Simulation for the RoboCup Logistics League with Real-World Environment Agency and Multi-level Abstraction

Frederik Zwilling, Tim Niemueller, and Gerhard Lakemeyer

Knowledge-based Systems Group, RWTH Aachen University, Germany
{zwilling, niemueller, gerhard}@kbsg.rwth-aachen.de

Abstract. RoboCup is particularly well-known for its soccer leagues, but there are an increasing number of application leagues. The newest one is the Logistics League where groups of robots take on the task of in-factory production logistics. It has two unique aspects: a game environment which itself acts as an agent and a focus on planning and scheduling in robotics. We propose a simulation based on Gazebo that takes these into account. It uses the exact same referee box to simulate the environment reactions similar to the real game and it supports multiple levels of abstraction that allow to focus on the planning with a high level of abstraction, or to run the full system on simulated sensor data on a lower level for rapid integration testing. We envision that this simulation could be a basis for a simulation sub-league for the LLSF to attract a wider range of participants and ease entering the robot competition.

1 Introduction

Research on autonomous mobile robots in industrial applications has significantly increased during the last years. In industry, mobile robots are the most complex variant of cyber-physical systems, embedded devices that combine computational resources with physical interaction. The context is the Industry 4.0 movement whose goals are adaptive production capabilities, where material flows and production processes are dynamic and factories can output a variety of product types. In this scenario, a group of mobile robots can be used to move material and handle machines, eventually delivering the resulting products.

The RoboCup Logistics League Sponsored by Festo (LLSF) strives to address these issues in a competitive scenario to foster research, system integration, and robotics education relevant to industry. While aspects like competition and scoring points are clearly influenced by RoboCup, other aspects like the referee box as an order issuing system and time as a critical factor are driven by demands from industrial applications. Many basic robotics problems like self-localization, collision avoidance and perception must also be tackled in the LLSF. The character of the 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 LLSF, 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* to support research and development. There are three core aspects in this context:

1. The simulation should be a turn-key solution with simple interfaces,
2. the world must react as close to the real world as possible, including in particular the machine responses and signals, and
3. various levels of abstraction are desirable depending on the focus of the user, e.g. whether to simulate laser data to run a self-localization component or to simply provide the position (possibly with some noise).

With this work, we provide such an environment. It is based on the well-known Gazebo simulator addressing these issues: (1.) its wide-spread use and open interfaces in combination with our models and adapters provides an easy to use solution; (2.) we have connected the simulation directly to the referee box, the semi-autonomous game controller of the LLSF, so that it provides precisely the reactions and *environment agency* of a real-world game; (3.) we have implemented *multi-level abstraction* that allows to run full-system tests including self-localization and perception or to focus on high-level control reducing uncertainties by replacing some lower-level components using simulator ground truth data. This allows to develop an idealized strategy first, and only then increase uncertainty and enforce robustness by failure detection and recovery.

We propose a new simulation sub-league for the LLSF based on the Gazebo simulator at different levels of difficulty using the multi-level abstraction, to attract more teams and ease entering the LLSF robotics competition.

In Sect. 2 we give a brief introduction to the LLSF, followed by related work in Sect. 3, before we get into the details of the simulation in Sect. 4. We describe several applications of the simulation in game strategy evaluation in Sect. 5. We propose a simulation sub-league and conclude in Sect. 6.

2 RoboCup Logistics League

RoboCup is an international robotics competition particularly well-known for its various soccer leagues. The Logistics League Sponsored by Festo (LLSF) is an application-oriented major league since 2012 regarding simplified production logistics. Groups of up to three robots have to plan, execute, and optimize the material flow in a factory automation scenario and deliver products according to dynamic orders. Therefore, the challenge consists of creating and adjusting a production plan and coordinate the group of robots [1].

