Team Description Paper robOTTO RoboCup@Work 2022

Christoph Steup, Hauke Petersen, Leander Bartsch, Inga Brockhage, David Hausmann, Niklas Harriehausen, Adrian Köring, Franziska Labitzke, Hanna Lichtenberg, Wiebke Outzen, Fabian Richardt, Jurek Rostalsky, Sanaz Mostaghim, Arndt Lüder, and Stephan Schmidt

Otto-von-Guericke University, 39106 Magdeburg, Germany
{`steup@ovgu.de`,
WWW: `http://www.robotto.ovgu.de/`

Abstract. Team **robOTTO** is the *RoboCup@Work League* team of the Otto-von-Guericke University Magdeburg, formerly participating in the *Robocup Logistics League* since its founding in 2010. In our team, we combine the expertise from Computer Science, Electrical and Mechanical Engineering to solve @Work's unique challenges while fostering knowledge exchange between the different leagues.

Keywords: RoboCup@Work, robOTTO, RoboCup2022

1 Introduction

robOTTO was founded as a *RoboCup Logistics*[5] team in 2010 by nine students from different fields, enabling exchange of knowledge and views between the members. After achieving second place 2010 in Singapore, the team continued to attend following RoboCup competitions with further successes in 2012 (4th place) and 2013 (2nd place). The transition from *RoboCup Logistics* to *@Work League* helped to broaden the expertise of the team and laid the foundation for the *Crossover Challenge*[8] at the *@Work League*, resulting in a first place at world cup in Leipzig 2016. In 2012, a first attempt at a second competition resulted in an 8th place in the *2D Soccer Simulation League*. With regard to broad rule and equipment changes in the *Logistics League* in 2015 we decided to participate in the *@Work League*[4], as the Computer Science Faculty already had some experience on the **KUKA youBots**[1], thus providing a pool of experienced students and easy integration into courses and research projects. The team competed in the world cup 2015 in China and reached 6th place followed by the world cup in Leipzig in 2016, where we scored 4th. 2017 was very successful for us, resulting in a third place at the GermanOpen and a second place at the world cup in Japan. In Montreal (Canada) the team achieved the first place in the Arbitrary Surface Challenge. In the last WorldCup in Sydney (Australia) the team achieved the Vice-World-Champion title again. In 2020, no real-world cup was done, and the team achieved a best-presentation award in the Virtual

RoboCup Asia Pacific Open (VRCAP). In the RoboCup@Home competition in 2021, which was held completely virtual, the team achieved the 3rd place. Additionally, the teams' effort in organizing the competition and the streaming was recognized by incorporating Franziska Labitzke and Hauke Petersen in the Organization Committee of the league, Leander Bartsch in the Technical Committee and Christoph Steup in the Executive Committee. Additionally, Christoph Steup is the current main maintainer of the new official Referee Box of the league.

2 Team Structure

Task	Name	Field / Role
Organisation	Sanaz Mostaghim	Responsible Professor
	Arndt Lüder	Liason to Mechanical Engineering
	Stephan Schmidt	Liason to Mechanical Engineering
	Christoph Steup	Team Leader (EC-Member)
	Hauke Petersen	Co-Team Leader (OC-Member)
	Franziska Labitzke	Public Relations Officer (OC-Member)
Navigation	Niklas Harriehausen	Mechanical Engineering
Hardware	Leander Bartsch	Electrical Engineering (TC-Member)
	Inga Brockhage	Electrical Engineering
Recognition	Adrian Köring	Computer Vision
RobotCoordination	Wiebke Outzen	Computer Science
	Fabian Richardt	Computer Science
	Jurek Rostalsky	Mathematics
	Hanna Lichtenberg	Computer Science
	David Hausmann	Computer Science

Table 1. Overview of **robOTTO** team members by task and field of studies or role

Currently, the team consists of 12 active members and 8 new members currently in training, whereas the professors are not involved in the development and only provide guidance and organizational support. The team is composed of students, who will leave the team after finishing their studies, as well as postgraduates, which allows the team to combine fresh ideas and approaches, with the experience of the veterans. Depending on the new members and their backgrounds, the team can largely benefits from a diverse set of expertise. The current team members provide a large spectrum of topics from cybernetics and computer science to electrical engineering to the team as shown in Table 1. The new students provide new ideas and also new challenges to the team. Currently, the new members try to ease the use of the robot in the competition to minimize human errors and improve the efficiency of the setup process for new arenas to lessen the time till the first training run.

