Team Description Paper robOTTO RoboCup@Work 2017

Christoph Steup, Martin Seidel, Jonathan Beckhaus, Philipp Busse, Alexandra Grube, Christina Grüll, Nils Harder, Lukas Hoyer, Johannes Kopton, Adrian Köring, Rodion Marynych, Hauke Petersen, Kai Rüssel, Sanaz Mostaghim, Sebastian Zug, Arndt Lüder, and Stephan Schmidt

Otto-von-Guericke University, 39106 Magdeburg, Germany
{steup,maseidel}@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, RoboCup2017

1 Introduction

robOTTO was founded as a *RoboCup Logistics*[4] 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 change from *RoboCup Logistics* to *@Work League* helped to broaden the expertise of the team and in fact eased the *Crossover Challenge*[7] 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 regards to broad rule and equipment changes in the *Logistics League* in 2015 we decided to participate in the *@Work League*[3], 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.

2 Team Structure

Currently, the team consists of 13 active members, whereas the professors are not involved in the development and only provide guidance and organizational

Task	Name	Field / Role
Organisation	Sanaz Mostaghim	Responsible Professor
	Sebastian Zug	TC Member
	Arndt Lüder	Liason to Mechanical Engineering
	Stephan Schmidt	
	Christoph Steup	Team Leader
	Martin Seidel	Co-Team Leader
Navigation	Philipp Busse	Computer Science / OC Member
Hardware	Nils Harder	Electrical Engineering
	Kai Rüssel	Mechatronics
Manipulation	Rodion Marynych	Electrical Engineering
	Hauke Petersen	Mechatronics
Recognition	Johannes Kopton	Cybernetics
	Adrian Köring	Computer Vision
RobotCoordination	Jonathan Beckhaus	Computer Science
	Alexandra Grube	Business Informatics
	Christina Grüll	Medical Engineering
	Lukas Hoyer	Computer Science

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

support. Since the team is mainly composed of students, who leave the team after finishing their studies, we have a young and dynamic team of students. 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 business informatics to medical 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, as they found the work-flow quite difficult.

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 relate 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.

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 colour data for recognition and use the depth image as a highly flexible distance sensor.

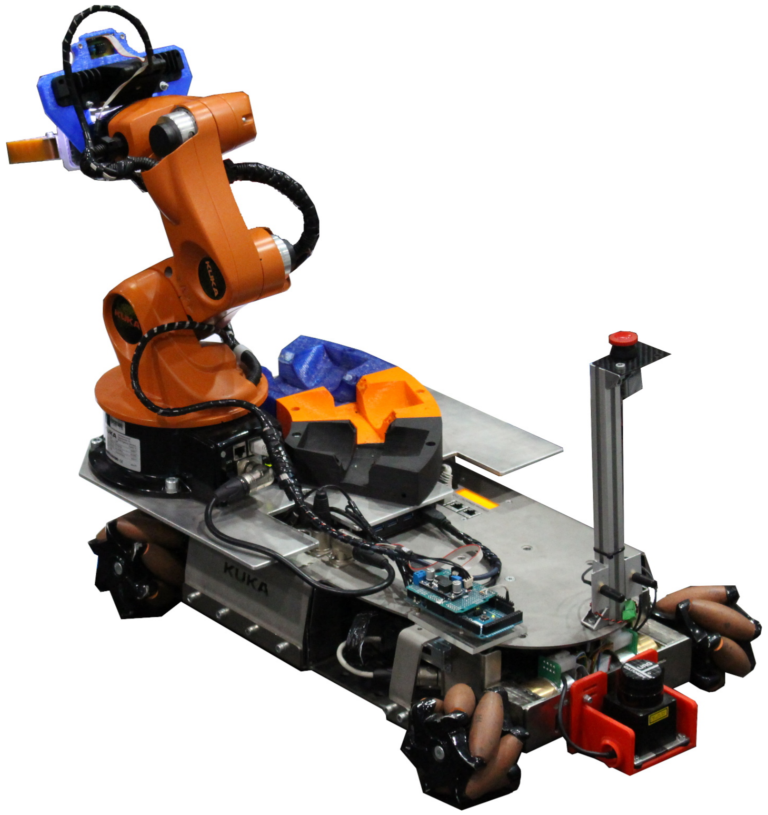

Fig. 1. Modified KUKA youBot

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 an custom 3D-Printed mount using servos by **Dynamixel** and **Finray**-fingers by **Festo**, which are controlled by an **Arduino**. 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 An **Intel Core i7 NUC** was fitted into the robot as a replacement for the original Atom-based computer to enable more complex algorithms for navigation, path planning and task optimization. Additionally, an **Arduino** was fitted to the chassis to integrate various embedded components like servos, time-of-flight distance sensors and LEDs. 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.

3.2 Future Modifications

Batteries and Power Supply We originally planned to replace the batteries and power-supply completely for a lithium-ion based system, but the experiences gathered while transporting batteries by plane for last years' world cup in Hefei put a stop to that plan as lithium batteries in the required size are not allowed to be transported on passenger planes without special permits. Our alternative solution for the mid-term will be to replace the internal charging circuit to enable full capacity charging and prevent deep discharge of one battery cell which seems to be caused by the control-panels power draw even if the robot is turned "off".