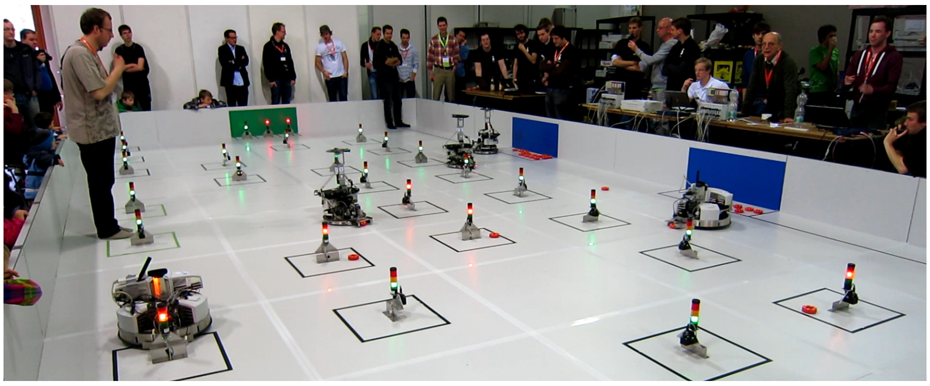

Fig. 1. Carologistics (three robots with omni-vision tower) and TUMBendingUnits (robots on the left and right) during the LLSF finale at the German Open 2014

Since 2014, the two formerly separate playing fields have been merged into one $11.2\,m \times 5.6\,m$ in size (cf. Fig. 1). Two teams are playing at the same time competing for points, (travel) space and time. Each team has an exclusive input storage (blue areas) and delivery gates (green area). Machines are represented by the RFID-readers with signal-lights on top. The signal-lights indicate the current status of a machine, such as "ready", "producing" and "out-of-order". There are three delivery gates, one recycling machine, and twelve production machines per team. Material is represented by orange pucks with an RFID tag. At the beginning all pucks have the raw material state S_0, and can be refined through several stages to final products using the production machines. These machines are assigned a type which determines what inputs are required and what output will be produced, and how long this conversion will take. Fig. 2

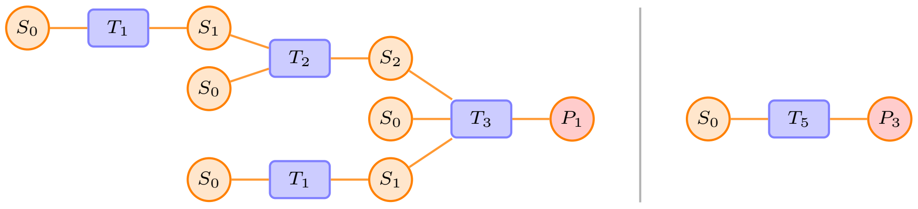

Fig. 2. Production Chains for two types of products.

shows the production trees for two final products, P_1 and P_3. The latter is rather simple, it requires only a single step, but it scores only a small amount of points. For P_1, four refinement steps are required. Only raw material pucks are available at the beginning, all others must be produced by the robot using the appropriate machines. The machines are distributed randomly across the field (mirrored at the narrow middle axis so both teams have the same travel distances). The rulebook [2] describes the game in more detail. For 2015, the league has decided to introduce real production steps through the use of the Festo Modular Production System [3] to make the game easier to understand.

The game is controlled by the referee box (refbox), a software component which keeps track of puck states, instructs the light signals, and posts orders to the teams [3]. After the game is started, no manual interference is allowed, robots receive instructions only from the refbox. Teams are awarded with points for delivering ordered products, producing complex products, and recycling.

The standard robot platform of this league is the Robotino by Festo Didactic [4]. The Robotino was developed for research and education and features omni-directional locomotion, a webcam, infrared distance sensors and bumpers. It is also equipped with a static puck holder to move pucks. The teams may equip the robot with additional sensors and computation devices. For example, the robots of the Carologistics team in Figure 1 are equipped with a laptop, a omni-directional camera and a laser range finder. In 2014, Festo released the new version 3 of the robot featuring better driving, computing unit, and extensibility.

