 DOI: 10.3217/978-3-85125-528-7-18

Industrial Grasping - An Autonomous Order Picking System*

Julia Nitsch and Gerald Steinbauer

Institute for Software Technology
Graz University of Technology, Austria
{*jnitsch,steinbauer*}*@ist.tugraz.at*

Abstract
Automated storing, retrieving, and delivering items is an important part of Industry 4.0 application. For low-volume this task is done usually manual. In this paper we present an architecture and a proof-of-concept implementation for order picking using the robot Baxter from Rethink Robotics. The main contribution besides providing full functioning prototype is a dependable control architecture.

1. Introduction

Industry 4.0 is one of the keywords, when we talk about the next level of production. Industry 4.0 represents the 4^{th} industrial revolution and promises improvement of productivity through automated, self-organizing and self-optimizing processes. It addresses the needs of high-quality products which are also highly customized but still ready for mass production.
This work contributes to the field of Industry 4.0 by developing an assistant robot for order picking. Such robots share the environments with humans. In a typical warehouse system items can be stored in larger transport boxes. The transport boxes again can be stored in shelves to save space. If a specific item needs to be picked the transport box first needs to be pulled out of the shelf and then the item can be picked and delivered. This procedure is called order picking. For items with a moderate frequency this type of picking is usually done by hand which is a monotonic and time consuming task. In our scenario we tend to automatize that task.
The system we propose is based on a 3-TIER architecture. The planning layer uses an artificial intelligence (AI) planner to generate a list of skills the robot has to execute. The planner outputs a list of skills, the robot needs to execute in order to achieve its goal. Skills are composed of skill primitives. These primitives can perform perception, manipulation, grasping tasks or any combination of those. Failures are already detected at the level of the primitives where local recoveries can be performed. If these recoveries fail too, these errors are reported to the executive layer. This architecture ensures the detection and recognition of failures. Together with appropriate steps for recovery dependable execution is achieved. The proposed architecture was realized as a proof-of-concept implementation using the two arm robot Baxter from Rethink Robotics. For details about the realized system we refer the interested reader to [10].
The reminder of this paper is organized as follows. In the next sections we briefly discuss related research and the target environment. In Section 4. the proposed system architecture is presented. Due to the space constraints we focus on skill primitives. In the next section we briefly present an evaluation focused on the skill primitives. In section 6. we draw some conclusions.

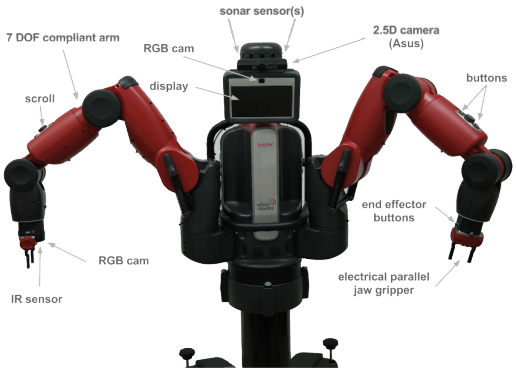
*This work was supported by incubed IT GmbH.

