Team Description Paper

Jon Martin, Daniel Ammon, Dominik Heigl, Florian Gramß, Alexander Gsell, and Tobias Fink

Institute of Technology Georg-Simon-Ohm,
Kesslerplatz 12, 90489 Nrnberg, Germany
http://www.th-nuernberg.de

Abstract. This is an amazing abstract. It will be more substantial when more text ist added to the main part.

1 Introduction

The Autonohm@work team at the Nuernberg Georg Simon Ohm University of Applied Sciences was founded in September 2014. The team consists of Bachelor and Master students leaded by a research assistant who guides and supervises them. To develop a functional mobile-robot- manipulator the different groups in the team had put much effort and knowledge into research to develop the packages to manage the robot. Our main focus is attended to mobile manipulation, object perception and navigation in a unconstrained environment. Only two members of the team that took part last year in Magdeburg continue. To compensate that, new members have joined us and the cooperation between the rescue and the atwork teams has been intensified.

2 Hardware Description

We use the KUKA youBot omni-directional mobile platform, which is equipped with a 5 DOF manipulator (Figure 1). At the end effector of the manipulator an Intel RealSense camera with a motion sensor has been mounted. Next to the camera we replaced the standard gripper from youBot with an also youBots soft two-finger gripper. Thanks to it we are able to grasp bigger and more complex objects more precisely.

A Hokuyo URG-04LX-UG01 laser scanner at the front of the youBot platform is used for localization and navigation. We are planning to add a second laser scanner of the same type on the back of the robot. This improves localization quality and ensures better obstacle avoidance, mainly when driving backwards with the robot.

Last year we used the internal computer, together with an external ASUS Mini PC (4 GB RAM, Intel Core i3). We used the internal computer in the youBot to start-up the motors and also for the SLAM. This was a huge error because we added an enormous data transfer between the two computers, what slowed down the complete system. To avoid the communication problems and

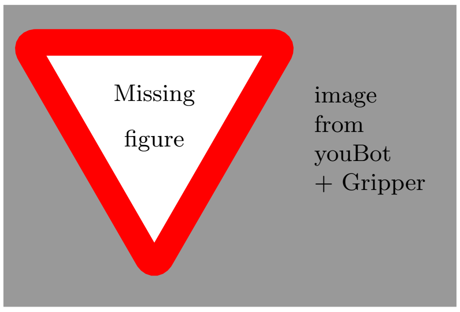

Fig. 1. KUKA youBot Plattform

latency between them both, we decided to run everything, except the motor drivers, on the external PC. We also replace the slow i3 for a more powerful CPU Intel Core i7-4790K, 4x 4.00GHz. Table 1 shows our new hardware specifications.

fill in specs and format table

For connecting the two PCs we use a router mounted on the bag of youBot. With external machines we can connect to the routers network and communicate with both PCs on the youBot.

We also added an Inertial Measurement Unit (IMU) to gather information about the heading of the platform. On the IMU runs a fusion algorithm which provides the orientation of the platform as quaternion or Euler angles.

Table 1. Hardware Specifications

PC 1	
CPU	XXXXXXXX
RAM	XXXXXXXX
HDD	XXXXXXXX
OS	Ubuntu 14.04
PC 2	
CPU	XXXXXXXX
RAM	XXXXXXXX
HDD	XXXXXXXX
OS	Lubuntu 14.04
Grasp	
a	XXXXXXXX
b	XXXXXXXX
c	XXXXXXXX
Lidar	
a	XXXXXXXX
b	XXXXXXXX
c	XXXXXXXX
Router	
a	XXXXXXXX
b	XXXXXXXX
c	XXXXXXXX
IMU	
a	XXXXXXXX
b	XXXXXXXX
c	XXXXXXXX

3 Software

The software architecture is based on the Robot Operating System ROS. The system runs with Ubuntu 14.04 and ROS Indigo installed. The ROS communication infrastructure is used to communicate and pass information between the nodes, as for example camera data or execution orders for the 5 DOF manipulator.

Several software tools are needed for image processing and controlling the system. In the following we generally describe the software we use when developing and in the contest.

cgdb For online-debugging we use cgdb. It is a command line interface to the GNU debugger software.
dd We create images from the hard discs to recover from a hard disc failure at the contest.
eclipse For all C++ development we use eclipse IDE.
helmut chrony
chrony
git Our software is maintained via git version control system. Git enables us working simultaneously on our software tree.
htop To watch active processes and CPU-Load we use htop.
openssh-server It provides access to the robot via ssh protocol.
QtSixA We steer our youBot with a Playstation 3 joystick. We need this software to connect the joystick to the PC via bluetooth.
screen Each time we accessing the platform through ssh we enter a screen session. With this software we are able to open multiple command line sessions, set a split screen and reconnect to a session if network connection drops.
stress We used stress to test our thermal management. It immediately can switch all CPU load to 100%, so it simulates worst case CPU load.
vim For developing over ssh connection on the robots PCs, or editing ROS-Launchfiles we use vim text editor with several plugins.

