Enhancing Software and Hardware Reliability for a Successful Participation in the RoboCup Logistics League 2017

Till Hofmann[1], Victor Mataré[2], Tobias Neumann[2], Sebastian Schönitz[3], Christoph Henke[3], Nicolas Limpert[2], Tim Niemueller[1], 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] Cybernetics Lab IMA/ZLW & IfU, RWTH Aachen University, Germany

Abstract. In 2017, the RoboCup Logistics League (RCLL) has seen major changes to the playing field layout, which allows for more configuration variants that are now generated randomly and automatically, leading towards a more realistic smart factory scenario. The Carologistics team developed a new strategy for exploration and improved existing components with a particular focus on navigation and error handling in the behavior engine and high-level reasoning. We describe the major concepts of our approach with a focus on the improvements in comparison to last year, which enabled us to win the competition in 2017.

1 Introduction

The RoboCup Logistics League focuses on multi-robot coordination in an industrial smart factory environment. The domain presents interesting challenges for research in automated reasoning, planning, and scheduling.

The Carologistics team has participated in RoboCup 2012–2017 and in the RoboCup German Open 2013–2017, winning first place in both competitions from 2014 to 2017. Overall, our success is due to a stable middleware foundation, incremental refinement of components, and continuous recruitment of new talent.

In 2017, the league has seen a shift in focus towards the production phase. The exploration phase has been simplified, most notably by removing the requirement for teams to identify machines by means of a signal light code. The production phase in turn was extended in time and a new type of modular production system (MPS) was introduced. The size of the playing field was increased from $12m \times 6m$ to $14m \times 8m$, while the zone size was reduced. This increased the number of possible zones for MPS placement from 24 to 106. Unambiguous MPS positioning rules were formulated, so that the playing field layout can now be automatically generated by the Refbox. Point rewards have been re-balanced to reflect the increased emphasis on production instead of exploration.

This paper is based on last year's edition [1], highlighting in particular how the Carologistics team adapted to the new challenges posed by the updated game logic. For a general description of the RCLL we refer to [2,3,4,5].

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. Our software is based on the Fawkes robotics framework [7], which integrates modules for all essential functional components.

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 1. It features an additional webcam to identify machine markers, a RealSense depth camera to recognize the conveyor belt, and two Sick laser range finders.

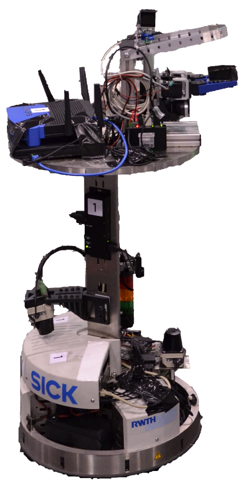
Fig. 1. Carologistics Robotino 2017.

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). 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, a custom-made gripper based on Festo fin-ray fingers and 3D-printed parts are 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 in order to increase 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 Mechanical Adaptations

To increase the agility of the robot, an additional back-facing laser scanner was added. As shown in Figure 2 on the left, the laser is mounted just above

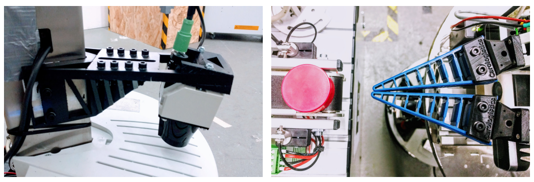
Fig. 2. Back-facing laser and new gripper

the Robotino's computational unit under a slope of 12°, enabling it to perceive other robots' main body and not only the center pillar.

Another adaptation was made to the gripper system. Some of the grippers were damaged in past games and had to be replaced by the new version of Festo flex fingers, which required a newly designed mounting as shown in Figure 2 on the right. The design consists of two easily separable parts. The first part is attached to the gripper while the second part is attached to the Dynamixel motor. Both parts are mounted together via a dovetail guide to enable quick replacement of a damaged gripper during a match.