3 Related Work

There is a plethora of existing robotic simulations. Here, we mention two particular examples used in other RoboCup leagues and the Robotino simulator provided by Festo as a comparison. We also mention the Fawkes robot software framework and related systems, as it is used for agent strategy evaluation and as a prototyping development environment for the simulation.

3D Soccer Simulation League (3DSSL) The 3DSSL is based on the Open Source multi-robot simulator SimSpark and currently uses the Nao as model for the players. Software agents communicate with a soccer server to announce their actions but also to communicate with other players. SimSpark features multi-platform support, is scriptable via Ruby and supports automatically enforced rules to realize soccer rules. Today, solutions for body control and perception must be developed similar to as if a real robot were used. The agent has to take limited skills of the simulated robot and possible problems in their execution into account [5]. SimSpark is mostly specialized on the 3DSSL, thus Gazebo brings many more existing models for, e.g., sensors we could reuse. Especially due to the extensibility of the robots in the LLSF, we therefore prefer Gazebo.

Rescue Simulation League (RSL) The RSL aims to benchmark software agents and robots in a disaster scenario. A part of the RSL is the Virtual Robot Competition (VRC) which has the task to find victims in a disaster environment with a team of robots and a human operator. Important research issues of this league are victim detection, utility-based mapping of the environment, autonomous navigation and multi-robot coordination. To foster transfer of the research results to a real application, the simulation features graphical and physical realism [6]. The VRC is based on the multi-robot simulator USARSim. Initially focused on urban search and rescue simulations it has evolved into a general purpose simulator which is also used to simulate RoboCup@Home scenarios [7]. USARSim uses the Unreal graphics and PhysX physics engine. These allow a graphically and physically realistic environment. The simulator can be interfaced with ROS what allows having the same interface as on a real robot [8,9]. We preferred Gazebo instead of USARSim because of the plethora of available models and because USARSim relies on non-free components like the Unreal engine.

Robotino Sim Professional Robotino Sim Professional[1] is a simulator for the Robotino developed by its manufacturer Festo. It is used for the Robotino in general. An environment resembling the LLSF is *not* provided. Its closed nature make it rather hard to extend or modify and it does not run on the Linux operating system. Therefore it is unsuitable as an open simulation.

3.1 Fawkes

Robot software systems are increasingly complex in the number of components as well as in their interactions. Therefore, a robot software framework which provides a suitable middleware, basic modules and auxiliary libraries is typically used. An often used candidate is the *Robot Operating System* [10] (ROS). In this work, we use the component-based framework Fawkes[2] [11]. It uses a hybrid blackboard and messaging middleware for inter- and intra-component communication. Compared to ROS, it supports a closer integration of the various components. These are implemented as run-time loadable plugins consisting of one or more threads. These threads can be invoked coordinated in a common main loop or run concurrently.

We use Fawkes for two primary functions. First, it is used to implement and integrate the software components that drive our robot, including self-localization, navigation and collision avoidance, and behavior control. Fawkes serves as one particular example how to connect to the simulation. Similarly other frameworks like ROS could be used. Second, we have implemented simulated inter-robot communication and the connection between the referee box and the simulation as Fawkes plugins. We intend to integrate these two aspects into Gazebo plugins for general use at a later time. We will give more details on this in the following section.

