Winning the RoboCup Logistics League with Visual Servoing and Centralized Goal Reasoning

Tarik Viehmann[1], Nicolas Limpert[2], Till Hofmann[1], Mike Henning[2], 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

Abstract. The RoboCup Logistics League (RCLL) is a robotics competition in a production logistics scenario in the context of a Smart Factory. In the competition, a team of three robots needs to assemble products to fulfill various orders that are requested online during the game. This year, the Carologistics team was able to win the competition with a new approach to multi-agent coordination as well as significant changes to the robot's perception unit and a pragmatic network setup using the cellular network instead of WiFi. In this paper, we describe the major components of our approach with a focus on the changes compared to the last physical competition in 2019.

1 Introduction

The Carologistics RoboCup Team is a cooperation of the Knowledge-Based Systems Group (RWTH Aachen University) and the MASCOR Institute (FH Aachen University of Applied Sciences). The team was initiated in 2012 and consists of Doctoral, master's, and bachelor's students of both partners who 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.

In previous competitions [7,6], we have pursued a distributed approach to multi-agent reasoning, where each robot acts on its own and coordinates with the other robots to resolve conflicts. This year, we have pursued a different strategy: Instead of having multiple agents each acting on its own, we now use one central goal reasoner that assigns tasks to each robot. This allows a more long-term strategy and avoids coordination overhead. Additionally, we have changed our approach to perception and manipulation. Instead of a pointcloud-matching approach that uses RGB/D data to iteratively determine an object's pose, we use a neural network to determine the bounding box of an object in an RGB image and then use closed-loop visual servoing to approach the object. Finally, we have taken first steps towards switching our navigation to ROS 2 and multi-agent path finding.

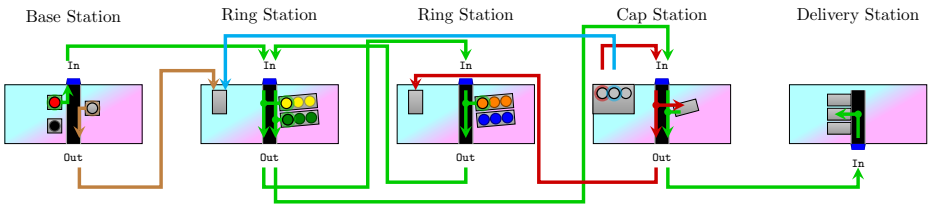


Fig. 1: Path for a product (green) of highest complexity, along with possible options to supply material for the ring assemblies (blue, brown and red). product in the RCLL.

In the following, we summarize the RCLL in Section 2 and provide an overview of our system, starting with our robot platform in Section 3. In Section 4, we present our software architecture and continue by describing our advances towards multi-agent path planning in Section 5, before we explain our approach to perception and visual servoing in Section 6. In Section 7, we summarize our approach to high-level decision making and describe our new centralized approach to multi-agent coordination, before we conclude in Section 8.

2 The RoboCup Logistics League

The RoboCup Logistics League (RCLL) [12] is a RoboCup [10] competition with a focus on smart factories and production logistics. In the RCLL, a team of mobile robots has to fulfill dynamically generated orders by assembling workpieces. The robots operate and transport workpieces between static production machines to assemble the requested products or to supply the stations with material necessary to perform some assembly steps. The major challenges of the RCLL include typical robotics tasks such as localization, navigation, perception, and manipulation, with a particular focus on reasoning tasks such as planning, plan execution, and execution monitoring.

The game is controlled by a semi-automatic Referee Box (refbox) [18]. The refbox generates dynamic orders that consist of the desired product configuration and a requested delivery time window for the product, which must be manufactured by the robots of each team. Each requested product consists of a base piece (colored red, black, or silver), up to three rings (colored blue, green, orange, or yellow), and a cap (colored black or gray), resulting in 246 possible product configurations. The complexity of a product is determined by the number of required rings, where a C0 product with zero rings is a product of the lowest complexity, and a C3 product with three rings is a product of the highest complexity. Each team has an exclusive set of seven machines of five different types of Modular Production System (MPS) stations. To manufacture a requested product, the team has to execute a sequence of production steps by means of operating the MPS stations. An exemplary production is shown in Figure 1.