2.3 Software Frameworks

The Fawkes robotics framework [7] serves as a central integration and coordination system. It implements a hybrid blackboard and messaging mechanism for inter-component communication, a vast library of primitives for I/O and data processing, and a flexible API for thead time-slicing and synchronization. Furthermore, it provides an adapter that allows us to integrate ROS nodes [8]. This has proven to be a great asset over time, given the continuous and rapid advancement of robotic technologies. It allows us to flexibly evaluate and integrate new technologies from the ROS ecosystem (cf. Section 3.1).

The overall software structure is inspired by the three-layer architecture [10] as shown in Figure 3. 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.

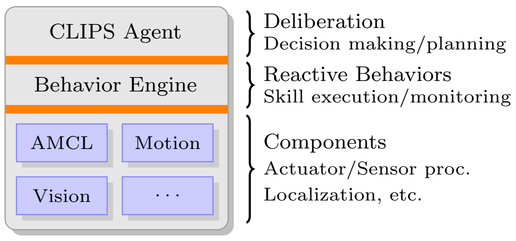

Fig. 3. Behavior Layer Separation [9]

3 Advances to Functional Software Components

3.1 Driving

To increase navigation flexibility we decided to change our navigation approach to the ROS Navigation Stack (navstack). It consists of several packages for local and global path planning, obstacle representations in an occupancy grid (occ-grid) and recovery behaviours to unstuck the robot. Apart from the very active maintaining community around the navstack, other benefits are the numerous options to further increase navigation performance with other ROS packages. One example for this is a ROS node we use to check the velocity commands computed by the move_base for validity. As we cannot guarantee that the local planner is real-time capable, we have to verify collision-free locomotion.

One of the key advantages of ROS is the ability to adapt to environmental challenges. If the robot has to take care of velocity constraints in certain areas, one could design a ROS node that provides a velocity veto functionality to overwrite motion commands or to set parameters for the navigation during runtime.

On top of the navstack we still make use of a global planner called NavGraph, which utilizes the MPS positions detected in exploration phase. This is directly implemented in Fawkes and simply sends new goal poses to the navstack throughout its own plan. The benefit here is that we can represent domain features within the NavGraph's environmental representation (e.g. MPS positions and/or highways which allow faster motion).

In particular we use the costmap_2d package as a representation of an occ-grid. This graph representation is used by the global and local planners managed by the move_base node,[4] which loads a global and a local planner as plugins. move_base is responsible for receiving a desired goal and managing planning and motion execution as well as failure handling.

The global planning is performed by a graph-based planning approach represented in the ROS package called global_planner.[5] This planner implements the A* search algorithm [11] to efficiently find paths within an occ-grid from start to goal. However, in practice we get a lot of navigation goals that are inside obstacles. By design, the global planner fails in this situation and thus we have to find a cell that is close to the desired goal. Finding this free cell is performed by an adoption of the potential field method [12]. This allows us to robustly find paths leading to a position close to an infeasible goal.

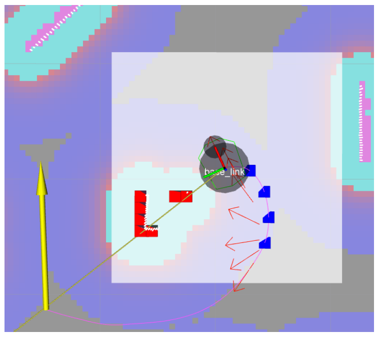

Fig. 4. move_base screenshot

Paths generated by the global planner are forwarded by the move_base to the local planner. For this we equip the teb_local_planner [13] (teb stands for "Timed Elastic Band") to achieve trajectories that are optimized w.r.t. execution time, obstacle distance and kinematic constraints. Kinematic constraints tell the local planner that we are moving with omnidirectional locomotion, thus we can save time in rotating to the final orientation of the final goal waypoint during motion.

In unfortunate situations the navigation gets stuck (either the global planner does not find any valid path or the local planner can't find legal motion commands which do not lead to collisions). On a conceptual level, the move_base executes recovery behaviours in these situations. Our former navigation makes use of an escape behaviour (particularly another potential field) which we simply ported to be used as a recovery behaviour in the move_base.