2. Related Work

Numerous works exist about high-level planning for robot systems solving complex tasks using sets of simpler system capabilities. This system capabilities are called skills. Dividing a complex task in such skills has multiple benefits like flexibility, reuse of skills and good software portability.
Pederson et al. shows in [13] the division of a complex task a sequence of multiple subtasks (= skills). Skills are described being the *fundamental building blocks* or the *system capabilities*. If a new complex task should be executed, the system needs not being reprogrammed. It is sufficient to simply reorder the skills. Similar to our approach, an execution monitor surveys the outcome of the skills.
In [12] skills are ported to different robotic platforms. Skills get further decomposed into skill primitives. With this detailed decomposition the hardware level is abstracted from the skills itself. The advantage of modularity and the abstraction of tasks is pointed out.
The authors in [14] introduce a 4-TIER architecture, with the same idea of abstraction for skills and skill primitives as in the previous papers. The lowest layer ensures the hardware abstraction and so the re-usability on different platforms. The next layer contains action and perception primitives. The top layers handles the planning task. As the previous addressed work, this abstraction is used for easing the human robot-interaction. All these papers show a clear distinction between tasks, skills and primitives and focus on portability and easy execution of complex new tasks. But their focus is on human-robot interaction. The human in the loop defines a new task through reordering skills. The next works present a successful task planning utilizing artificial intelligence (AI) planner instead of humans in the loop ordering skills. In [6] Huckaby defined skills with preconditions and effects in the model space of the problem. The initial state and goal are stated in the process space. They proposed PDDL [7] as planning language.
In [6] the focus lies on the high-level. It is assumed that skills and their primitives always succeed. In [11] a system is proposed which transfers the high-level description from the AI planner to a behavioral state machine. Failures in the primitive execution are detected by a vision system and recoveries are performed.
Finally some works addressing the order picking problem are discussed. The authors in [9] present a mobile bin picking system. Items are picked from a box standing on the ground and placed at a delivery station. The high-level of this system is a finite state machine (FSM).
In [3] a software architecture and their implementation for grasping objects is presented. Some of these concepts are used in our work too. The collision environment is a 3D occupancy grid excluding robot parts. Known and recognized objects are represented as geometric primitives or as mesh models of the objects. In [1] the authors present a pick and place approach where they have to deal with known and unknown objects, cluttered workspace and noisy sensor data.

3. Target Environment and System

For the a proof-of-concept implementation of the proposed order picking system we use the robot Baxter from Rethink Robotics (see Fig. 1a). It is a two-arm robot with internal sensors such as cameras and proximity sensors in the wrists. In order to get a global overview of the environment we added a RGBD camera on top. The environment Baxter operates in is shown in Figure 5b. It is a mock-up of a typical manual storage.



(a) Baxter with its inbuilt and additional mounted sensor.

(b) Environment in which Baxter performs the order picking task.

Figure 1: Robot Baxter and the environment it operates in.

4. Architecture

In Figure 2 the conceptional overview of the proposed 3-TIER architecture is shown. The 3 layers for planning, executive and behavioral control are separated. The communication is clearly defined. The top layer represents the planning layer. The planner uses the information of the domain and the problem to generate a plan. The plan is a sequence of skills that have to be executed to reach a given goal. The plan is forwarded to the next layer. The executive layer takes care of the execution of each skill. It knows about the composition of the skill primitives. The primitives are located in the behavioral layer. The advantage of this abstraction is its clear structure and its modularity. For further reading about the 3-TIER architecture please see [8, p. 244–277].

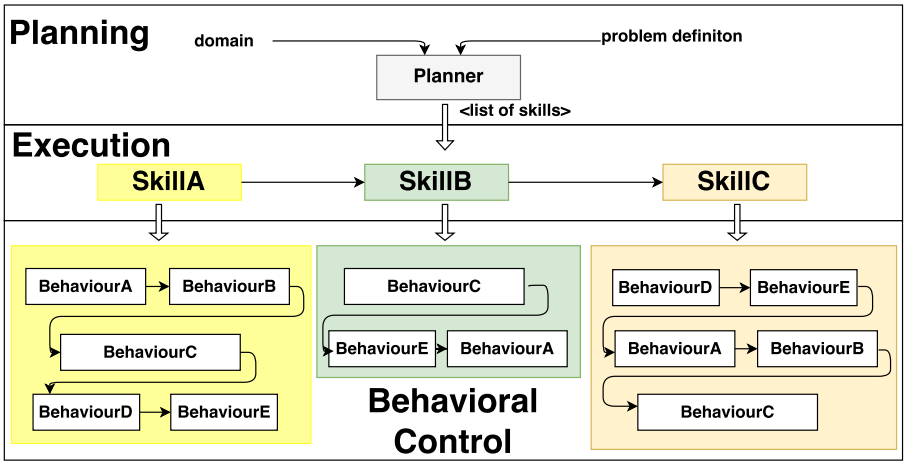


Figure 2: Overview of proposed system's architecture.

