Team Description Paper - SWOT RoboCup@Work 2021

D. Blümm, L. Kraus, F. Mast, F. Spieß, L. Reinhart, and T. Kaupp

University of Applied Sciences Würzburg-Schweinfurt, 97421 Schweinfurt, Germany

https://robotik.fhws.de/fhws-robotik/forschung/projekte/robocupatwork/
robocupatwork@fhws.de

Abstract. Team **SWOT** is the RoboCup@Work League team of the University of Applied Sciences Würzburg-Schweinfurt. Our team consists of engineers from the fields of electrical and mechanical engineering as well as mechatronics.

Keywords: RoboCup@Work · SWOT · RoboCup2021.

1 Introduction

SWOT was founded by Prof. Dr. Tobias Kaupp and Prof. Dr. Norbert Strobel alongside one employee and four students in 2020. Our team is part of the University of Applied Sciences Würzburg-Schweinfurt. The development of the system started about one year ago and since that the team put a lot of time and effort to reach the current state. Aside from both professors there are now two employees and four students participating. The professors and employees are mainly supporting by organizing and guiding the students. The main development is done by four master of electrical engineering students.

Table 1. Overview of SWOT members by task and field of studies or role

Task	**Name**	**Field/Role**
Organisation	Tobias Kaupp	**Responsible Professor**
	Lucas Reinhart	**Team Leader**
	Florian Spiess	**Co-Team Leader**
Navigation	Fabio Mast	**Computer Science**
Hardware	Daniel Blümm	**Electrical Engineering and Additive Manufacturing**
	Max Dobmann	**Electrical and Mechanical Engineering**
	Martin Löser	**Mechanical Engineering**
Manipulation	Lukas Kraus	**Robotics**
Recognition	Lukas Kraus	**Computer Vision**
Robot Coordination	Fabio Mast	**Computer Science**
	Daniel Blümm	**Computer Science and System Integration**

2 Robot Description

The robot consists of five main components:

- mobile robot platform from Evocortex GmbH
- UR3e robot arm from Universal robots
- Hand-e gripper from Robotiq
- Intel RealSense D435i
- Intel UP Xtreme

These components were purchased individually and form a valid base while enabling easy integration of additional parts. In the following, we will take a closer look at the different components and their technical specifications.

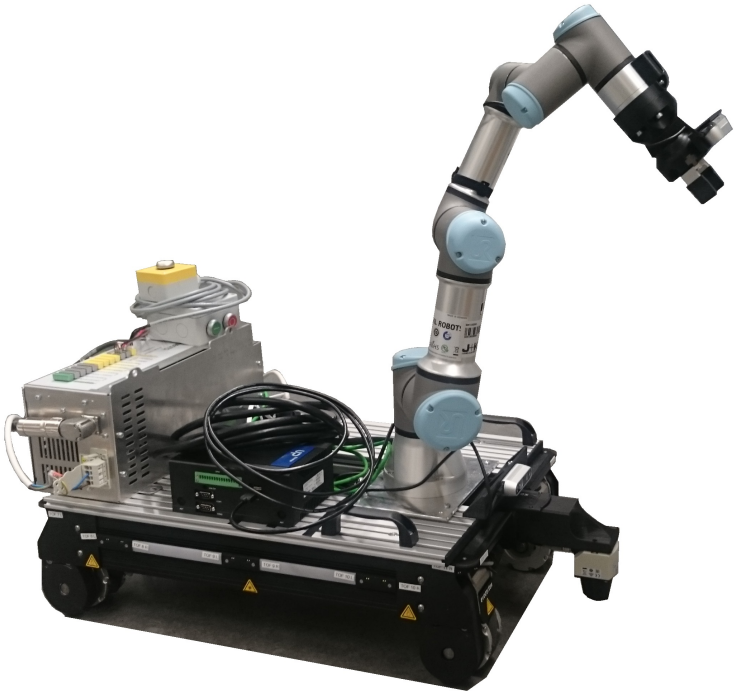

Fig. 1. The current robot with the Evocortex as base, the UR3e, and the Hand-e

Evocortex: This platform is a base to tackle the @Work leagues' tasks. It already comes with a bunch of sensors and additional useful functions. A premounted Sick TIM561 LiDAR-sensor in the front of it enables localization inside the environment. Due to its Mecanum wheels, the platform can move omnidirectionally on the ground, which enables it to maneuver even in narrow and cramped areas. It also has a total of thirteen Time of Flight (ToF)-sensors, eleven of them mounted all around the Evocortex to detect obstacles like walls and avoid collisions. The other two ToF-Sensors are mounted on the bottom to recognize holes. Additionally, there is a camera mounted facing the ground. A removable plate covers the Evocortex' internals and allows easy exchange, as well as the addition of components and sensors.[1]

UR3e and Hand-e: Universal Robot's UR3e robot arm and Robotiq's Hand-e manipulator form a system that enables gripping, lifting, and moving parts weighing up to three kilograms. With this setup, it can reach parts at a distance of 50 centimeters away, which is more than sufficient for the grasping tasks in @Work league.[2][3]