3 Robot Description

The standard **KUKA youBot** does not provide any sensory equipment. Modification were necessary to use the robot in the @Work league. To minimize effort and maximize results most of the additions are COTS, used in the robotics community. Our modification relates to the sensory equipment consisting of an additional camera and two laser scanners and to the manipulation system extended with a specialized gripper. Additionally, we switched to a more powerful PC.

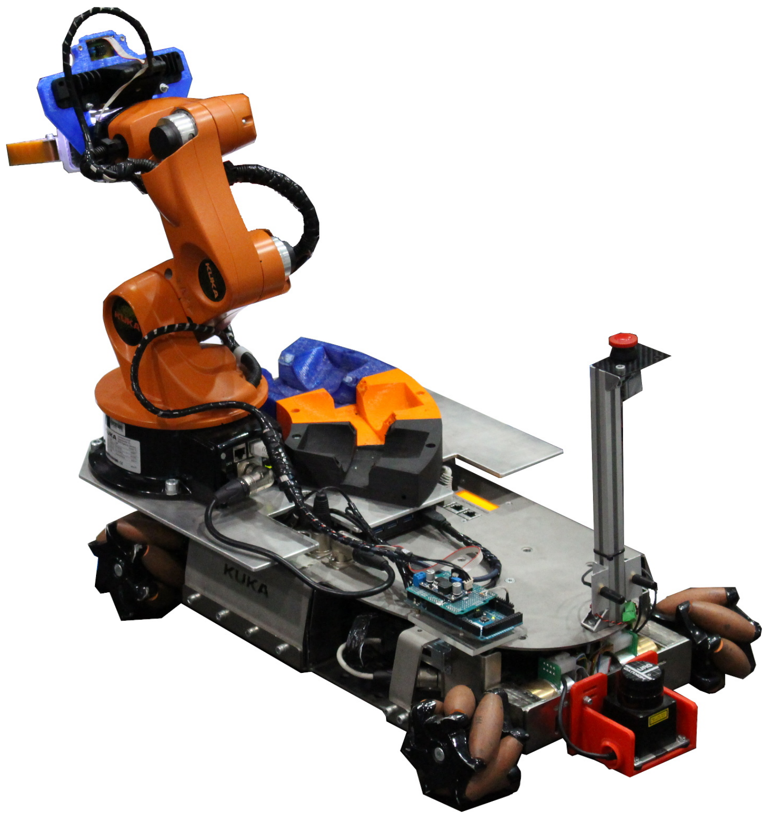

Fig. 1. Modified KUKA youBot

3.1 Changes to the standard Platform

Camera We use an **Intel RealSense** RGB-D camera, which provides registered point clouds as well as an RGB-D image. Currently, we focus on the color data for recognition and use the depth image as a highly flexible distance sensor. Around the lenses and projectors of the **RealSense** we mounted an oval-shaped ring of LEDs to improve lighting conditions and enable a reliable object detection and classification. During navigation, the **RealSense** camera is used to detect *barrier tape.*

Gripper The objects of the @Work league have varying shapes and sizes. After preliminary tests, we observed that the normal metal gripper on the **KUKA youBot** cannot reliably handle many of these objects. Our current gripper is based on a custom 3D-Printed mount using a single servo by **Dynamixel** and **Finray**-fingers by **Festo**, which are controlled by an embedded board. Current development focus on correctly identifying and handling error cases like the loss of an object. Additionally, we aim to improve the grasping of not perfectly aligned objects.

