The b-it-bots RoboCup@Work 2013 Team Description Paper

Sergey Alexandrov, Nirmal Giftsun, Praveen Ramanujam, Matthew Roscoe, Mallikarjuna Vayugundla,
Frederik Hegger, Nico Hochgeschwender, Jan Paulus, Gerhard K. Kraetzschmar

Bonn-Rhein-Sieg University of Applied Sciences
Department of Computer Science
Grantham-Allee 20, 53757 Sankt Augustin, Germany
Email: <first_name>.<last_name>@inf.h-brs.de
Web: http://www.b-it-bots.de

Abstract. This paper presents the b-it-bots RoboCup@Work team and its current functional architecture for the KUKA youBot robot. We describe the underlying software framework and the developed capabilities required for operating in industrial environments including features such as robust manipulation and object recognition.

1 Introduction

The b-it-bots RoboCup@Work team at the Bonn-Rhein-Sieg University of Applied Sciences (BRSU) was established at the beginning of 2012. The team consists of Bachelor, Master and PhD students, who are advised by one professor. The results of several research and development (R&D) as well as Master theses projects are going to be integrated into a highly-functional robot control software system. Through these types of course modules, our RoboCup@Work team is strongly interwoven with the Master by Research course in Autonomous Systems, which is offered at the BRSU[1]. Our main research interests includes mobile manipulation in industrial settings, omni-directional navigation in unconstrained environments, environment modeling and robot perception in general. Our approach is to first identify and evaluate in each subfield the state-of-the-art and the best practice solutions currently available. We then implement and integrate applicable algorithms to develop our own custom approaches with a focus on robustness in uncertain industrial environments.

2 Robot Platform

The KUKA youBot[2] is the applied robot platform of our RoboCup@Work team. It is equipped with a 5 DoF manipulator, a two finger gripper and an omni-directional platform. In the front and the back of the platform, two Hokuyo

[1] http://www.inf.h-brs.de/MAS.
[2] http://www.youbot-store.com

URG-04LX laser range finders are mounted to support robust localization and navigation. A sensor tower on the back platform hosts a Microsoft Kinect camera for common perception tasks, like scene segmentation and object recognition. Further, a Logitech webcam is mounted on the gripper for visual servoing. The internal computer is supported by a second high performance laptop which performs the computational expensive perception tasks.

3 Robot Software Framework

The underlying software framework is based on the ROS framework[3]. We use its communication infrastructure based on publish/subscribe, service server/client and action server/client to pass information between our components. The framework also provides interfaces to various common hardware devices like laser scanners or cameras. The wide range of various tools are utilized for visualization, testing and debugging the whole system.

Due to the deployment of ROS, we were able to migrate functionality from our Care-O-bot 3 robot to the KUKA youBot with less effort. All functional components are implemented as basic ROS nodes and can be connected to more cognitive capabilities to solve complex tasks. An example of a complex task in industrial robotics is "pick and place" (e.g. “bring a bolt to the production line and screw it on mounting point ’A’"). To perform this task, the robot needs to combine several actions such as navigation, object recognition, and object grasping. The actual scheduling is realized as a finite state machine (FSM). As the number of capabilities of the robot grows, more complex combinations of tasks can be performed. Although this created this versatility, scheduling by the developer is no longer feasible. Hence, we will deploy an additional task planning component as a replacement for the current FSM approach.

4 Navigation

In addition to map-based navigation, there is a necessity to control the robot in its local coordinate system with respect to the sensor data to make the interaction with the objects and environment much more precise. We have two components to handle these interactions.

4.1 Relative Motion Component

This component takes care of exhibiting behavior such as moving forward/backward/right/left or rotating clockwise/anti-clockwise based on odometry data.

4.2 Base Positioning Component

Placing the base appropriately to increase the reachability of the manipulator is necessary to have a robust manipulation. We have two strategies to deal with this problem.

[3] `http://www.ros.org`

Base Align-to-Workspace: A simple and effective approach has been used in order to align the robot perpendicular and in a certain distance to a workspace. A linear regression model is applied to fit a line to each front laser scan. The resulting orientation and distance to this line is fed forward to a closed loop controller which tries to minimize both "errors" (up to a certain threshold) by commanding linear and angular velocities to the robot's base.

Optimal Base Placement: The previously described component is used to compensate for inaccuracies in the map-based navigation. It ensures that the platform will be in the field of view (FOV) of the camera and the robot is able to locate and recognize objects on the respective platform. The recognition stage will provide a list of recognized objects. But the desired object which the robot needs to grasp and not be reached from the current position. Therefore, an additional component generates a distribution of optimal base poses based on the probability of manipulating an object successfully considering all the kinematic, map and task constraints. This component will let the robot to choose an optimal place to manipulate the object with high probability of success ensuring robustness.