4 Simulation

A robotic simulation is a tool to ease testing and debugging of robotic applications, and in the context of the LLSF we also want to lower the entry barrier to participate. We have developed a simulation of the LLSF based on Gazebo [12]. An example scene is shown in Fig. 3. The simulation is designed to achieve the five goals realism, multi-level abstraction, compatibility with the real robot,

[1] `http://www.festo-didactic.com/int-en/learning-systems/software-e-learning/robotino-sim-view/robotino-sim-professional.htm`

[2] Fawkes is Open Source software and available at `http://www.fawkesrobotics.org`

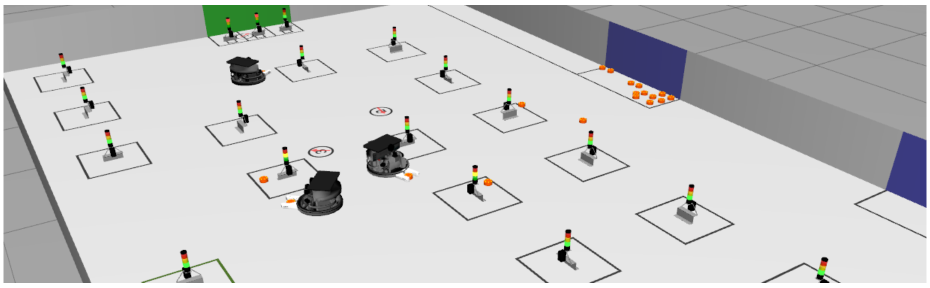

Fig. 3. The simulation of the LLSF in Gazebo. The circles above the robots indicate their localization and robot number.

expandability and allowing multi-robot strategy evaluation. The realism determines how similar a robotic system behaves in the simulation and the real world. That includes sensor data, physical and logical behavior in the simulation that is similar to their real-world counterpart, e.g. with similar noise and precision. Compatibility with the real robot allows the robot software to operate in the simulation with the same interfaces as on the real robot. This allows an easy transfer from a robot system that works in the simulation to the real world. The LLSF is a multi-robot scenario. Therefore we need to simulate multiple robots at the same time efficiently. The LLSF allows additions of arbitrary sensors and the rules are steadily evolving [3]. Therefore the simulation must be expandable.

4.1 Gazebo

As a basis for our simulation, we use the Open Source robot simulator Gazebo [13] (`http://gazebosim.org`). It can simulate various robots and their environment in a 3D world and is used in many applications, for example in the *DARPA Virtual Robotics Challenge* which is about solving challenging tasks in a disaster scenario and requires a high realism of the simulator. Gazebo uses the *Ogre* rendering engine for graphically realistic environments with reflections, shadows and detailed textures. This is important for the simulation of camera sensors as reflections of light sources and inhomogeneous lighting can cause problems. Gazebo can use both physics engines *Open Dynamics Engine (ODE)* or *Bullet*.

A Gazebo simulation environment is described as a *world* which contains certain objects according to models, the light sources, and parameters e.g. for the physics simulation or rendering. *Models* describe physical objects such as a table or a robot. The model is built out of *links* of various geometries and *joints* which connect two links and define possible relative motions. Beside the physical and visual description of objects, there are also *plugins*. These consist of executable code that interacts with the simulation at run-time. They typically model the behavior of objects. For example, there are plugins for getting sensor data, applying motor commands and world-plugins for spawning new models. We will also use plugins later to connect the simulation to the LLSF referee box.

Out of the box, Gazebo includes a variety of generalized sensors and models of common robots, sensors and environment-objects. In the Gazebo framework, there is support for a variety of sensors. This includes cameras, contact-sensors, GPS, inertial measurement units, laser and sonar sensors. This is important as arbitrary additions of sensors are allowed in the LLSF. To adapt the simulation for a specific team, these models can make this process faster. Gazebo provides modeling and programming interfaces to extend the simulation to specific needs. Gazebo also provides a connection to ROS.

As we have built our Robotino system using Fawkes, we have replaced the hardware accessing plugins by simulation adapters. The interfaces towards the other components remain unchanged, such that sensor processing, path planning, or behavior control continue to function without modifications.

Protocol Buffers (protobuf) are a data interchange format developed by Google[3]. Given a message specification it generates native (de-)serialization code for various programming languages. Protocol buffers are used for Gazebo's internal communication and as an external interface by means of a publisher/subscriber model (orange connections in Fig. 4). The refbox uses protobuf messages to communicate with robots (UDP broadcast) and the simulation (multiplexed stream protocol). Additionally we have implemented a module to simulate typical wifi communication problems like latency or packet loss.

