The Carologistics RoboCup Logistics Team 2019

Till Hofmann[1], Nicolas Limpert[2], Victor Mataré[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 Carologistics team participates in the RoboCup Logistics League for the eighth 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 2019 are David Bosen, Mario Claer, Sebastian Eltester, Christoph Gollok, Mostafa Gomaa, Daniel Habering, Till Hofmann, Nicolas Limpert, Morian Sonnet and Tarik Viehmann.

This paper is based on the last year's team description [6].

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. Doctoral, master, and bachelor students of both 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 team has participated in RoboCup 2012–2017 and the RoboCup German Open (GO) 2013–2019. 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.

In the following we describe some of the challenges of the RCLL with a focus on the changes introduced in 2019. In Section 2 we give an overview of the hardware changes we are going to deploy in 2019. We continue by describing updates to the functional software components in Section 3 and to behavior components in Section 4 before concluding in Section 5.

1.1 RoboCup Logistics League 2019

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. [7,14,12]). After the league switched from purely symbolic production to Festo's physical Modular Production System (MPS) in 2015 [14], the rules and field layout have been incrementally refined to focus on challenges that are relevant to the Industry 4.0 movement [13].

For 2019, the biggest rule change is the introduction of *competitive orders*. For regular orders, each team can deliver the requested product and scores independently of the other team. For a competitive order, the team that delivers first gets bonus points, while the second team gets points deducted. For this year, there will be only one competitive C_0 order during the regular game, which may be increased in the future.

On the technical side, the MPS stations will be equipped with barcode scanners, which will allow the tracking of workpieces, and thus to give points for (partial) production steps.

2 The Carologistics Platform

The standard robot platform of this league is the Robotino by Festo Didactic [5]. As in previous years, we are using the Robotino Version 3 with many hardware modifications and a custom software stack starting from the OS level. Most of our hardware modifications have been described in our previous team description papers, so here we will focus on new developments only.

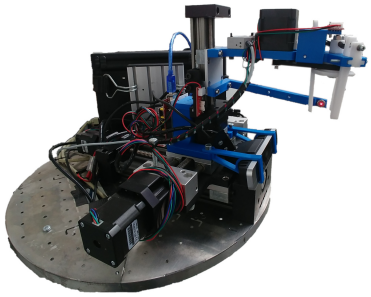

Fig. 1. New manipulator with 3 linear axes and self-centering gripper

2.1 Gripper System

The gripping system is an incremental update to the 3-axis system developed in 2018 [4]. The assembly of the linear axes is mostly kept the same, except for some refinements to 3D-printed connective parts to save filament and increase stability. The gripper itself has been redesigned to grip the workpiece from above with three instead of two fingers (cf. Figure 1). The advantage we hope to gain from this is increased robustness and precision because the workpiece will always center between the three spring-loaded fingers, independently of the positioning error. Another advantage of the new design is that it retracts fully behind the robot's circular base shape, which should significantly reduce the risk of damage and simplify path planning.

2.2 Architecture and Middleware

The software system of the Carologistics robots combines two different middlewares, Fawkes [8] 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 [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. Changes to the lowest layer are 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 [8].

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 a new path planning module and a revised conveyor belt detection.

3.1 Path Planning

Since our change to ROS *move_base* in 2017, the whole navigation system has proven to be a flexible and reliable approach well-suited for the difficult navigation tasks in the RCLL. The classic separation between global and local path planning is of major use to quickly react to obstacles crossing the way in a small local frame. The global path planner does not have to take kinematic constraints into account as the Robotino base can move freely in x, y, θ at any time, given that that there is enough free space around the robot.

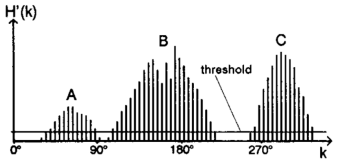

Fig. 2. Vector Field Histogram representing 3 different obstacles and the selected angle of motion [2].

Since 2017, we initially made use of the teb_local_planner as a controller running within the *move_base* responsible for executing the planned path. The major advantage of this planner among many others available is its strafing capability. However, due to the kinematic freedom of the platform, we decided that the complex workflow of the *teb_local_planner* is not needed for us. As such, we decided to implement our own local planner based on the Vector Field Histogram approach [2], giving us a loop rate of roughly 90 Hz and as such higher translational and angular velocities on average. The general idea of VFH is to identify obstacles given a discrete set of laser beams (and as such being robust against sensor noise) and select a motion direction given the histogram

and the actual goal. The actual goal of VFH is the current goal on the global path - Figure 2 shows an example histogram.

