obstacles gathered by the laser scanners and an inflation of the two to set up the configuration space. As the whole ROS *move_base* approach only plans with the current occ-grid information it cannot represent dynamic obstacles by default. To overcome this limitation we share parts of the global paths generated by the robots among each other. The shared paths from one robot are respected by the other robots with a simple prioritization paradigm; each robot's priority is identified by its number (e.g. 1 for Robotino 1) resulting in a lower priority with a rising number. To keep the number of software changes low, we decided to add this information to

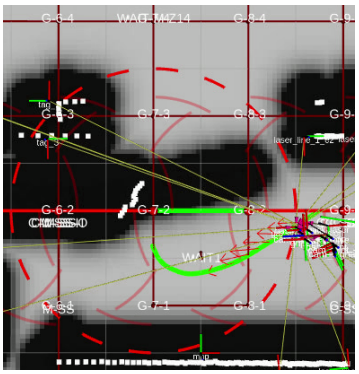

Fig. 5. Robotino on the bottom right adheres to path from other Robotino

the occ-grid. We do not need the fully detailed path information but only a few points representing a rough estimation of the path. Interpolating these allows us to draw path information as cells of the occ-grid. Figure 5 shows an example where the bottom right robot used path information from an approaching robot with a higher priority. The figure shows the protruding path information in front of the approaching robot.

3.3 Conveyor Belt Detection

The conveyor belts are rather narrow compared to the products and thus require precise handling. For reliable interaction, the error margin should be less than 3 mm. For 2018, we are changing the previous method [9] to use a more descriptive model for detecting the conveyor belt. We continue using an Intel RealSense F200 camera to detect the vertical front face of the conveyor belt. However, instead of fitting a plane, we now fit a 3D pointcloud model recorded from the conveyor belt itself. Much of the state of the art PCL-based method for 6DOF object recognition that is described in [15] is concerned with recognizing known shapes in an entirely unkown scene.

In our case we already have an initial guess that is computed simply from the MPS' position relative to the robot (cf. Section 3.1) and the approximate position of the conveyor belt on the MPS. This initial guess is sufficiently precise ($e \lesssim 1\,\text{cm}$) to run an Iterative Closest Point (ICP) algorithm[5] that reliably reduces the error to less than 2 mm.

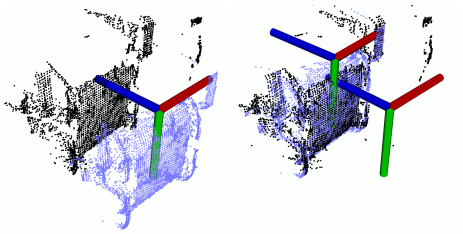

Fig. 6. Left: Model pointcloud (blue) roughly aligned to scene (black) based on initial guess. Right: After running ICP, the model is aligned to the scene precisely.

Among the termination criteria of ICP is the length of the last incremental transform. This criterion is watched, and if it is undercut, we reduce the correspondence threshold to allow ICP to ignore more model points that might not have a corresponding point in the scene due to self-occlusion (e.g. as it happens when viewing from a different angle than the model was recorded from). However this allows the algorithm to degenerate to a bad fit where only a small portion of the model points have close correspondences, while the majority is ignored. Consequentially, the last step of performing hypothesis verification is essential. For this purpose, we are currently evaluating the RANSAC-based algorithm proposed by [16].

[5] `http://pointclouds.org/documentation/tutorials/iterative_closest_point.php`

4 High-level Decision Making and Task Coordination

The behavior generating components are separated into three layers, as depicted in Figure 7: 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.

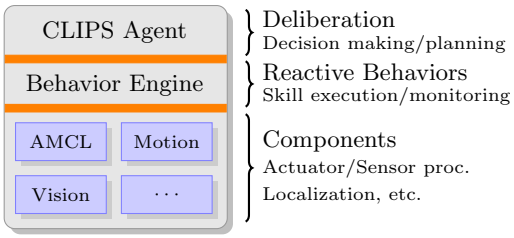

Fig. 7. Behavior Layer Separation [6]

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 [7]. 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.

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) [17]. 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 Robot Memory

In sufficiently complex robotics domains, we typically observe some degree of code duplication when common (e.g. geometric) operations are required at different levels of the framework. These operations typically revolve around an intermediate, factual knowledge level that consists of processed sensory information and agent execution state, often termed *world model*. This world model

is now synchronized between robots by the generic fact representation engine *Robot Memory* [18], which is based on MongoDB. The MongoDB back-end provides an easy way to gracefully handle robot failover and reduce network traffic through incremental updates. In addition to world model synchronization, it also provides the means for multi-robot coordination by serving as a back-end for resource locking.

4.3 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.