Intel RealSense D435i: This camera allows depth recognition and provides registered point clouds as well as RGB-D image data. With this data, we attempt to recognize objects by comparing it with the CAD files provided by the @Work league.[4]

Intel UP Xtreme: To strictly separate mobile platform control from other computational tasks we added the Intel UP Xtreme to the robot. The Jetson TX2 mounted inside the Evocortex runs a lot of resource-expensive software. To expand the computational resources, we decided to add the UP Xterme to the system.[5]

Planed future changes: Currently, we are planning to add another Sick LiDAR sensor. With our setup, it's not possible to reliably detect obstacles on the left, right, or backside of the robot. Thereby moving backward is difficult as only the ToF-Sensors can deliver data to avoid colliding with obstacles.

3 Software Architecture

The upcoming sections describe the different parts of the software. Our system runs on **Ubuntu 18.04** with the Robot Operating System (ROS) version Melodic. In the following a brief overview of the different software components is given. After that a more detailed descriptions on how all components work together to accomplish the tasks of RoboCup@Work league is provided.

3.1 Overview

Figure 2 shows the current software architecture as well as the interactions between the different parts.

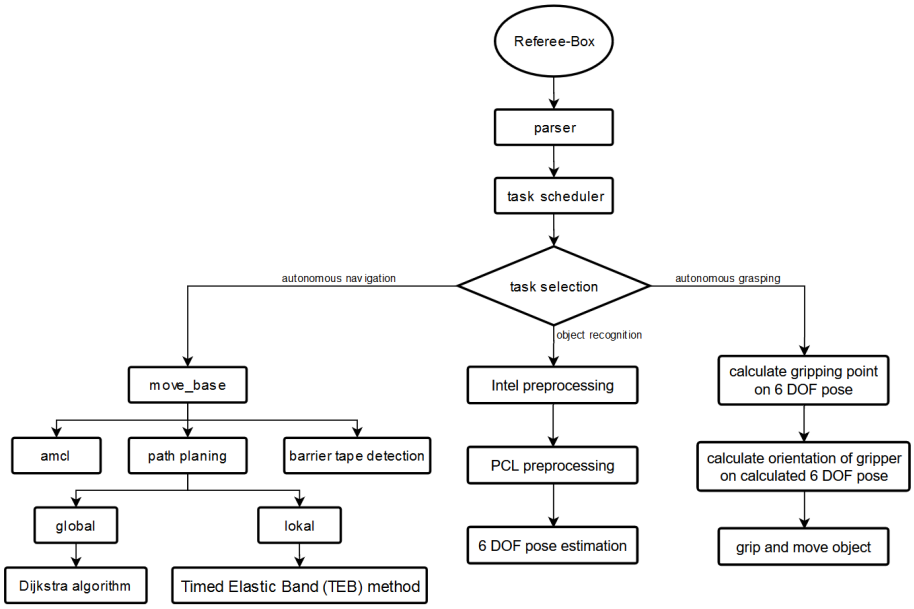

Fig. 2. Simplified representation of the main software components and their interactions

At the moment the software is a combination of four components:

1. Task scheduler
2. Autonomous navigation
3. Object recognition and pose estimation
4. Manipulation and control

The following sections describe these different parts and how they interact.

3.2 Task scheduler

The task scheduler is responsible for analyzing, sorting, and supervising the different tasks transmited by the Robocup's referee box. A simple parser converts the data of the message into a more sophisticated format. Afterwards the scheduler calculates from the given list of objectives the required movement, object recognition, and grasping tasks to achieve them. The tasks are then sorted to guarantee the most efficient execution. An example is a movement task followed by an object recognition task. In this case, the camera on the robot arm can already be moved to the correct position for object analysis. Meanwhile, the robot executes the last sub-task of the movement task e.g. orientating itself correctly in front of the workstation. Aside from analyzing and sorting tasks the task scheduler also starts their execution and handles unexpected situations, e.g. errors during gripping. Dealing with unexpected situations interrupts task execution and requires re-planning a certain part of one or more tasks. This ensures the robot is not damaging itself nor its surrounding.

3.3 Navigation

Before starting the navigation we create a map of the environment using gmapping [6] package. The ROS map_server[1] package is used. To navigate well, the robot needs to get information about the environment and its pose within the environment. It is achieved by sensors which capture data of the surroundings (camera, LiDAR) and internal states (odometry and imu). Our navigation is built around the move_base[2] package, which provides interfaces to other packages, e.g. those for sensor data interpretation, motion command computation and execution.

For the robot localization, we utilize the LiDAR sensor in combination with the ROS **amcl**[3] package. The amcl package implements the Adaptive Monte-Carlo-Algorithm, which is a type of particle filter [7]. It uses the prerecorded map to estimate the robot's current position.