With the changes addressed in 2.1 we have even less motion constraints and can finally plan with a point-mass with respecto to the robot being a round-shaped platform with no mechanical parts pointing outwards while the robot is moving.

During our usage of the *move_base* in general and particularly its very fixed behaviour defining a strict failure recovery procedure, we discovered that it makes sense to switch to a more flexible approach. Also, the changes for multi-agent path planning introduced in 2018 were implemented around the *move_base* without being a proper solution for multi-agent path planning. As such we decided to switch to ROS' robot_navigation locomotor[3] in general and in particular a fork of its locomove_base implementing the multi-agent solution introduced last-year in a single clean solution. We can now properly select what should happen when the globally planned path would actually cross the path of another robot, without having to give temporary goals to *move_base*.

3.2 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 2019, we will refine the ICP-based method developed in 2018 [4]. In 2018, the initial estimate for the conveyor pose was generated from a RANSAC-based line detection algorithm that uses the data from the SICK laser range finders to determine the approximate position of MPS table's flat side panel. From there, the approximate position of the conveyor belt can be calculated since its mounting position on the MPS table is known. However, tests and usage during RoboCup and German Open 2018 have shown that this method is the main source of errors since bad initial estimates tend to pull the correspondence optimizer towards an incorrect local optimum.

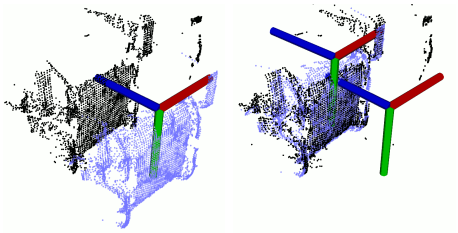

Fig. 3. Left: Model pointcloud (blue) roughly aligned to scene (black) based on initial guess. Right: After running ICP, the model is aligned to the scene precisely.

While the hypothesis verification [18] does eliminate most false positives, each failed attempt usually costs 1-5 seconds of precious game time. The problem is aggravated by the fact that the noise in the line detection increases the closer the robot moves towards the MPS, since the angle between the MPS panel and the laser beams hitting its edges becomes sharper.

So for 2019, the focus is on increasing the reliability of the initial estimate by incorporating additional data. One approach will be to use the previous plane-

[3] `https://github.com/locusrobotics/robot_navigation`

fitting approach [17] since it is fast and much more robust against bad initial estimates, thus forming a three-stage pipeline:

1. Laser-based line detection: Imprecise, but fast, robust and requires no initial estimate.
2. Plane fitting: Moderately precise, fast and robust against bad initial estimates.
3. ICP: Very precise, but slow and sensitive to bad initial estimates.

4 High-level Decision Making and Task Coordination

The behavior generating components are separated into three layers, as depicted in Figure 4: 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.

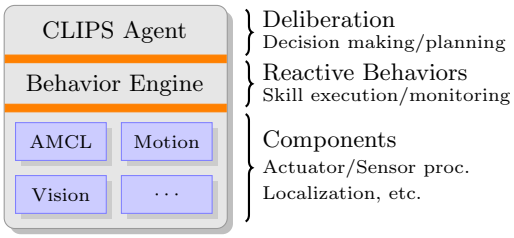

Fig. 4. Behavior Layer Separation [12]

The robot group needs to cooperate on its tasks, that is, the robots communicate information about their current intentions, acquire exclusive control over resources such as locations or machines, and share their beliefs about the current state of the environment.

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

4.2 Reasoning and Planning with the CLIPS Executive

We implemented an agent based on the CLIPS Executive (CX) [11], which uses a goal reasoning model [1]. 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 5. 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 [10]. The reasoner then *commits* to one of those plans, which is *dispatched*, typically by executing a skill of the behavior engine. Eventually, the goal is *finished* and the outcome is *evaluated* to determine the success of the goal.

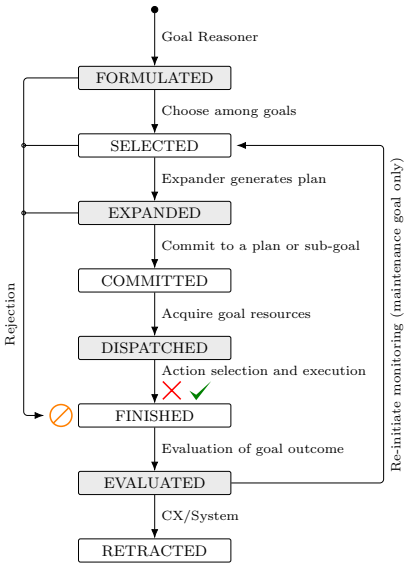

Fig. 5. The goal lifecycle with all possible goal modes [11].