Based on the experience from previous years [6], we implemented an agent for the RCLL based on the CLIPS Executive (CX) [19,20] which uses an extended and adapted goal lifecycle (cf. [21,22]) to define the control flow. The CX provides an explicit representation of the agent's world model, and its goals, plans, and actions. It separates the *domain model* with the available operators, predicates, and known facts from the *execution model*, which enhances the domain model by features that are only relevant for the execution of the plan, e.g., *exogenous actions* and *sensed predicates*. In contrast to the approaches described in [19,20], we currently do not use a planner, but instead use pre-defined plans.

4.4 Multi-Robot Coordination

The CX also provides means for multi-robot coordination, in particular resource locking. In comparison to our previous approaches to task coordination, we now use resources that are more closely connected to real-world objects. Instead of locking a particular task, we lock the MPS and the work piece required for that task. A resource lock can be tied to a goal, in which case all locks are released after a goal has finished. As an example, before delivering a product, the respective order is locked by the goal, because the goal will fulfill the order. In a different case, a lock is acquired and released with explicit lock and unlock actions within a plan. As an example, before an agent interacts with an MPS, it acquires the lock for the MPS, and directly releases the lock after the interaction. This allows tight coordination of multiple agents, e.g., two robots may use the same MPS right after each other. In addition to resource locks, we also use location locks to avoid two robots being in the same location. Similar to the locks described above, a location lock is acquired with an explicit lock action. However, for releasing the lock, we use distance-based unlocking, i.e., the location lock is only released if the robot reached a certain distance to the location.

4.5 Multi-robot Simulation in Gazebo

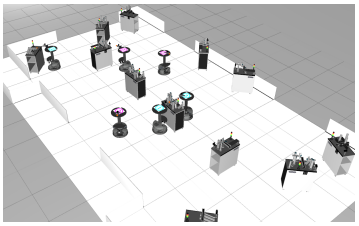

Fig. 8. Simulation of the RCLL 2015 with MPS stations [7].

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* [23,24] based on Gazebo[6].

This year, we extended the simulation with several error scenarios to test our agent's execution monitoring. The simulation now supports random failures during picking and putting, and it also lets the gripper drop the puck randomly. Additionally, it simulates systematic errors of an MPS, e.g., a cap station that always breaks during any interaction for a certain period of time. This way, we we were able to test the behavior of our agent in many scenarios that we have not encountered in a real-world game before.

5 League Advancements and Continued Involvement

We have been active members of the Technical and Executive Committees and proposed various changes for the league [4,5]. Additionally, we introduced and currently maintain the autonomous referee box for the competition and develop the open simulation environment described above.

5.1 Public Release of Full Software Stack

Over the past ten years, we have developed the *Fawkes Robot Software Framework* [10] 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 [25] and Standard Platform [26] soccer leagues, the RoboCup@Home [27,28] service robot league, and now in the *RoboCup Logistics League* [24].

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

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 2014 till 2016. This in particular *includes* the complete *task-level executive* component of 2014 and 2015, 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). We plan to do a similar software release after RoboCup 2018.

In addition to our software stack, we aim to release all our modifications to other software components. This includes a set of RCLL ROS packages[7], modifications of the ROS navigation stack[8], and a port of the tag tracking library Alvar to OpenCV3[9]. Furthermore, we package third-party libraries such as the Point Cloud Library (PCL) and the RealSense camera driver for the open-source operating system Fedora to allow easy installation of our software stack. We also maintain Fedora packages for the full ROS stack[10], including `desktop_full`, `move_base`, and `moveit`.

5.2 Planning Competition for Logistics Robots in Simulation

The first Planning and Execution Competition for Logistics Robots In Simulation[11] was held at the last year's International Conference on Automated Planning and Scheduling (ICAPS'17). In the simulation competition, the challenge is to efficiently plan in short time with dynamic orders and temporal constraints, and to provide an effective executive for multi-robot plans. The competition is based on the simulation developed by the Carologistics team, which has been extended to be able to run in a cluster or cloud-computing setup. The idea is to foster collaboration and exchange among the planning and robotics communities. Our team competed came in second place with an agent different to the agent used at RoboCup and based on OpenPRS [29]. The agent is described in more detail in [30].

6 Conclusion

In 2018, we developed a new agent based on the CLIPS Executive that provides an explicit goal representation including plans and actions with their preconditions and effects, and with a resource locking mechanism based on *Robot Memory*. We replaced our gripper with three linear axes and thus with three degrees of freedom that allows precise grasping without moving the Robotino base. We implemented a new conveyor vision component based on Iterative Closest Point

[7] `https://github.com/timn/ros-rcll_ros`
[8] `https://github.com/nlimpert/navigation`
[9] `https://github.com/morxa/alvar`
[10] `https://copr.fedorainfracloud.org/coprs/thofmann/ros/`
[11] `http://www.robocup-logistics.org/sim-comp`

(ICP) which uses a model of the conveyor belt to allow for a more precise detection. We augmented our path planning with velocity sharing to include the other robots' motions for planning to reduce travelling time while avoiding collisions with robots of our own team.