4.2 Architecture

The basic parts for the simulation environment are referenced in the world file, which contains the walls, ground floor, the machines, pucks, and markings. In Fig. 4, the middle box represents the Gazebo process. Based on the models (red) Gazebo can run and visualize the environment. It provides an application programming interface (API, green). The protobuf-based messaging middleware is used for internal communication as well as an external interface. Gazebo also hosts a number of plugins (blue). These plugins are active components (executable code) that typically maintain, provide, and process simulation data. For example, the LLSF environment plugin communicates with the refbox to provide the same field reactions as in the real world. If a puck is moved under a machine, a message is sent to announce this to the refbox, which in return sends instructions how the light signals change. The robot actually consist of a number of plugins, typically one per sensor or actuator. We use, for example, the built-in plugin for the laser data, and have a custom plugin for the robot movements.

The robot software itself is connected again using protobuf. It gathers data from and sends instructions to the robot-specific plugins. For now, we are using Fawkes for this connection. But the existing Gazebo-ROS integration would make the adaptation simple. Also, the refbox connection is currently proxied through a Fawkes plugin, which we will change before the public release such that the simulation itself can be be accessed from any framework.

To handle the simulation speed, which can be smaller than real-time on a slow computer or greater to speed up automated test runs, we publish the

[3] `https://developers.google.com/protocol-buffers`

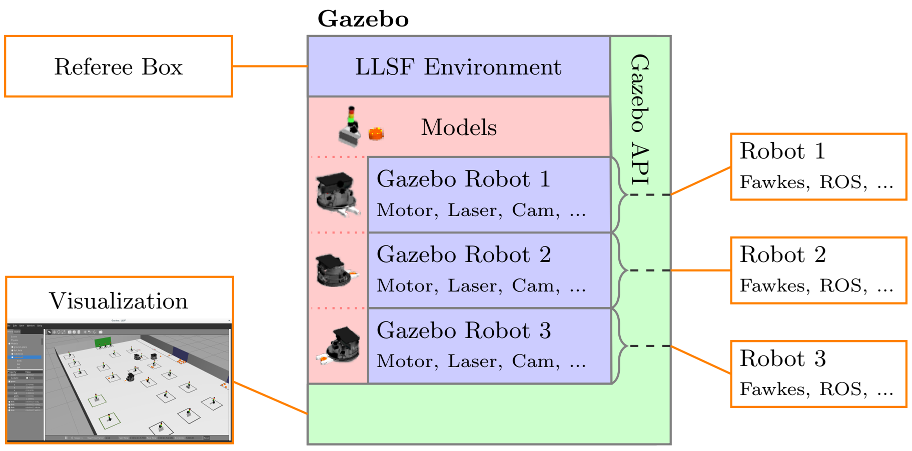

Fig. 4. Architecture overview of the simulation; blue means Gazebo plugins, red models, green Gazebo API and middleware, orange are components connected via protobuf middleware. The robots are driven by a system like Fawkes or ROS, referee box and visualization operate and visualize the environment.

simulation time and the current simulation speed, which can be used to estimate the simulation time between two synchronization messages. We extended the refbox to use the simulation time. In Fawkes, the simulation time is provided instead of the system time.

4.3 Simulation Interfaces

We provide three different ways to interact with the simulation: direct interaction using native Gazebo messaging, accessing the simulation through Fawkes, or using our Behavior Engine instruction interface. With this variety of interfaces it is possible to use existing own software, or re-use (parts of) our components.

Direct Interaction Using Gazebo's protobuf-based middleware most aspects of the simulation are directly accessible via topics. For example, models can be directly manipulated, e.g. to implement specific robot modifications. It also allows direct access to internal data, e.g. from additional sensors. The direct interaction is also accessible from the ROS connection provided by Gazebo.