[4] move_base wiki page at `http://wiki.ros.org/move_base`

[5] `http://wiki.ros.org/global_planner`

3.2 Reactive Behavior

In previous work we have described the Lua-based Behavior Engine (BE) [14]. It serves as the reactive layer to interface between the low- and high-level systems (see Figure 3). The BE provides a library of so-called *skills* that aggregate inter-component execution monitoring and reactive strategies into domain-related actions. Each skill describes a hybrid state machine (HSM), which can be conceived 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.

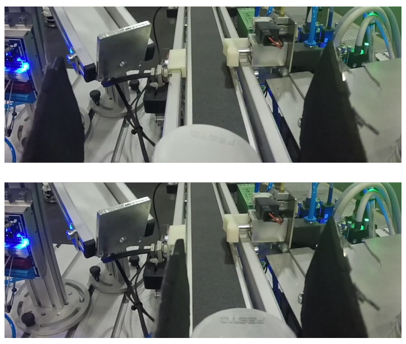
Fig. 5. Workpiece sits tilted on the rim (top), and is pushed down by moving the left finger individually (bottom).

One of the issues that continues to require substantial effort on the behavior layer is the MPS interaction. Since the current gripper design does not allow for sufficiently precise placement of workpieces on the conveyor belt, there is always a significant chance that workpieces may end up misaligned and tilted, with one edge sitting on the metal rim of the conveyor (Figure 5 top). In this case, MPS processing (i.e. cap/ring mounting/removal) would likely fail and the processing step would be wasted. Fortunately, the MPS interaction has been changed in 2017 so that the conveyor belt will only start moving *after* a prepare-instruction has been sent to the MPS. Our gripper design with an independent servo for each finger thus allows us to give the workpiece a nudge on each side to make sure it slides off the rim and rests completely on the conveyor before the MPS operation is started (see Figure 5 bottom).

4 High-level Decision Making and Task Coordination

4.1 Exploration

Due to the much simplified exploration phase, the old exploration agent had to be rewritten. Since the number of zones has increased from 24 to 106, the deliberate zone-by-zone strategy used in previous years is not feasible any more. The responsible CLIPS agent code could be reduced from over 900 LoC to 580 LoC. The idea is to let all three robots roam the playing field on predefined routes while a RANSAC-based line detection algorithm scans for patterns that look like the straight wall panel of an MPS. From the pose of the detected line we can compute a reasonable hypothesis of where to look for the tag and position the robot accordingly. Once a tag has been found, its pose is discretized to multiples of 45° and reported to the refbox. Since the playing field is always symmetric, we can compute the pose of the opposite MPS according to the logic specified in the rulebook, which allows us to use machines of both teams for exploration. Progress is immediately shared between all robots to avoid duplicate effort and to allow each robot to compute a navigational graph on its own.

4.2 Reasoning and Planning

As in previous competitions, we use a distributed, local-scope, and incremental reasoning approach [9] implemented in the rule-based production system CLIPS [15]. As shown in Figure 3, the CLIPS agent constitutes the decision making layer and builds on top of the behavior engine (cf., Section 3.2). We pursue the same general strategy as in previous years: Each agent acts on its own and synchronizes its world model with the other agents. A master agent ensures that the world models of all three agents are consistent. The master is determined dynamically through leader election. If the other agents cannot communicate with the master, a re-election is triggered. This ensures a consistent world model while dealing with robot and network failures in a robust way. Each agent follows an incremental strategy, i.e., in each step, it decides which step to take next based on its current world model. To avoid conflicts with the other robots, the currently selected next step is communicated to the other agents and a different step is selected if a conflict with another agent's strategy arises.

This year, we focused on improving the stability of the agent. In particular, we decided to focus on producing `C0`s (i.e., the product of lowest complexity) and implemented several recovery strategies, which allowed us to continue producing orders even in the event of a machine handling error. While the implemented monitoring strategies already improved error handling significantly in certain situations, we realized that a more principled approach is required to accomplish robust execution. Implementing such a principled execution monitoring strategy will be our goal for the next competitions.