3 The Carologistics Platform

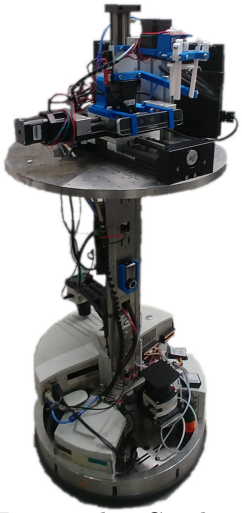


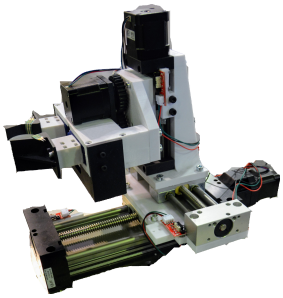
Fig. 2: The Carologistics Robotino

The standard robot platform of this league is the Robotino by Festo Didactic [9]. 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. The Carologistics Robotino is shown in Figure 2.

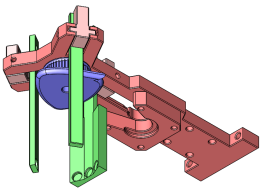
Sensors We use one forward-facing and one tilted, backward-facing SICK TiM571 laser scanner for collision avoidance and self-localization. Using a second laser scanner in the back allows us to fully utilize the omni-directional locomotion of the Robotino. In addition to the laser scanners, we use a webcam for detecting the MPS identification tags, and a Creative BlasterX Senz3D camera for conveyor belt detection.

3.1 Gripper System

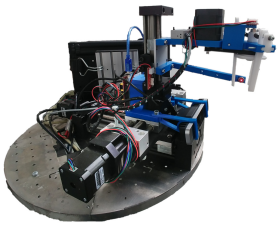
Our gripper system consists of three linear axes and a three-fingered gripper, as shown in Figure 3. The three axes are driven by stepper motors, which allows movements with sub-millimeter accuracy. The axes are controlled by an Arduino, which in turn receives commands from the Robotino main computer.



(a) The three linear axes driven by stepper motors



(b) The CAD model of the three-fingered gripper



(c) The complete gripper system

Fig. 3: The gripper system consisting of three linear axes and a self-centering gripper with three fingers

The gripper uses three fingers and grips the workpiece from above. This allows increased robustness and precision, as the workpiece is always centered between the three spring-loaded fingers, independent of positioning errors.

Since 2021, the laptop on top of the Robotino (cf. Figure 2) was removed, as the Robotino 4 is capabale of running our full software stack without further need for additional computational power. As the laptop also served as an access point, initially, a small access point was mounted to ensure WiFi connectivity.

3.2 Cellular Network Setup via Tailscale

The challenging characteristics of tens of competing wireless networks communicating across the different leagues are an ever existing issue at RoboCup. The change of our hardware components in terms of the network equipment attached to the Robotinos rendered our communication platform virtually unusable due to tremendous paket loss among systems trying to communicate across the playing field. In addition, the change to a central goal reasoning approach increased the dependency on reliable communication among the participating machines.

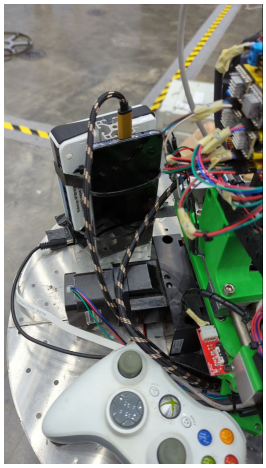


Fig. 4: Smartphone for USB based LTE tethering to the robot

To address these issues and allow us to compete properly, we switched from a local WiFi connection to the cellular network of a generic local provider using a Long Term Evolution (LTE) network. However, according to the current rules of the RCLL[3] "Communication among robots and to off-board computing units is allowed only using WiFi". This rule was mainly intended to prohibit wired connections, so we approached the other teams and the TC to get approval for the usage of the cellular network during the competition.