The motion planning of the mobile robot is divided into global and local path planning based on the prerecorded map. For the global path, we use the global_planner package, which we configured to use the Dijkstra algorithm [8]. This algorithm returns the shortest path between the start and the goal pose [9].

For local path planning, the teb_local_planner[4] is applied. It uses the Timed Elastic Band (TEB) method which is based on the Elastic Band Method. It first calculates a deformable, collision-free path from the global path. Then under artificial forces (internal forces), a short smooth path is created from that in real-time. By interpreting the captured sensor data (external forces), it can be deformed to keep a sufficient distance from obstacles. Time intervals are calculated

[1] `http://wiki.ros.org/map_server`
[2] `http://wiki.ros.org/move_base`
[3] `http://wiki.ros.org/amcl`
[4] `http://wiki.ros.org/teb_local_planner`

afterward indicating how long the robot needs to move from one configuration to the next. In relation to these two configurations and the time intervals, the TEB attempts to perform a weighted optimization in real-time [10].

3.4 Object recognition and object pose estimation

The Point Cloud Library (PCL)[11] is used for object recognition and object pose estimation based on the CAD data of the object. First, the workstation is scanned with the camera from multiple sides. As a second step, the depth images are merged. A point cloud is generated which is preprocessed by several filters including pass through, radius outlier removal, and uniform sampling. Then the background is removed by plane segmentation. The remaining points are used for object detection and determining the pose.

As a next step keypoints and surface normals are calculated both on the CAD model and in the point cloud. Around the keypoints, the surface changes are stored in histograms using a local descriptor. The descriptors of the scene and the model are then compared and a list of correspondences is generated. Finally, using Hough 3D[5] from PCL, the pose of the object is calculated. The pose is further refined using Iterative Closest Point (ICP).

3.5 Manipulation and Control

The gripping point is determined using an object database and the calculated pose position. In the database for each orientation of the object, a grasping point is defined approximately in the center of the object. After transforming the coordinates of the grasping pose into the base coordinate system, the calculated pose is transmitted to the MoveIt package which controls the robot arm. MoveIt[6] is a widely used ROS package enabling motion planning, manipulation of objects, and control. MoveIt then plans and executes a trajectory to grasp the object. Currently, the trajectory planning needs to be improved to make the system robust against collisions.

4 Applicability, reusability and relevance to industrial tasks

The applications developed for @Work league are tackling a wide range of tasks in the industry. Therefore, the main goal is the development of a modular system. All components are supposed to be portable to other robots and tasks. The main reason for this attempt is the fact that @Work league tasks can be adapted to solve a variety of problems not only in industry but also in the medical or private areas. Since our university just established a new B. Eng. Robotics course the applications developed as part of this project will be used for educational purposes as well.

[5] `https://pointclouds.org/documentation/hough__3d_8h_source.html`
[6] `https://moveit.ros.org/`

5 Conclusion

This paper described the current state of development for the participation in RoboCup@Work league. The information shows how the different tasks, e.g. navigation, perception, manipulation, and control are tackled with our current setup. Furthermore, it contains information regarding future changes of hardware and software.

Acknowledgements

Our special thanks go to the University of Applied Sciences Würzburg-Schweinfurt for providing the infrastructure and all the necessary components to build the system. We are also grateful to the Hans-Wilhelm Renkhoff foundation [12] for the financial support, to establish the core team.

References

[1] S. M. Daniel Ammon Hubert Bauer, "Betriebsanleitung evocortex f& e plattform," in. Evocortex GmbH, Nov. 2019, pp. 13–16, 28, 30–34.

[2] U. Robots, "E-series von universal robots," in. Universal Robots GmbH, 2019, pp. 17, 22, 23.

[3] Robotiq, *Hand-e adaptiver robotergreifer*, `https://robotiq.com/de/produkte/adaptiver-2-finger-robotergreifer-hand-e`, Zugriff am 28.02.2021.

[4] Intel, *Intel realsense depth camera d435i*, `https://www.intelrealsense.com/depth-camera-d435i/`, accessed 16.03.2021, 2021.

[5] ——, *Up xtreme edge compute enabling kit*, `https://up-board.org/up-xtreme-edge-compute-enabling-kit/`, accessed 03.03.2021, 2021.

[6] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *ICRA*, 2005, pp. 2443–2448.

[7] D. Fox, "Adaptive particle filters," *Neural Information Processing Systems*, vol. 14, pp. 713–720, 2001.

[8] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[9] ROS, *Ros wiki*, `http://wiki.ros.org/global_planner`, accessed on 22.03.2021.

[10] F. H. Martin Keller, "Planning of optimal collision avoidance trajectories with timed elastic bands," *IFAC Proceedings Volumes*, vol. 47, pp. 9822–9827, 2014.

[11] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[12] *Warema Renkhoff SE*, Jan. 26, 2020. [Online]. Available: `https://www.warema-group.com/en/WAREMA_group/WAREMA_Renkhoff_SE.php`.