5 Multi-robot Simulation in Gazebo

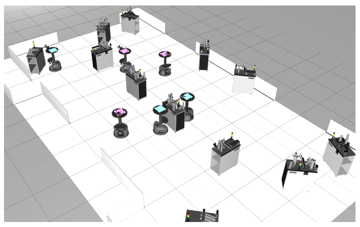
Fig. 6. Simulation of the RCLL 2015 with MPS stations [16].

We have continued working on the *open simulation environment* [17] based on Gazebo (see Figure 6). The simulation environment supports a full 3D physical simulation of an RCLL game including Referee Box (refbox), MPS placement and handling, and multi-robot communication. We see multiple advantages of such a simulation environment: For one, it lowers the burden for new teams, both in financial efforts and development requirements. A game setup including a full playing field, a set of MPSs, and three fully equipped robots has a high initial cost. Instead of buying a complete field, new teams can start developing in simulation and even compete in the simulation competition without buying any robot hardware, which may also increase the interest by non-robotic research communities (cf., next section). Additionally, the simulation environment allows for more rapid agent development. In a typical development cycle, all components of the lower and middle layer (see Figure 3) must be working before the agent development can start, as the agent usually relies on all the lower-level components. With the simulation, we can replace challenging tasks such as picking and putting products with simulated actions that always succeed. This way, agent development can start much sooner. Third, a simulation environment allows for rapid test cycles. This supports more elaborate integration and regression tests, which is crucial for such complex systems. Using the simulation in a fully automated fashion even offers the possibility of fully automatic tests.

This year, we improved the performance of the simulation by simplifying the models used in Gazebo. We added support for the new storage stations, and enabled random MPS placements by the refbox.

5.1 Planning and Execution Competition at ICAPS

The Planning and Execution Competition for Logistics Robots in Simulation[6] (PExC) [2] was held for the first time at the 27th International Conference on Automated Planning and Scheduling (ICAPS) 2017. The goal of the competition is to improve the cooperation between the planning and the robotics communities. The motivation stems from the observation that even though robotics domains often serve as motivating example for planning research, planning techniques are rarely used in robotic applications in general, and in the RoboCup Logistics League in particular. Both communities may benefit from a closer cooperation. For the robotics community, planning may improve the performance significantly by enabling both, more product deliveries and the production of more complex products. For the planning community, the RCLL serves as an interesting application scenario with a medium complexity and additional challenges in terms of closely integrated planning and execution.

PExC is based on the RCLL rules of 2016. The competition focuses on production and the exploration phase is skipped. All games are played in simulation (cf., Section 5), which lowers the barrier for new participants, especially from the planning community, while providing a realistic test scenario with simulated physics [17]. All simulations ran in a local small-scale cluster based on Kubernetes, which allows to play a large number of games in an automated fashion.

While contributing to the design, organization, and implementation of the competition, the Carologistics team also participated with two different approaches. The first approach uses the Procedural Reasoning System (PRS) [18]

[6] More information is available at `http://www.robocup-logistics.org/sim-comp`

with a centralized global strategy that uses a specific master agent that assigns tasks to all robots [19]. The second approach uses a (non-temporal) PDDL-based planner with macro actions [20] and postpones scheduling to be solved at run-time during execution.

6 League Advancements and Continued Involvement

Carologistics members are and have been active members of the Technical, Organizational, and Executive Committees and were involved in various changes that shaped the league, such as merging the two playing fields and using physical processing machines [5,4]. Furthermore, we are major contributors to the autonomous refbox [21] and to the open simulation environment described in Section 5. We have also been a driving factor in the establishment of the RoboCup Industrial umbrella league [4] and the cross-over challenge between the RCLL and RoboCup@Work league [22].

6.1 RCLL Referee Box and MPS Stations

The refbox was introduced in 2013 by the Carologistics team [5] as part of their work in the Technical Committee and has been maintained by the team since then. The goal of the refbox is to ease the work for referees by keeping track of the production state and by awarding points.