Each robot has a direct connection to the internet by using a smartphone, which tethers its LTE connection to the robot without using WiFi. As the robots expect a local network connection to each other, we equipped the VPN service Tailscale[4], which issues a static IP address to each robot and which is based on the WireGuard [4] network tunnel.

Albeit having some delay (100–200 ms), the UDP based connection was stable enough to reliably operate the robots and communicate to and from the central goal reasoning.

The authentication to join the Tailscale network is based on an existing identity provider (in our case we utilized our GitHub organization). In addition, the WireGuard tunnel encrypts the communication between the peers.

A drawback of this solution is the dependency on the cellular network infrastructure on site, which at the venue of RoboCup Bangkok was no issue. Additionally, we had to be mindful of the data usage as the we had limited data

[3] See Section 7 of `https://github.com/robocup-logistics/rcll-rulebook/releases/download/2022/rulebook2022.pdf`

[4] `https://tailscale.com/`

available on the chosen prepaid plans. By only sparsely using network-based visualization tools (such as pointcloud or camera output streams), we had more than 50% of our 20 GB limit available by the end of the tournament, hence it turned out to be a feasible solution.

4 Architecture and Middleware

The software system of the Carologistics robots combines two different middlewares, Fawkes [13] and ROS [19]. 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 [5], as shown in Figure 5. 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 communication between single components – implemented as *plugins* – is realized by a hybrid blackboard and messaging approach [13].

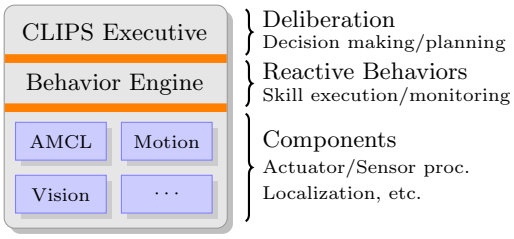


Fig. 5: Behavior Layer Separation [17]

Recent work within Fawkes includes the support to couple the reasoning component CLIPS Executive (CX) with multiple reactive behaviour engines (cf. Section 7.1) of remote Fawkes instances. This enabled us to use Fawkes to build a centralized reasoner controlling the robots to fulfill the tasks of the RCLL (see Section 7). Now each Robotino runs a Fawkes instance without a reasoning unit along with a ROS-based navigation stack (cf. Section 5). Additionally, a central computer runs a Fawkes instance with the CX (see Section 7.2) that deliberates about the production strategy and sends commands to the behavior engines running on the robots.

Also, while the current setup offers bridging capabilities between Fawkes and ROS, as ROS 2 [22] becomes more prominent, we also implemented interfaces between Fawkes and ROS 2 to prepare for a future switch to ROS 2. Since the Carologistics are using Fedora as operating system on the Robotino platforms, which is not officially supported by ROS 2, we work on providing appropriate packages as we already do for ROS 1[5]. Moreover, as an entry point of ROS 2 into the RCLL, we are currently porting the Robotino hardware driver from Fawkes to ROS 2. The Fawkes driver directly uses the hardware interfaces instead of the Robotino REST API, which lacks reliable time stamps.

[5] https://copr.fedorainfracloud.org/coprs/thofmann/ros/

5 Towards Path Planning in ROS 2

Our current setup utilizes our navigation stack as described in [6]. As a first use case for ROS 2 we actively work towards a multi-agent path finding (MAPF) solution with the help of the ROS 2 Navigation framework [11]. With the MAPF approach, it is possible to handle narrow situations or intersection scenarios, which are well known problems for our current single-agent navigation solution.

However, as the work on the ROS 2 solution is still in active development and not yet ready for usage in competitions, we chose to deploy the ROS based navigation from previous years. Notably, the network middlware DDS[6] deployed in ROS 2 is quite complex and we could not configure it robustly, which sometimes caused faulty pose state estimations leading to unpredictable navigation behaviour.