Fawkes Our system already comes with many components for interaction with the simulation. Communication with these components happens over the Fawkes middleware. For example, Fawkes provides access to generated camera images to run a perception module, or directly access the puck positions over the blackboard for a higher abstraction level. The interfaces are unified across simulation and the real robots, allowing a seamless migration between the two.

Lua-based Behavior Engine (BE) The BE provides a framework and tools for the development, execution, and monitoring of reactive behaviors. It forms a mid-level layer between the high-level reasoning and the low-level execution. It models basic skills as hierarchical hybrid state machines and typically handles parameter estimation (e.g. determining coordinates for an entity name) and basic failure recovery (for local failures like losing a product while traveling). Skills are exposed to the high level system as function for interleaved execution and provide success/failure and error information feedback. For details of the modeling and execution we refer to [14]. By using the Behavior Engine, it is possible to reuse our set of basic skills and build an agent on top. It also allows for the composition of skills to form more complex skills or even an agent itself.

4.4 Multi-level abstraction (MLA)

MLA [15] is the ability to choose to either simulate low-level sensor data or to directly extract higher-level information. In terms of simulation interfaces, the abstraction level is defined based on the Gazebo topic chosen as input data. In Fawkes, this manifests in different components which are used to access simulation data. For example, as depicted in Fig. 5, the simulation (red boxes) provides both, laser data and pose information in different topics. In Fawkes, either an accessor plugin is used (higher abstraction), or the simulated laser data is fed to the localization component (lower abstraction). Additionally, for actuation MLA, we provide the BE as a basis for high-level control software. Note that Fawkes and the BE serve as one particular example, but they are not a requirement. Similar structures can be implemented for example using ROS.

While this is not an aspect unique to our simulation, the explicit specification is necessary to clarify the capabilities of the simulation, in particular to attract new users from the planning community. MLA is also useful during development, as it allows to perform full integration tests with all software components (including sensor processing), or to focus on high-level evaluation.

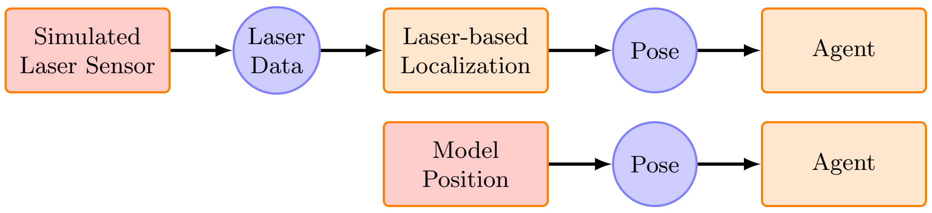

Fig. 5. Multi-level abstraction example: the simulation can either provide the data to run a laser-based self-localization component or directly provide the position.

5 Applications

We have used the simulation for rapid testing, performance evaluation and as a development environment. We have also conducted experiments to verify the similarity between simulation and real world robot behavior.

5.1 Agent Strategy Evaluation

With a focus on the planning and scheduling aspects in the LLSF, efficient testing of such systems becomes an important concern. Our first application of the simulation is aimed at the evaluation of different game strategies. We have developed tools to run several games unattended over night and present the results later for comparison, e.g. in terms of achieved score. Our reasoning component, based on the CLIPS rules engine, implements incremental task-level reasoning [16]. We evaluated different numbers of robots on the field or static and dynamic strategies, where robots would either pursue the same goal (like producing a P_1 product) the whole time or where they could change their role at run-time [12]. This allowed us to steer our development efforts more informed to improve the system's overall performance.

The CPU usage of a simulation game on a desktop with an Intel Core i7-3770 at 3.40 GHz with 16 GB of RAM is shown in Fig. 6. The system was utilized at about 60% of capacity including the simulation and 3 robots running all necessary components except visual perception which was given by the simulation.