In 2017, significant changes to the refbox have been introduced. For one, the refbox has been adapted to the new playing field. The new rules of the field layout have been encoded into the refbox and an external field configuration generator developed by team GRIPS[7] has been integrated. This allows a random, automatic generation of playing fields during the game setup and eases the referees' tasks while increasing the automation in the league, and therefore increases the league's similarity to a smart factory.

Furthermore, significant work has been put into the communication between refbox and MPS. In past competitions, communication between the refbox and MPS was not very robust, which led to a number of game restarts due to a malfunctioning playing field. For this reason, a new communication method has been introduced and a first version has been implemented for the refbox. The goal is to switch to the new communication method for the 2018 competitions. With the new implementation, network failures will only lead to a delay in communication but will not affect the state of the MPSs nor the game.

6.2 Public Release of Full Software Stack

For eleven years we have developed the *Fawkes Robot Software Framework* [7]. Its development has been mainly driven for various RoboCup leagues [23]. It has been used in RoboCup@Home, MSL, SPL and has now been evolved for the RCLL, also shown in Figure 7.

[7] `http://www.robocup.tugraz.at`

While the domain-independent Fawkes framework is being developed in public, Carologistics was the first team in the RCLL to also release its full RCLL-specific stack.[8]

6.3 RoboCup Technical Challenge

In the RCLL, each MPS has an Augmented Reality (AR) tag, which encodes a unique identifier in a 5×5 grid of black and white squares, and which can be used to identify the machine. However, with the goal to adapt to common industrial logistics scenarios, the Markerless Machine Recognition and Production challenge requires a machine recognition without using tags.

We solved this challenge by training a neural network which has a videostream as input and the likelihood of the machine types as output. We did this in a three step approach: Taking data, labeling data, and training the neural network.

Most of our data was taken at the RoboCup German Open 2017. Using the Behavior Engine (cf., Section 3.2), we drove around each MPS and took pictures from different positions. We paid special attention to the diversity of the images: different light conditions, multiple backgrounds, and varying blurriness. We took a training set of 5000 images for each machine type. To label the images, we developed a program that allows for an interactive and convenient labeling.

Afterwards, we trained Google's pre-trained Inception-v3 model [24] and can usually recognize an MPS with one image. If a single image is not sufficient, we drive around the MPS until we are confident enough to report the machine type.

7 Conclusion

Following the major changes of 2015 and 2016, the year 2017 has been a year of stabilization both for the RCLL and for the Carologistics team. The team itself saw major shifts in personnel, with many long-time members phasing out and substantial recruitment of new talent. On a technical level, it became apparent

[8] 2016 release: `https://www.fawkesrobotics.org/projects/rcll2016-release/`, release for 2017 is in preparation

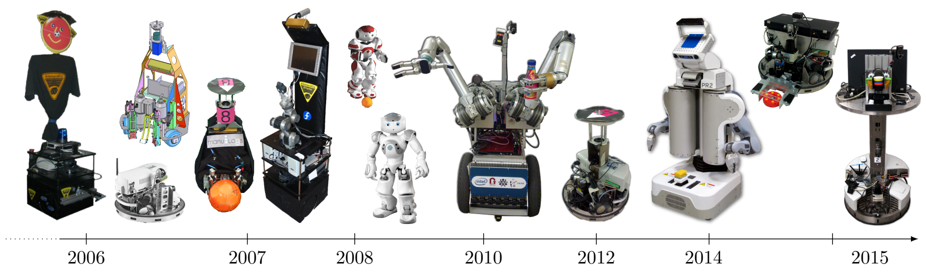

Fig. 7. Robots running (parts of) Fawkes which were or are used for the development of the framework and its components [23].

that many core components have reached a state of relative maturity, while others have evolved into a degree of complexity that warrants a principled re-engineering.

One particular example is the CLIPS agent, which encodes a wealth of smart heuristics and recovery strategies. However, the codebase goes back to the 2012 version of the RCLL, and continuous ad-hoc fixes have grown the state model to a point where the control flow is difficult to understand and a principled error handling approach is no longer possible. Thus, one of the major challenges of next year's competition will be to design a more structured agent framework with a more sophisticated execution monitoring strategy.