6 Perception

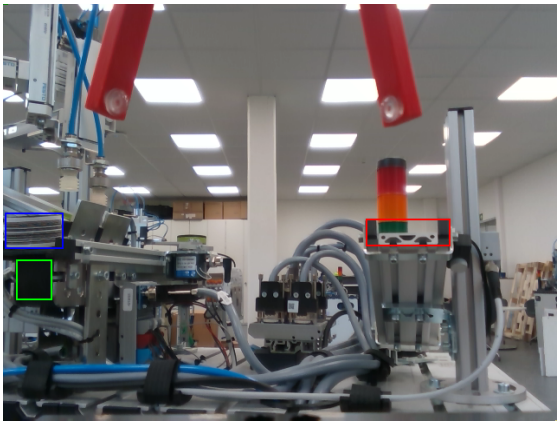


Fig. 6: Object detection with YOLO [23]. It detects objects of the three classes *conveyor belt* (green), *workpiece* (blue), and *slide* (red).

Every production step in the RCLL comes down to a pick-and-place task on or from a narrow conveyor belt that is only a few millimeters wider than the workpiece itself. Since producing a medium-high complexity product can already involve 18 pick or place operations and a single manipulation error is likely to result in total loss of the product, reliability (and therefore precision) is of paramount importance. In previous years [6], we have relied on a multi-stage procedure to detect conveyor belts of the target MPS stations. At its core, our previous approach used a model fitting approach based on the Iterative Closest Point (ICP) algorithm. It iteratively compared the current RGB/D pointcloud to a previously recorded model of the goal location (e.g., the conveyor belt) and computed a transformation from the current to the target position [2]. While this

[6] https://www.dds-foundation.org/

approach worked reliably, the iterative model matching of pointclouds made it comparably slow. Also, the approach relied on a good quality of the reference pointcloud, minor modifications to the machines often resulted in failed manipulation attempts.

For these reasons, we have replaced the pointcloud-based method by a simpler approach that only uses RGB camera images and point-based visual servoing (PBVS). It uses YOLOv4 [20,3] to detect objects in the image of the RGB camera, as shown in Figure 6. The approach works in several stages [23]:

1. As long as the object of interest has not been detected near the expected position, the robot navigates to a pre-defined position near the expected goal location.
2. As soon as an object of the correct class has been detected in proximity to the expected position, the robot's base and its gripper are positioned simultaneously, using a closed-loop position-based visual servoing approach.
3. Once the robot reaches a position near the goal position, the robot's base is stopped while the PBVS task continues to position the gripper relative to the detected object.

The visual servoing task iteratively computes the distance between the current robot's pose and the goal pose based on the current object position. Therefore, the object detection needs to be fast enough to match with the control frequency of the robot. While YOLOv4 performed better, YOLOv4-tiny was sufficiently precise and fast enough for this task.

6.1 ARUCO Tag Detection

As of 2022 the rulebook of the RCLL requires ARUCO tags [24] in order to represent type and side of each machine. In comparison to the previously used ALVAR approach, ARUCO tags are commonly used and software solutions are widely available. We opted for the OpenCV based implementation[7] which required proper integration into Fawkes. During the development we encountered the need to actively calibrate the cameras for each robot to achieve a usable reported tag pose. The ALVAR-based solution did not require active calibration.

7 Behavior Engine and High-Level Reasoning

In the following we describe the reactive and deliberative layers of the behavior components. In the reactive layer, the Lua-based behavior engine provides a set of skills. Those skills implement simple actions for the deliberative layer, which is realized by an agent based on the CX [16], a goal reasoning framework that supports multi-agent coordination.

[7] `https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html`

7.1 Lua-based Behavior Engine

In previous work we have developed the Lua-based Behavior Engine (BE) [14]. 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 target node of the edge becomes the active state. 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.