Additionally we use the following ROS packages:

amcl For localization problem we eventually use amcl package. It implements a particle filter algorithm. (see chapter 7)
navigation For global/local path planning, path regulation and obstacle detection/avoidance we plan to use the navigation stack from ROS.
robot-pose-ekf For fusing the data from the IMU and the wheel encoders we use the robot-pose-ekf. It is an implementation of an extended Kalman filter algorithm. It enables us to mix the two systems and improves the odometry data from the wheel encoders.
stdr-simulator We use it for simulating everything except the actuator. Every team member can test software through simulation in STDR, before testing with the robot in real.
youbot-driver Our youBot platform is controlled with this package. So we are able to send movement commands from PC1 to PC2 over ROS. On PC2, which is connected to the motor drivers and the actuator via EtherCAT protocol, youbot-drivers are running.

dominik: add software for vision please

Finally our own software packages:

ohm-tsd-slam For recording a map from the environment we use out own SLAM algorithm. (see chapter 6)

particle-filter For localization problem we eventually use our own particle filter instead of amcl package. (see chapter 7)

statemachine To control the robot we have implemented a statemachine algorithm with the statemachine framework from our laboratory. (see chapter 9)

4 Image Processing

add some text here

5 Image Manipulation

Intel RealSense camera the orientations and positions should be determined by given objects. Then this information is made available for the robot.

6 Mapping

what should come here:

1. describe our slam algo
2. describe improvements from last robocup
3. picture from a map
4. refs to our tsd papers
5. github source

7 Localization

As a first attempt on the RoboCup atWork competition, we decided not to try the Precision Placement and Conveyor Belt test. It was a good idea because during this attempt we had big problems with the localization and navigation, which are basic functions. This limited us to try further and more difficult tests. We gain experience and improve our software so we will try this time more difficult tests.

consistent style of RaW

We have three different strategies concerning the localization problem. The first one is to use our own particle filter algorithm. The second is to use the amcl package from ROS. And the third strategy is to use our SLAM algorithm mentioned in chapter 6 for localization. Because we have not yet decided which solution to use, we will describe all three in short.

7.1 Particle-Filter (TH-Nuernberg)

We are working on an own particle filter algorithm at our laboratory. Its functionality is close to amcl localization. If we get it working in time, we were proud to use our own localization algorithm at the contest.

7.2 Particle-Filter (ROS-AMCL)

The navigation stack from ROS-System includes a package called amcl (adaptive Monte Carlo localization). It provides a particle filter algorithm for robot localization. We already checked the compatibility of amcl algorithm and our SLAM approach. So we are able to record a map with our SLAM algorithm and afterwards locate and navigate in that map via amcl and navigation stack from ROS.

7.3 SLAM for Localization

SLAM means Simultaneous Localization and Mapping. This is because while recording a map, the robot needs to know its position in the map - so it must locate itself in the map while building the map. If we disable the mapping part from our SLAM algorithm, we are able to load a previously recorded map and locate the robot. The problem with this approach is that if the algorithm fails to process one measurement, the localization is lost in most cases and we have to quit the run. Particle filters are able to recover from such failures.

if we need more: multible laser scanner setup

8 Navigation

what should come here:

1. describe navigation stack in short (we will use navigation from localization stack)
2. describe problems and approach from last robocup

jon

9 Mission Planning

For the main control of the system, a State Machine with singleton pattern design is used. Every state is designed to be as small as possible. For the German Open 2015, we implemented three main states divided on smaller sub-states: move, grasp and deliver.

Once we get the tasks from the referee box, the first step for every test is to drive to a specific position and an specific orientation. Once we are on the desired position we smoothly approach to the service area and perform the task of grasping or delivering an object.

statemachine image: use yed uml for statemachine and export it as nice cropped pdf

The State Machine algorithm could be found on GitHub under our Laboratory's repository: "autonohm/obviously".

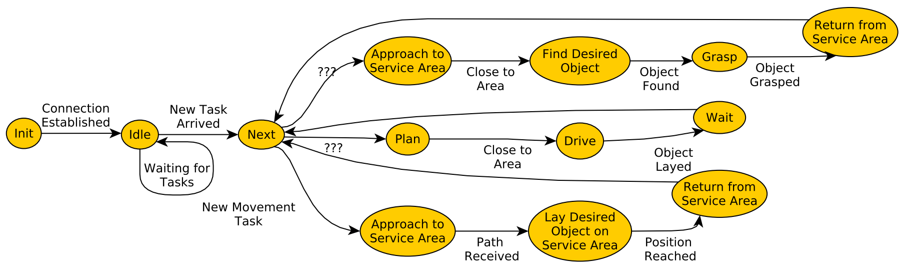

Fig. 2. Structure of the Statemachine

10 Object Manipulation

To grasp objects reliably an exact position from the object perception is needed. The position of objects will be calculated based on information, received from optical/infrared sensors (2D and 3D). After the calculation is finished the robot will navigate to a pregrasp position. Once the base has reached the final position, kinematics will lead the arm near the object. For precise gripping a 2D/3D optical/infrared sensor has been attached to the end effector. In gripping stance the arm- 2camera will be activated to measure the final gripping pose. Because manipulation is an upcoming issue in our robotic institute, we decided to build our own inverse kinematic.

11 Conclusion

In this paper we gave a brief description about our robots modification and functions. We use and develop existing software to make it even better but we also have to invent new methods and software.