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 [10,15], we currently do not use a planner, but instead use pre-defined plans.

4.3 Multi-Robot Coordination

The CX also provides means for multi-robot coordination, in particular *world model synchronization*, *mutual exclusion*, and *resource allocation* [11]. To cooperate effectively, each agent must share (parts of) its world model with the other agents. The CX implements world model synchronization using a shared database [16,20]. Each robot uses a database instance for local (agent-specific) and global (shared) world model facts. The global world model database is synchronized as part of a replica set with the global instances of the other robots.

Based on the replicated database, the CX also implements a locking mechanism. To lock a mutex, an agent must request a *majority acknowledgement*, thereby avoiding two agents to hold the same mutex. To allow *mutual exclusion*,

the CX specifies two actions *lock* and *unlock*, which may be used by the agent just as any other action. Additionally, each goal may be associated with one or multiple resources that are required in order to dispatch the goal. If one resource is currently unavailable, the goal is *rejected*. Once a goal is *retracted*, its acquired resources are released automatically.

5 Conclusion

In 2019, we are continuing the development of an agent based on the CLIPS Executive, which provides an explicit goal representation including plans and actions with their preconditions and effects. Work on the mechanical side of the manipulator system is focused on a radical redesign of the gripper itself. The conveyor detection pipeline is extended to combine the benefits and eliminate the downsides of all previous approaches. For path planning, we switched from *move_base* to *robot_navigation* with a new implementation of multi-agent path planning.

The website of the Carologistics RoboCup Team with further information and media can be found at `https://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[4].
We appreciate the financial and organizational support by the Cybernetics Lab IMA & IfU, RWTH Aachen University.
We gratefully thank T. Niemueller for his continued support and contributions both to the league and to our team.

References

1. Aha, D.W.: Goal Reasoning: Foundations, Emerging Applications, and Prospects. AI Magazine **39**(2) (Jul 2018)
2. Borenstein, J., Koren, Y.: The vector field histogram-fast obstacle avoidance for mobile robots. IEEE transactions on robotics and automation **7**(3), 278–288 (1991)
3. 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)
4. Hofmann, T., Limpert, N., Mataré, V., Schönitz, S., Niemueller, T., Ferrein, A., Lakemeyer, G.: The Carologistics RoboCup Logistics Team 2018. Tech. rep., RWTH Aachen University and FH Aachen University of Applied Sciences (2018)
5. 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)
6. Neumann, T., Hofmann, T., Mataré, V., Henke, C., Schönitz, S., Niemueller, T., Ferrein, A., Jeschke, S., Lakemeyer, G.: The Carologistics RoboCup Logistics Team 2017. Tech. rep., RWTH Aachen University and FH Aachen University of Applied Sciences (2017)

[4] `http://gepris.dfg.de/gepris/projekt/288705857`

7. 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)
8. 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)
9. Niemueller, T., Ferrein, A., Lakemeyer, G.: A Lua-based Behavior Engine for Controlling the Humanoid Robot Nao. In: RoboCup Symposium 2009 (2009)
10. Niemueller, T., Hofmann, T., Lakemeyer, G.: Clips-based execution for pddl planners. In: ICAPS Workshop on Integrated Planning, Acting and Execution (IntEx) (2018)
11. 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)
12. 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)
13. 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)
14. 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)
15. 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 (Jun 2017)
16. Niemueller, T., Lakemeyer, G., Srinivasa, S.: A Generic Robot Database and its Application in Fault Analysis and Performance Evaluation. In: IEEE International Conference on Intelligent Robots and Systems (IROS) (2012). https://doi.org/10.1109/IROS.2012.6385940
17. 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)
18. Papazov, C., Burschka, D.: An efficient ransac for 3d object recognition in noisy and occluded scenes. In: Asian Conference on Computer Vision. pp. 135–148. Springer (2010)
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. Zwilling, F.: A Document-Oriented Robot Memory for Knowledge Sharing and Hybrid Reasoning on Mobile Robots. Master's thesis, RWTH Aachen University (2017)