Computing and Connections We updated our Intel NUC to a newer 8th generation version providing 4 real cores with 45W TDP to enable more complex algorithms for navigation, path planning and task optimization. We now use two Teensy ARM-Cortex M4 powered embedded boards to integrate the gripper and our custom power supply. The hardware and software architecture is modularized through defined interfaces to provide a stepping stone for students inexperienced with programming to experiment without needing the whole development stack used on the main robot.

Laser Scanner Mounting Brackets The team uses **Hokuyo URG-04LX** laser scanners. These provide appropriate distance measurements in a 240° radius with a maximum distance of 5.5 m. A reliable localization is possible if the laser scanners are exactly parallel to the ground. To this end, the team designed reliable, adjustable mounting brackets, which were 3D printed by the team to prevent tilt errors even at the edge of the scanner's measurement range and shield the expensive sensors in the case of accidental collisions.

Replacement of the Upper Case The newly designed and manufactured upper case completely replaces the old top cover of the YouBot. It provides the option to access the underlying electronics easy, while extending the space inside the robot. This enables the possibility of another **Intel Core i7 NUC**, the power supply of the grippers and our **USB3** hub to be stored inside the robot, see 3.1. Furthermore, the inventory, emergency switch and arm have pre-build mountings on the new case. Thanks to multiple mounting holes, the inventory can be offset, and additional parts can be added easy to the surface. The new cover proved very successful in the competition of 2019 and 2021.

Lithium Polymer-based Battery System The original lead-based batteries of the YouBot have two major drawbacks. Firstly, they are incredibly heavy, which increases the transportation cost when flying. Secondly, their contained energy is rather scarce compared to the power necessary for all the components. Currently, our bot lasts for max 30 minutes, which is quite limited. Because of these reasons, we decided to switch to Lithium-Polymer-based batteries. But instead of using single cells, we use Bosch tool batteries, which are widely available and raise no suspicion at the airport. To this end, we integrated two of-the shelf buck-boost DC-DC converters for 12V and 24V as well as a customly designed PCB to hold voltage and current measurement, as well as emergency, switches to cut the different power rails. The current setup uses two batteries in parallel, which allows us to Hot-Swap the batteries without powering the bot down and up. Because of the ongoing Corona-Pandemic, we could not test the new system in a real environment, but our observation suggest a significant increase in endurance.

4 Software Architecture

In this section, we want to describe the main software components and how they interact. We use the **Robot Operating System (ROS)**[6] in the *Melodic* version running on **Ubuntu 18.04 LTS**. The main advantages are the communication abstraction and the great number of easy-to-use debug tools.

4.1 Overview

Fig. 2 shows the interaction of our software components. They are described in detail in the following sections.