Replacement of the Upper Case For the time being, the upper case of the **youBot** is a bended steel plate, that's connected to the bottom panel. On the top there are recesses for the display, various ports (USB, LAN, power connection) and threaded holes to mount the Arm. Beneath the steel case is the electronics.

We have extended the upper case with an aluminium plate. It provides more place for an inventory and gives the opportunity to offset the arm to the side, see figure 1.

In the future the upper case will be completely replaced by a new construction consisting of several parts. It will provide the option to access the underlying electronics easy, while extending the space inside the robot. This enables another **Intel Core i7 NUC** and the **Arduino** to be stored inside the robot, see 3.1. Furthermore, the inventory, emergency switch and arm will 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.

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)**[5] in the *Kinetic* version running on **Ubuntu 16.04**. 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.

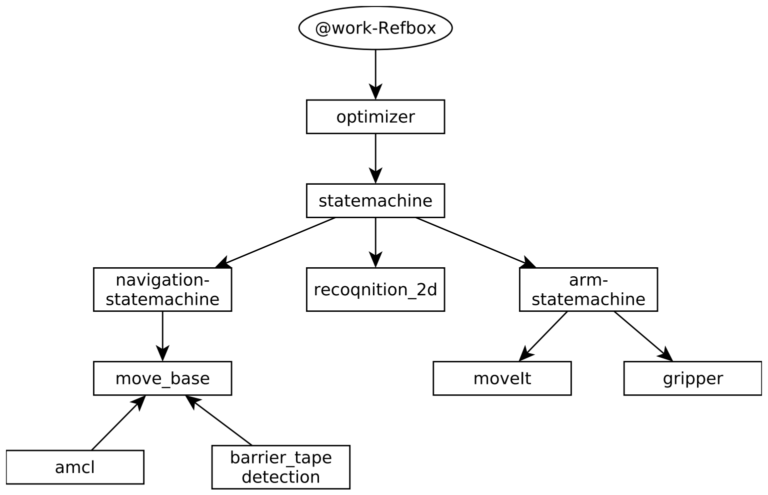

Fig. 2. Overview of the main software components of the robOTTO @Work framework.

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 parametrised 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 allow us to store, track and replay changes to the robots and the arenas state. Additionally, we add 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 robots current behaviour and choices to ease debugging.

Self learning Object Recognition - recognition_2d Last year we switched from 3D point cloud recognition to a 2D image approach. Therefore, we implemented an image processing based segmentation algorithm. For classification the images are passed through a self-implemented *artificial neural net (ANN)*. The *ANN* takes the preprocessed RGB image (or perhaps depth map in the future) as its input and provides a probability value for each object as output. The training of the *ANN* uses the recorded and annotated example image streams as input and outputs the parameters of the *ANN*. The object features are learned automatically during the training of the neural net. The orientation of the objects is determined using linear approximation. Our current goal is to improve the recognition results under poor lighting conditions and with unfavourable backgrounds.

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 behaviour 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 familiarise 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)**[2] 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 behaviour depending on a multiple factors like CPU load and floor conditions.

Furthermore, our team member Hauke, is developing a global path planner in his master thesis. His planner is based on the **Open Motion Planning Library (OMPL)**[6]. The **OMPL** provides a set of tree- and road-map-based planners that are easily interchangeable. Besides the typically positions, this global planner will compute the orientations of the robot along the path.

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 favour of **MoveIt!**[1]. **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 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 a feature allowing quick movements within a small area without time-costly kinematic calculations, an option much needed for the upcoming implementation of the Precision Placement Task.

4.2 Future Software Components

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 by segmentation, separating the pixels showing the barrier tape from everything else and saving the result in a binary mask. Using the mask and the camera pose

[1] http://moveit.ros.org/

- together with the assumption of a planar floor - we then calculate a 3D point cloud that, as a whole, indicates the position and extend of the barrier tape in the arena.

The image segmentation is achieved using a convolutional neural network. The networks' input is the camera image and the corresponding output will be a mask containing the segmentation. Training of the network is achieved using a data set of already segmented images. Those images are either generated or recorded and labelled in test runs, when the barrier tape position is known beforehand.

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.

Precision Placement SubTask To start the *PPT-SubTask* the Robot picks up the object and moves to the PPT-Table. The Vision recognizes the specified hole. Once the robot is situated in front of the right opening the arm picks up the object out of the inventory and moves directly above the hole the object needs to be placed in.

To cope with the movement errors of the arm the PPT controller moves the object slightly in all possible directions, while the visions compares object and opening and feeds back the information to the PPT controller. The controller calculates the best position, which is needed to successfully place the object in the hole. After the object is at the optimal position above the opening, the PPT controller tells the arm to lower it closer to the opening. During the lowering process the vision checks the alignment of object and opening to ensures a successful process. As soon as the object is close enough above the opening the gripper drops the object straight into the hole.

5 Conclusion

With the influx of new team members and the continued participation by last years members we are cautiously optimistic that we will be able to build upon the work done last year while cleaning up the artefacts left by frantic development during the world cup. 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. The additional members in our team allow us to develop the non-existent functionalities in parallel.

Acknowledgements

We would like to thank Sebastian Zug, chair of the work group for Embedded Smart Systems and Sanaz Mostaghim, chair of Intelligent Systems Group for their 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. Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.
3. 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.
4. 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.
5. 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, pages 1–6. Kobe, 2009.
6. Ioan A Sucan, Mark Moll, and Lydia E Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.
7. 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.