7.2 Reasoning and Planning with the CLIPS Executive

We implemented an agent based on the CLIPS Executive (CX) [16], which uses a goal reasoning model [1] based on the goal lifecycle [21]. A goal describes objectives that the agent should pursue and can either *achieve* or *maintain* a condition or state. The program flow is determined by the *goal mode*, which describes the current progress of the goal. The mode transitions are determined by the goal lifecycle, which is depicted in Figure 7. When a goal is created, it is first *formulated*, merely meaning that it may be relevant to consider. The goal reasoner may decide to *select* a goal, which is then *expanded* into one or multiple plans, either by using manually specified plans or automatic planners such as PDDL planners [15]. The reasoner then *commits* to one of those plans, which is *dispatched*, typically by executing skills of the behavior engine. Eventually, the goal is *finished* and the outcome is *evaluated* to determine the success of the goal.

7.3 Central Coordination

We utilize the CLIPS Executive framework to implement a central reasoner, which dispatches skill commands to the individual robots via the remote blackboard feature of Fawkes. In contrast to the distributed incremental approach pursued in the past [6,8], the central reasoner only maintains a single worldmodel, without the overhead of complex coordination and synchronization mechanisms required in the previous approach.

Setup An off-field laptop runs a Fawkes instance with the CLIPS Executive and its dependencies. It connects to the blackboards of the remote Fawkes instances running on each Robotino over a TCP socket by subscribing as a reader to all necessary interfaces. This allows the central agent to read data from and send instructions to the robots. The most crucial communication channel is the

Skiller interface, which is used to trigger skill execution and obtain feedback. Exploration tasks may require sensory feedback to locate machines based on their tags and laser feedback. The exploration results are then sent back to the navigator on the robots.

However, sending raw sensor data via the network can be a drawback of this setup compared to our previous distributed approach, where only processed worldmodel data was shared. This is especially critical in competitions where bandwidth and connection quality is suboptimal. To avoid this issue, the data could be pre-processed on the robot such that only the relevant information is sent, which is planned in the future.

Central Goal Reasoning In contrast to our previous incremental approach [8], our new approach focusses on a long-term strategy, driven by a two-layer system: Decisions to commit to an order result in the creation of a goal tree with all necessary goals to build the requested product. Those decisions are made by filtering all available orders according to multiple criteria, such as estimated feasibility of attached time constraints, expected points and the workload required on each machine to assemble the product. Supportive steps, such as providing material to mount rings or caps are not part of the order-specific trees, but rather are maintained dynamically in a separate tree that contains all those tasks across all pursued orders and may perform optimizations based on the requested support tasks (e.g., providing a cap to a cap station yields a waste product at that cap station, which can be used as material at a ring station if any pursued order needs it, else it needs to be discarded). Essentially, the goal creation step defines the long-term strategy and goals for specific orders persist as long as the order is actively pursued. Figure 8 shows which goals are created if an order of complexity 1 and with a single material required to mount the ring is chosen.

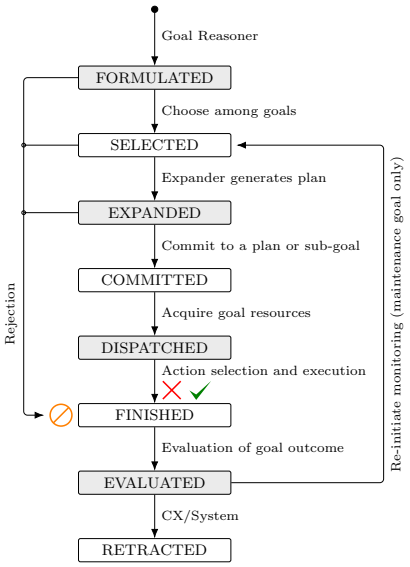


Fig. 7: The goal lifecycle with all possible goal modes [16].

The second layer consists of short-term decisions such as the distribution of robots to the respective goals, which is done lazily. Whenever a robot is idling, the reasoner evaluates the set of formulated goals that are currently executable for that robot and selects one among them. The selection is made by again filtering all the possible candidates. However, the criteria are much less complex and for now are based on a static priority of each goal (depending on the complexity of

the belonging order), as well as constraints imposed through the corresponding goal tree (e.g., if goals must be executed in sequence).