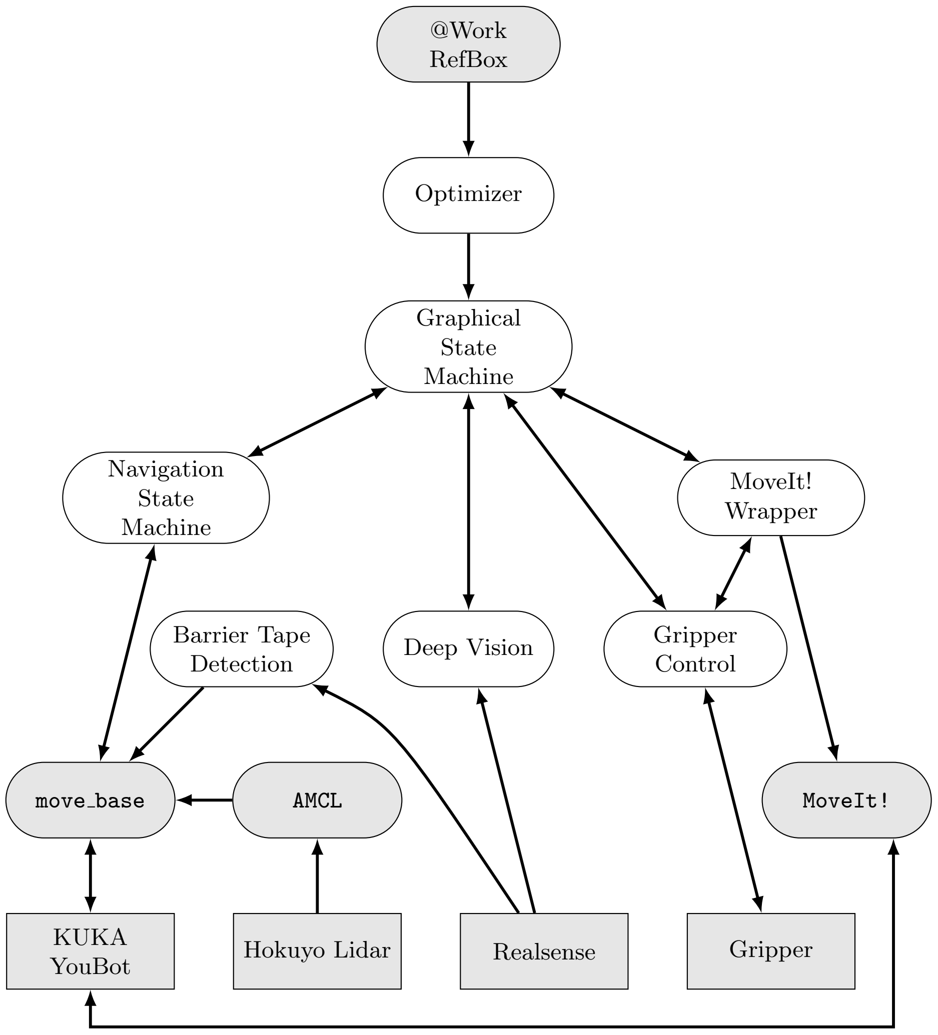

Fig. 2. Overview of the main software components of the robOTTO @Work framework.

State-Machine and Optimizer Core elements of the robot software architecture are the optimizer and the state-machine, which are responsible for coordinating the other modules of the robot like vision, navigation and arm movement. The transportation tasks, which the robot has to fulfil, are generated by the @work referee box of the league and transferred to the robot. On the robot side, the *receiver_node* is used for processing and forwarding those messages to the optimizer. The optimizer searches for a sequence of transport tasks which reaches the maximum score within the time limit. Subsequently, it generates *SubTasks* as an input for the state machine. Typical *SubTasks* are picking up an object, placing an object or moving the robot to a workstation.

The state machine contains a logic for every *SubTask*. The logic determines a graph of parametrized actions which are necessary to perform the *SubTask*. Common actions are the *MoveAction*, *ArmAction* and *VisionAction*. They control the sub-state machines of the corresponding robot modules. For instance, a simplified *PickLogic* consists of following actions:

1. **ArmAction** moves the arm to the pose for barrier tape recognition.
2. **MoveAction** moves the robot to a specified workstation.
3. **ArmAction** moves the arm to the pose for object recognition.
4. **DetectAction** looks for the specified object on the workstation.
5. **ArmAction** picks up the object which was localized by the vision and place it in the inventory.

Because a reliable state machine is critical for the success of a robot performance, it is one of the most intensively unit tested modules of the robot.

World Model The world cup 2017 in Leipzig showed that the complexity of the tasks and the environment is difficult to handle in our current software architecture. To incorporate additional information on the state of the arena or the robot, lots of changes to code and interfaces were necessary. To mitigate this engineering issue, we decided to manage the information on the world in a central component. The resulting world model component allows us to store, track and replay changes to the robots and the arena's state. Additionally, we added support to add and modify the data in the world model of a specific task and visualize it. The major benefit of this approach is the stability of interfaces between our functional *SubTask* components, as well as the centralized point for team members to add and request information on the world. Finally, the visualization tool gives us better insights into the robot's current behavior and choices to ease debugging.