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

Acknowledgements. We gratefully acknowledge the financial support of RWTH Aachen University and FH Aachen University of Applied Sciences.
T. Hofmann and V. Mataré were supported by the DFG grants *GL-747/23-1* and *FE-1077/4-1* (respectively) on Constraint-based Transformations of Abstract Task Plans into Executable Actions for Autonomous Robots[12].
T. Niemueller was supported by the German National Science Foundation (DFG) research unit *FOR 1513* on Hybrid Reasoning for Intelligent Systems[13].

References

1. Neumann, T., Hofmann, T., Mataré, V., Henke, C., Schönitz, S., Niemueller, T., Ferrein, A., Jeschke, S., Lakemeyer, G.: The Carologistics RoboCup Logistics Team 2017. Technical report, RWTH Aachen University and FH Aachen University of Applied Sciences (2017)
2. Niemueller, T., Reuter, S., Ferrein, A.: Fawkes for the robocup logistics league. In: RoboCup Symposium 2015 – Development Track. (2015)
3. RCLL Technical Committee: RoboCup Logistics League – Rules and Regulations 2018. Technical report, RoboCup Logistics League (2018)
4. 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)
5. 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)
6. 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)
7. 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)
8. 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)
9. Niemueller, T., Neumann, T., Henke, C., Schönitz, S., Reuter, S., Ferrein, A., Jeschke, S., Lakemeyer, G.: Improvements for a Robust Production in the RoboCup Logistics League 2016. In: RoboCup Symposium – Champion Teams Track. (2016)
10. 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)

[12] `http://gepris.dfg.de/gepris/projekt/288705857`
[13] `http://www.hybrid-reasoning.org`

11. 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)
12. Gat, E.: Three-layer architectures. In Kortenkamp, D., Bonasso, R.P., Murphy, R., eds.: Artificial Intelligence and Mobile Robots. MIT Press (1998) 195–210
13. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics **4**(2) (1968) 100–107
14. Lu, D.V., Hershberger, D., Smart, W.D.: Layered costmaps for context-sensitive navigation. In: Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, IEEE (2014) 709–715
15. Aldoma, A., Marton, Z.C., Tombari, F., Wohlkinger, W., Potthast, C., Zeisl, B., Rusu, R.B., Gedikli, S., Vincze, M.: Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation. IEEE Robotics & Automation Magazine **19**(3) (2012) 80–91
16. Papazov, C., Burschka, D.: An efficient ransac for 3d object recognition in noisy and occluded scenes. In: Asian Conference on Computer Vision, Springer (2010) 135–148
17. Niemueller, T., Ferrein, A., Lakemeyer, G.: A Lua-based Behavior Engine for Controlling the Humanoid Robot Nao. In: RoboCup Symposium 2009. (2009)
18. Zwilling, F.: A document-oriented robot memory for knowledge sharing and hybrid reasoning on mobile robots. Master's thesis, RWTH Aachen University (2017)
19. Niemueller, T., Hofmann, T., Lakemeyer, G.: Clips-based execution for pddl planners. In: ICAPS Workshop on Integrated Planning, Acting and Execution (IntEx). (2018)
20. Niemueller, T., Lakemeyer, G., Leofante, F., Abraham, E.: Towards clips-based task execution and monitoring with smt-based decision optimization. In: Workshop on Planning and Robotics (PlanRob) at International Conference on Automated Planning and Scheduling (ICAPS), Pittsburgh, PA, USA (June 2017)
21. Harland, J., Morley, D.N., Thangarajah, J., Yorke-Smith, N.: An operational semantics for the goal life-cycle in BDI agents. Autonomous Agents and Multi-Agent Systems **28**(4) (7 2014)
22. Roberts, M., Vattam, S., Alford, R., Auslander, B., Karneeb, J., Molineaux, M., Apker, T., Wilson, M., McMahon, J., Aha, D.: Iterative goal refinement for robotics. Working Notes of the Planning and Robotics Workshop at ICAPS (2014)
23. 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)
24. 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)
25. Beck, D., Niemueller, T.: AllemaniACs 2009 Team Description. Technical report, Knowledge-based Systems Group, RWTH Aachen University (2009)
26. 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)
27. Schiffer, S., Lakemeyer, G.: AllemaniACs Team Description RoboCup@Home. TDP, Knowledge-based Systems Group, RWTH Aachen University (2011)
28. 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)

29. Alami, R., Chatila, R., Fleury, S., Ghallab, M., Ingrand, F.: An architecture for autonomy. The International Journal of Robotics Research **17**(4) (1998)
30. 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)