5 Object Perception

Perception of objects relevant for the industrial environments is particularly challenging. The objects are typically small and often made of reflective materials such as metal. We use a Microsoft Kinect camera which provides both intensity and depth images of the environment. This enables effective scene segmentation and object clustering. But the spatial resolution is low even at the close range, and a significant degree of flickering corrupts the images. The information captured in a single frame is often not sufficient to determine the object type. We have therefore devised a three-stage pipeline (see Figure 1) which involves data accumulation over several consecutive frames.

The first stage is concerned with scene segmentation, or, more precisely, finding the workspace. We capture a single point cloud and apply a passthrough filter to restrict the FOV, which removes irrelevant data and reduces the computational burden. Next we perform plane segmentation with a region-growing algorithm. It is possible that the algorithm outputs more than one planar polygon. In this case we apply orientation constraints to remove irrelevant (e.g. non-horizontal) planes. Among the remaining we select the one with the maximum area. Finally, we shrink the polygon by several centimeters to make sure that it does not include the edge of the workspace.

The second stage is data accumulation. We filter each frame to keep only the points above the workspace polygon, which we then merge into an occupancy octree. Our experiments have shown that 30 frames is a reasonable tradeoff between the running time and the amount of information accumulated.

The final stage is object recognition. We partition the accumulated point cloud into clusters and fit minimal bounding boxes around them. The dimensions

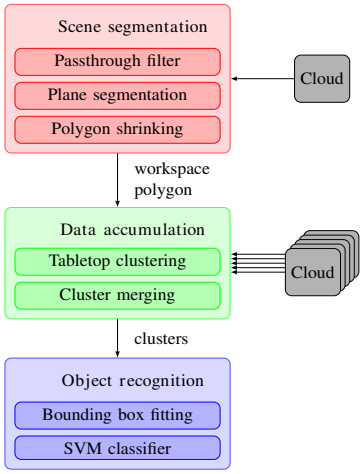

Fig. 1. Object perception pipeline

of the bounding box, the number of points in the cluster, and the average color of the points serve as an input to a classification neural network, which we have trained beforehand. Based on this information it outputs the predicted object type.

The obvious limitation of the implemented system is that it considers only the dimensions of an object, but not its shape. In the future work we would like to add shape descriptors and possibly other high-level features to increase the variety of objects that the system can recognize.

6 Object Tracking

In order to facilitate proper retrieval of objects which have been placed on the rear platform of the youBot we use object tracking to register and track the location of the objects. This is done by first recognizing an object and the location where it was placed. This is then stored for future use when we want to retrieve said object. We will then perform quick object recognition using the last known location of the object to limit our search window. The goal of this work is currently to eliminate the need for hardcoded placement position with an eye to expanding to other applications.

7 Object Manipulation

We have used two methods for object manipulation i.e. object grasping. The first method is based on visual servoing which is used as a fall back in the absence of grasp planning. The second method uses grasp planning and inverse kinematics to reach the grasp pose.

7.1 Visual Servoing

As a fall back to when the manipulation planning fails we use visual servoing to finish the grasp action. Once the arm has been placed within range of the target object we use visual servoing to align the robot (base/arm) with the target object. The gripper of the youBot is also aligned with the target object in order to better facilitate the grasping motion.

7.2 Grasp Planning

In order to plan how to actually grasp a known object, offline grasp planner has been integrated. A database with a set of grasps for each object has been generated using either manually setting grasp points or by using domain specific frameworks like OpenRAVE[4] or GraspIT![5]. Possible grasps are pruned based on various criteria like reachability, collision avoidance etc. We are planning to integrate a cluster based online planning approach for online grasp planning. Further, it is planned to incorporate the ability to grasp novel objects and to deal with positioning errors as well as sensor uncertainties.

8 Conclusion

In this paper we presented the functional core components of the current software architecture for the KUKA youBot robot. Besides the development of new functionality, we will also focus on porting existing components from our Care-O-bot 3 robot to the youBot platform. The migration from one robot to another was and is still an exhaustive exercise. In our current EU FP7 funded project BRICS (Best Practice in Robotics)[6] we are exploring first steps towards an improved software development methodology in robotics. We applied the component-oriented development approach defined in BRICS for creating our software which resulted in high feasibility when several heterogeneous components are composed into a complete system.

Acknowledgement

We gratefully acknowledge the continued support of the team by the B-IT Bonn-Aachen International Center for Information Technology and the Bonn-Rhein-Sieg University of Applied Sciences. In addition, the research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. FP7-ICT-231940-BRICS (Best Practice in Robotics).

[4] http://www.openrave.org
[5] http://www.cs.columbia.edu/~cmatei/graspit
[6] http://www.best-of-robotics.org