Deep-Learning-based Vision - Deep Vision We currently only use 2D images for object detection, even though the RealSense also provides 3D data. Our object detection system is based on the TensorFlow Object Detection API. We created a custom bootstrapping and augmentation system to generate additional training data. The major work in this system was the manual generation of training data. To ease the process, we developed a custom tool to check the detection and classification output and generate manually corrected training samples. To only addition to the original detection API is an additional object orientation detection code to also output a 2D rotation of the object on the workstation.

Path Planning for Industrial Robots The navigation of industrial robots has to be developed with multiple competing influences in sight, as fast movement and collision avoidance are both critically important to successfully participate in *RoboCup@Work*. Other factors are more subtle, like predictability of behavior and easy maintenance and adaptability. The last two points are especially important in the context of RoboCup as a competition of students, where team members and responsibilities switch regularly and members have to be able to familiarize them self with the code, often on a short notice.

The current navigation stack used by robOTTO supports two different approaches. The first approach uses the **DWA-Planner (Dynamic Window Approach)**[3] which is open-source code and the standard planner for holonomic platforms in **ROS**. Since, it is quite complex with many configurable parameters, we use a slightly modified parameter-set which was made available publicly by the **b-it-bots** team for usage with the **KUKA youBot**.

The second approach is a minimal implementation of a local planner which relies on the global planner for object avoidance to keep the complexity and feature duplication down. It was developed after preliminary tests with the **DWA**-Planner showed unpredictable and often oscillating behavior depending on a multiple factors like CPU load and floor conditions.

Integration of the MoveIt! Trajectory and Kinematics Stack The previously used kinematics stack for the manipulator used by the team, **SAMIA**, was built by former member Stefan as a by-product of his Master's Thesis and subsequently adapted for the @Work competition. But with Stefan gone, we now face problems maintaining and extending the codebase. This led to the decision to abandon our own stack in favor of **MoveIt!** [2]. **MoveIt!** is an open-source motion planning framework originally developed by Willow Garage, which unifies motion planning kinematics, collision checking and dynamic three-dimensional environment representations. As it was initially developed to be used with **ROS**, it offers a high degree of integration with existing packages and tools, such as **RViz**. The stack's incorporation into our code, as well as the creation of the arm-state-machine Figure 2 and interfacing with our main state machine, was done by Hauke. Since then, it proved to be a viable alternative to our previous solution and had successfully been used at GermanOpen 2016 in Magdeburg and RoboCup 2016 in Leipzig. One of the stack's issues on our platform was the calculation time required to generate a valid movement trajectory. For this purpose, we have developed a **ROS** package which allows caching static trajectories (e.g. when placing an object in one of the inventory slots) and integrate them in dynamic trajectory paths to speed up the calculations. Currently, we are working on the integration of MoveIt's Planning Scene to allow complex motion planning in environments with obstacles. We want to use this approach for ”tight” spaces like shelves or workstations with walls around it.

Barrier Tape Detection The Barrier Tape Detection is asked to spatially locate barrier tape strips given a camera image and position. To achieve this, we first detect the barrier tape in the image. This is done using a semantic segmentation of the camera image also used for object detection and recognition. However, in this case a convolutional neural network is used, which combines the task of filtering, segmentation and recognition. We used **Tensorflow** to implement the network and train it. The output of the network is an image of white pixels, where the barrier tape was detected. The training samples are generated using images of the used barrier tapes in the competition overlaid on various background images and lighted by **Blender**. To get the world coordinates of the barrier tape, we cast rays from the camera through the white pixels in the mask into the scene and find intersections with the ground plane. All such intersection points finally form the resulting point cloud, which is fed to the navigation stack as an additional sensor. The only manually configured part of the *Barrier Detection* is the calibration of the mapping of the 2D camera coordinates to the 2D map coordinates of the navigation.