Execution Monitoring In order to become resilient to failures during the game, we handle execution errors in similar fashion as the distributed agent did: The damage is assessed, the worldmodel is updated accordingly and recovery methods are invoked if necessary (e.g., retrying failed actions or removing a goal tree, when the associated product is lost). In addition to the implications of a failed goal, the central agent needs to recognize a robot that not responding, de-allocate assigned tasks from it and react on successful maintenance. This is realized through a heartbeat signal, which is sent periodically by each robot. Lastly, the central agent itself may suffer a critical error that completely shuts it down. Even then the system is able to pick up on the work done so far by maintaining a persistent backup of the current worldmodel in a database and by restoring it, if necessary.

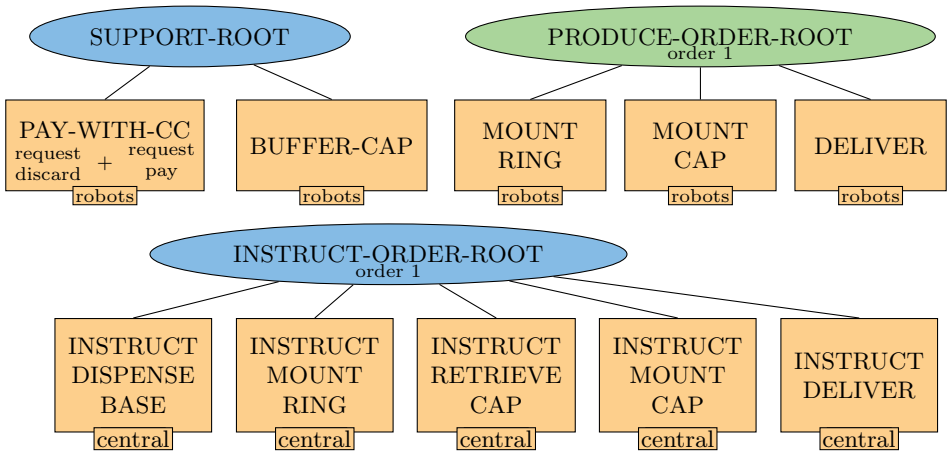


Fig. 8: Structure of goal trees for a single order. Ellipsis nodes are inner nodes, where blue ones always select the highest executable goal among them, while green ones only select the left-most child goal. Orange nodes denote the actual goals that are physically executed, either through the robots, or by the central instance itself, incase it only involves communication with the refbox.

8 Conclusion

In 2022, we have deployed a central agent based on the CLIPS Executive, which provides an explicit goal representation including plans and actions with their preconditions and effects. Focus was put on making explicit decisions about the orders that should be started. We replaced our established reasoner that grew over many past competitions, because we believe that its heavy focus on robustness made substantial tradeoffs to peak performance, which is not necessary anymore. This is mainly due to our matured gripping system. Unexpected

side effects due to the dependency on a centralized reasoning server arrived in the form of WiFi issues, which we managed to overcome by utilizing cellular network-based communication with the help of a VPN and USB tethering from off-the-shelf smartphones.

Our perception setup is extended by a machine learning approach to detect workpieces, conveyor belts, and slides, which is used to approach the object with closed-loop visual servoing. This procedure turned out to be more robust and faster compared to our previous pointcloud-based approach.

A lot of effort was put into the integration of ROS 2 and towards multi-agent path planning to ensure fast and reliable navigation in tight and narrow environments. While not finished yet, important steps such as bridging Fawkes and ROS 2 are already implemented. We recon that completely exchanging most of our major components was an ambitious roadmap and the ROS 2 integration could not be finished yet, due to the amount of other tasks we worked on. Nevertheless, we believe that the decision to switch to ROS 2 is right and will benefit us in future competitions.

Acknowledgements. The team members in 2022 were Tom Hagdorn, Mike Henning, Nicolas Limpert, Hendrik Nessau, Simon Roder, Matteo Tschesche, Daniel Swoboda and Tarik Viehmann.

We gratefully acknowledge the financial support of RWTH Aachen University and FH Aachen University of Applied Sciences.
This work is partially supported by the German Research Foundation (DFG) under grants *GL-747/23-1*, *2236/1* and under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612. This work is partially supported by the EU ICT-48 2020 project TAILOR (No. 952215).