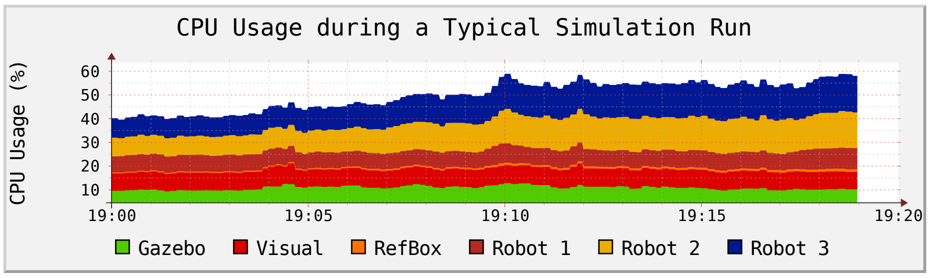

Fig. 6. CPU usage during a simulation game; X axis shows the system time during the experiment; Y axis areas show percent of CPU time used stacked on top of each other.

5.2 Simulation vs. Real World

There is typically a gap between the simulation and the real robot. For example, the lighting of the scene tends to be more uniform and stable in a simulation. Still some gaps can be bridged, for example using accurate friction parameters for the pucks or to have similar noise and precision in the laser data. We have compared simulation runs to our performance at the German Open 2014. We ran the same software versions, the only differences being that hardware modules were replaced by simulation modules and machine-signal perception is given by the simulation. In ten simulated games the system achieved an average score of 75.3 points with a standard deviation of 10.1. In the last five games of the competition we achieved 61.6 points with a standard deviation of 11.6. Overall teams scored up to 71 points with average 19.1 ± 23.3. While this is certainly not an exhaustive comparison, we believe that the data provides strong evidence that the simulation behaves accurately compared to the real world.

We have further corroborated this by comparing trajectory repeatability in simulation and the real world. Fig. 7 shows the concave hull around position information (robot's self-localization) recorded at 1 Hz during ten back and forth drive operations between machines M1 and M12. As we can see the overall trajectory execution is similar with only small differences at larger free areas where higher speeds can be achieved. Perception performance on simulated images for the camera perspective depends on the methods used as producing the bright spotlights on active lights is problematic. Example images are shown on the website.

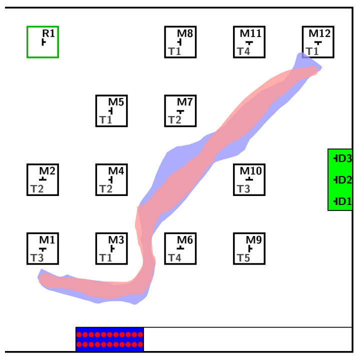

Fig. 7. Trajectory comparison of real world moves (blue) and simulated moves (red) on a half field.

Student Hackathon 2013 We conducted a hackathon with about 30 participants in cooperation with the student organization Bonding. The scenario was to recover prioritized color-coded items from an LLSF-like environment. Small student groups had to implement the mission behavior code using the Behavior Engine and the simulator. Then, the very same code was run on the real robot. Several teams accomplished the task during one night of development.

6 Conclusion

The Logistics League has unique aspects like environment agency, where the environment acts like an agent itself, and a focus on planning and scheduling in robotics. We have developed a simulation based on Gazebo which takes these into account. It connects to the referee box of the LLSF to provide the same reactions as the real environment. Multi-level abstraction allows to choose whether to process simulated sensor data like images or directly use information like signal light states. This way, the simulation can provide multiple levels of difficulty. This allows to run full integration tests or to focus on the development of the planning and scheduling system. As Gazebo is widely used and provides a messaging middleware, it is open for systems of other teams. We provide adapters for the Open Source robot software framework Fawkes, but others can be added easily.