Graph-based Visual State-Machine We finished the integration of FlexBE [7] as our new state-machine description mechanism. All our old state-machine code as well as the previously separated RTT-State-Machine are now encoded in the new system. As a byproduct, we simplified the state-machine for the manipulator and integrated extended planning capabilities. The new state-machine still needs to be evaluated in realistic contexts of competitions to see if it fulfills the expectations of more flexible and faster adaptations on site.

5 Future Software Components

The following components are currently worked on and may be used in the WorldCup 2022 if the development is finished, and they are sufficiently tested.

5.1 New Camera and Vision Pose

We bought the new and probably last Intel RealSense L515 and are currently aiming to integrate it in our manipulator. However, the new vision concept is fundamentally different by using a different vision pose as well as a different mount point for the camera, allowing more efficient vision operations. Additionally, the new system shall allow us to handle shelves and high tables.

5.2 Specialized Near-Field Localization

We are currently working on a specialized near-field localization mechanism to enable precise localization of the robot close to workstations to remove our specialized navigation state-machine, which moves manually when very close to workstations. The new approach will enhance AMCL using semantic localization of tables and fuse the resulting pose information using an Unscented Kalman Filter.

5.3 New Optimizers

To speed up and enhance the quality of the resulting behavior of the robot, we develop two new optimizers using a Branch-and-Bound and a Genetic-Algorithm-based approach. These optimizers shall allow us to handle dynamic failures of sub-tasks within the run and provide, in general, better and faster results than the old one.

6 Conclusion

With the influx of new team members and the continued participation by last year's members, we are cautiously optimistic that we will be able to build upon the work done last year. Expected changes to the rules of the competition mandate some overhauls of components like manipulation movement and tables height estimation. The last participation left us with a code base, solving most of the tasks of the @work league. This enables us to focus this year on testing and improving the robustness of the working solutions, while adapting them to the new rules.

Acknowledgements

We would like to thank Sanaz Mostaghim, Chair of Computational Intelligence for her support during the year and providing us with access to tools and the necessary facilities for storage and testing. The team and all former members would also like to thank Arndt Lüder for his engagement and support since our founding in 2010.

References

1. Rainer Bischoff, Ulrich Huggenberger, and Erwin Prassler. Kuka youbot-a mobile manipulator for research and education. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.
2. Sachin Chitta. Moveit!: an introduction. In *Robot Operating System (ROS)*, pages 3–27. Springer, 2016.
3. Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.
4. Gerhard K Kraetzschmar, Nico Hochgeschwender, Walter Nowak, Frederik Hegger, Sven Schneider, Rhama Dwiputra, Jakob Berghofer, and Rainer Bischoff. Robocup@ work: competing for the factory of the future. In *Robot Soccer World Cup*, pages 171–182. Springer, 2014.

5. Tim Niemueller, Daniel Ewert, Sebastian Reuter, Alexander Ferrein, Sabina Jeschke, and Gerhard Lakemeyer. Robocup logistics league sponsored by festo: a competitive factory automation testbed. In *Automation, Communication and Cybernetics in Science and Engineering 2015/2016*, pages 605–618. Springer, 2016.
6. Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3.2, pages 1–6. Kobe, 2009.
7. Philipp Schillinger. An approach for runtime-modifiable behavior control of humanoid rescue robots. Master's thesis, Technical University Darmstadt, 2015.
8. Sebastian Zug, Tim Niemueller, Nico Hochgeschwender, Kai Seidensticker, Martin Seidel, Tim Friedrich, Tobias Neumann, Ulrich Karras, Gerhard Kraetzschmar, and Alexander Ferrein. An integration challenge to bridge the gap among industry-inspired robocup leagues. In *RoboCup Symposium*, pages 1–12, 2016.