References

1. Aha, D.W.: Goal Reasoning: Foundations, Emerging Applications, and Prospects. AI Magazine **39**(2) (Jul 2018)
2. Besl, P.J., McKay, N.D.: Method for registration of 3-d shapes. In: Sensor fusion IV: control paradigms and data structures. vol. 1611, pp. 586–606. Spie (1992)
3. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: Optimal speed and accuracy of object detection (2020), `https://arxiv.org/abs/2004.10934`
4. Donenfeld, J.A.: Wireguard: next generation kernel network tunnel. In: Proceedings of the Network and Distributed System Security Symposium (NDSS). pp. 1–12 (2017)
5. Gat, E.: Three-layer architectures. In: Kortenkamp, D., Bonasso, R.P., Murphy, R. (eds.) Artificial Intelligence and Mobile Robots, pp. 195–210. MIT Press (1998)
6. Hofmann, T., Limpert, N., Mataré, V., Ferrein, A., Lakemeyer, G.: Winning the RoboCup Logistics League with Fast Navigation, Precise Manipulation, and Robust Goal Reasoning. In: RoboCup 2019: Robot World Cup XXIII. pp. 504–516. Springer International Publishing (2019)
7. Hofmann, T., Mataré, V., Neumann, T., Schönitz, S., Henke, C., Limpert, N., Niemueller, T., Ferrein, A., Jeschke, S., Lakemeyer, G.: Enhancing software and hardware reliability for a successful participation in the RoboCup Logistics League

2017. In: RoboCup 2017: Robot World Cup XXI. pp. 486–497. Springer International Publishing (2018)
8. Hofmann, T., Viehmann, T., Gomaa, M., Habering, D., Niemueller, T., Lakemeyer, G.: Multi-agent goal reasoning with the CLIPS Executive in the Robocup Logistics League. In: Proceedings of the 13th International Conference on Agents and Artifical Intelligence (ICAART) (2021). https://doi.org/10.5220/0010252600800091
9. 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)
10. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: RoboCup: The Robot World Cup Initiative. In: Proc. 1st Int. Conf. on Autonomous Agents (1997)
11. Macenski, S., Martín, F., White, R., Clavero, J.G.: The marathon 2: A navigation system. arXiv preprint arXiv:2003.00368 (2020)
12. 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)
13. 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)
14. Niemueller, T., Ferrein, A., Lakemeyer, G.: A Lua-based Behavior Engine for Controlling the Humanoid Robot Nao. In: RoboCup Symposium 2009 (2009)
15. Niemueller, T., Hofmann, T., Lakemeyer, G.: CLIPS-based execution for PDDL planners. In: ICAPS Workshop on Integrated Planning, Acting and Execution (IntEx) (2018)
16. Niemueller, T., Hofmann, T., Lakemeyer, G.: Goal reasoning in the CLIPS Executive for integrated planning and execution. In: Proceedings of the 29th International Conference on Planning and Scheduling (ICAPS) (2019)
17. 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)
18. Niemueller, T., Zug, S., Schneider, S., Karras, U.: Knowledge-Based Instrumentation and Control for Competitive Industry-Inspired Robotic Domains. KI - Künstliche Intelligenz **30**(3), 289–299 (2016)
19. 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)
20. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
21. 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)
22. Thomas, D., Woodall, W., Fernandez, E.: Next-generation ROS: Building on DDS. In: ROSCon Chicago 2014. Open Robotics, Mountain View, CA (sep 2014). https://doi.org/10.36288/ROSCon2014-900183
23. Tschesche, M.: Whole-Body Manipulation on Mobile Robots Using Parallel Position-Based Visual Servoing. Master's thesis, RWTH Aachen University (May 2022), `https://kbsg.rwth-aachen.de/theses/tschesche2022.pdf`
24. Wubben, J., Fabra, F., Calafate, C.T., Krzeszowski, T., Marquez-Barja, J.M., Cano, J.C., Manzoni, P.: Accurate landing of unmanned aerial vehicles using ground pattern recognition. Electronics **8**(12), 1532 (2019)