Acknowledgments. The team members in 2017 are Christoph Gollok, Mostafa Gomaa, Daniel Habering, Christoph Henke, Till Hofmann, Daniel Künster, Nicolas Limpert, Victor Mataré, Tobias Neumann, Tim Niemueller, Sebastian Schönitz and Carsten Stoffels.

We gratefully acknowledge the financial support of *RWTH Aachen University* and *FH Aachen University of Applied Sciences.*

We thank the *Hans Hermann Voss-Stiftung*[9] for their financial support, as well as the *Festo Didactic SE*[10] and *SICK Sensor Intelligence*[11] for sponsoring our efforts by providing hardware and manufacturing support.

V. Mataré and T. Hofmann were supported by the DFG grants *FE-1077/4-1* and *GL-747/23-1* (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. 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: Robot World Cup, Springer (2016) 589–600
2. 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 (2016)
3. Deppe, C., Karras, U., Neumann, T., Niemueller, T., Rohr, A., Steinbauer, G., Ewert, D., Harder, N., Jentzsch, S., Meier, N., Reuter, S., Uemura, W.: RoboCup Logistics League - Rules and Regulations 2017 (2017) Available at `http://www.robocup-logistics.org/rules`.
4. 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)

[9] `http://www.hans-hermann-voss-stiftung.de/`
[10] `https://www.festo-didactic.com`
[11] `https://www.sick.com`
[12] `http://gepris.dfg.de/gepris/projekt/288705857?language=en`
[13] `https://www.hybrid-reasoning.org`

5. 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)
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. 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)
10. Gat, E.: Three-layer architectures. In Kortenkamp, D., Bonasso, R.P., Murphy, R., eds.: Artificial Intelligence and Mobile Robots. MIT Press (1998) 195–210
11. Russell, S., Norvig, P., Intelligence, A.: A modern approach. Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs **25** (1995) 27
12. Barraquand, J., Langlois, B., Latombe, J.C.: Numerical potential field techniques for robot path planning. IEEE Transactions on Systems, Man, and Cybernetics **22**(2) (1992) 224–241
13. Rösmann, C., Feiten, W., Wösch, T., Hoffmann, F., Bertram, T.: Trajectory modification considering dynamic constraints of autonomous robots. In: Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on, VDE (2012) 1–6
14. Niemueller, T., Ferrein, A., Lakemeyer, G.: A Lua-based Behavior Engine for Controlling the Humanoid Robot Nao. In: RoboCup Symposium 2009. (2009)
15. Wygant, R.M.: CLIPS: A powerful development and delivery expert system tool. Computers & Industrial Engineering **17**(1–4) (1989)
16. 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) – WS on Planning in Robotics. (2015)
17. 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)
18. Alami, R., Chatila, R., Fleury, S., Ghallab, M., Ingrand, F.: An architecture for autonomy. The International Journal of Robotics Research **17**(4) (1998)
19. 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)
20. Hofmann, T., Niemueller, T., Lakemeyer, G.: Initial results on generating macro actions from a plan database for planning on autonomous mobile robots. In: Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS), Pittsburgh, PA, USA (2017)
21. Niemueller, T., Zug, S., Schneider, S., Karras, U.: Knowledge-Based Instrumentation and Control for Competitive Industry-Inspired Robotic Domains. KI - Künstliche Intelligenz **30**(289–299) (2016)
22. Zug, S., Niemueller, T., Hochgeschwender, N., Seidensticker, K., Seidel, M., Friedrich, T., Neumann, T., Karras, U., Kraetzschmar, G., Ferrein, A.: An Inte-

gration Challenge to Bridge the Gap among Industry-inspired RoboCup Leagues. In: RoboCup Symposium. (2016)
23. Niemueller, T., Reuter, S., Ferrein, A.: Fawkes for the RoboCup Logistics League. In: RoboCup Symposium 2015 – Development Track. (2015)
24. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the Inception Architecture for Computer Vision. ArXiv e-prints (December 2015)