Having to develop and maintain a team of robots can be prohibitively costly in terms of maintaining a local playing field. Other communities we want to reach might not even be interested in low-level robotics software. Therefore, we propose to establish a simulation league directly associated with the LLSF. It would serve as an entry to the LLSF, which is simplified by multi-level abstraction and the components we provide, and as a catalyst for new planning strategies, which could be developed and tested in simulation before being ported to the real robots. At RoboCup 2014, we seek a discussion with interested parties within and outside the LLSF to form an interest group to pursue this goal.

Our software components, instructions how to setup the simulation, all evaluation data, and further information is available on the project website at `http://www.fawkesrobotics.org/projects/llsf-sim/`.

Acknowledgments. We thank the Carologistics RoboCup Team for their tremendous effort to develop a system which served as a basis for the presented work.

F. Zwilling and T. Niemueller were supported by the German National Science Foundation (DFG) research unit *FOR 1513* on Hybrid Reasoning for Intelligent Systems (`http://www.hybrid-reasoning.org`). We thank the anonymous reviewers.

References

1. Niemueller, T., Ewert, D., Reuter, S., Ferrein, A., Jeschke, S., Lakemeyer, G.: RoboCup Logistics League Sponsored by Festo: A Competitive Factory Automation Testbed. In: RoboCup Symposium. (2013)
2. LLSF Technical Committee: RoboCup Logistic League sponsored by Festo – Rules and Regulations 2014. `http://www.robocup-logistics.org/rules` (2014)
3. Niemueller, T., Lakemeyer, G., Ferrein, A., Reuter, S., Ewert, D., Jeschke, S., Pensky, D., Karras, U.: Proposal for Advancements to the LLSF in 2014 and beyond. In: Proceedings of the International Conference on Advanced Robotics (ICAR) - 1st Workshop on Developments in RoboCup Leagues. (2013)
4. 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)
5. Boedecker, J., Asada, M.: SimSpark–Concepts and Application in the RoboCup 3D Soccer Simulation League. In: SIMPAR 2008: Workshop on The Universe of RoboCup Simulators. (2008)
6. Akin, H.L., Ito, N., Jacoff, A., Kleiner, A., Pellenz, J., Visser, A.: RoboCup Rescue Robot and Simulation Leagues. AI Magazine (2013)
7. van Noort, S., Visser, A.: Extending Virtual Robots towards RoboCup Soccer Simulation and @Home. In: RoboCup Symposium. (2012)
8. Carpin, S., Lewis, M., Wang, J., Balakirsky, S., Scrapper, C.: USARSim: A robot Simulator for Research and Education. In: IEEE International Conference on Robotics and Automation (ICRA). (2007)
9. Kootbally, Z., Balakirsky, S., Visser, A.: Enabling codesharing in Rescue Simulation with USARSim/ROS. In: RoboCup Symposium. (2013)
10. 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)
11. Niemueller, T., Ferrein, A., Beck, D., Lakemeyer, G.: Design Principles of the Component-Based Robot Software Framework Fawkes. In: Int. Conf. on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR). (2010)
12. Zwilling, F.: Simulation of the RoboCup Logistic League with Fawkes and Gazebo for Multi-Robot Coordination Evaluation. Bachelor's thesis, RWTH Aachen University, Knowledge-Based Systems Group (December 2013)
13. Koenig, N., Howard, A.: Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator. In: Int. Conf. on Intelligent Robots and Systems. (2004)
14. Niemueller, T., Ferrein, A., Lakemeyer, G.: A Lua-based Behavior Engine for Controlling the Humanoid Robot Nao. In: RoboCup Symposium. (2009)
15. Beck, D., Ferrein, A., Lakemeyer, G.: A Simulation Environment for Middle-Size Robots with Multi-level Abstraction. In: RoboCup Symposium XI. (2008)
16. Niemueller, T., Lakemeyer, G., Ferrein, A.: Incremental Task-level Reasoning in a Competitive Factory Automation Scenario. In: AAAI Spring Symposium 2013 - Designing Intelligent Robots: Reintegrating AI. (2013)