4.1. Planning Layer

The top layer of the 3-TIER architecture is the planning layer. The planning layer uses a domain and problem description of the given environment and task. It is based on the Planning Domain Definition Language (PDDL) modeling the system capabilities (further on called skills), the current state of the environment and the goal state. The authors of [6] showed that PDDL is an appropriate choice for

robotic tasks. The domain contains information about all the objects which can appear in the environment. Further it knows about the skills the robot is able to perform. A skill is defined through its name and the its parameters. A skill has a precondition and an effect. The precondition is the state the environment needs to be in before the action can be performed. The effect of a skill is the description of the environment after the skill is performed.
The problem describes the initial state s_0 and the goal g, which are a set of propositions [4]. The goal state s_g is a state, that satisfies g. First object instances are defined, which occur in the environment and their initial properties are stored. The planner takes the domain and the problem description and generates a list of skills, which need to be performed to solve the given problem. We use the planner SGPlan6 [5]. This list of skills is forwarded to the executive layer of the 3-TIER architecture.
For solving the order picking task, the following skills are required: *moveBoxFromLevelToTray*, *movBoxFromTrayToLevel* and *graspItem*. Lets assume an environment containing transport boxes *BOX_A*, *BOX_C* and a shelf with levels *LEVEL_1*, *LEVEL_2* and the goal of picking one item from *BOX_C* placing it at the delivery box *DBOX_C* and picking two items from *BOX_A* placing it at delivery box *DBOX_A*. The planner comes up with the following plan (see Listing 1).The name of the skill is the first parameter, followed by the parameters the skill requires. So the first skill, which has to be performed is *moveBoxFromLevelToTray*. The *BOX_C* is moved from *LEVEL_2* to the *TRAY*.

Listing 1: Output of planner for example domain and problem.

```
0 (MOVEBOXFROMLEVELTOTRAY BOX_C LEVEL_2 TRAY)
1 (GRASPITEM BOX_C DBOX_C TRAY)
2 (MOVEBOXFROMTRAYTOLEVEL BOX_C LEVEL_2 TRAY)
3 (MOVEBOXFROMLEVELTOTRAY BOX_A LEVEL_1 TRAY)
4 (GRASPITEM BOX_A DBOX_A TRAY)
5 (GRASPITEM BOX_A DBOX_A TRAY)
6 (MOVEBOXFROMTRAYTOLEVEL BOX_A LEVEL_1 TRAY)
```

4.2. Executive Layer

The executive layer receives a list of skills from the planner. The executive layer handles the execution of single skills. Each skill is composed of skill primitives, which are the fundamental building blocks of each skill. The executive layer knows about this decomposition and ensures that primitives are executed in right order to guarantee a successful skill execution. This decomposition of the skills *moveBoxFromLevelToTray*, *moveBoxFromTrayToLevel* and *graspItem* is shown in Table 1. The composition of skill primitives for each skill is intrinsic knowledge of this layer. Further it monitors the outcome of each primitive. This layer has also the opportunity to perform recovery behaviors, if primitives fail. If no recovery can be performed or the recovery fails, this failure is reported to the planning layer. The decomposition of the *moveBoxFromLevelToTray* and its execution is shown in Figure 3.

skills	**moveBoxToRack**	**graspItem**	**moveBoxToLevel**
skill primitives	detectHandle graspHandle moveArmToSupportPose pullBox moveBox deliverBoxOnTray	detectItem graspItem deliverItem	detectHandle graspHandle moveArmToSupportPose pullBox moveBox deliverBoxOnLevel

Table 1: Within this table the skill primitive composition of all skills are listed.

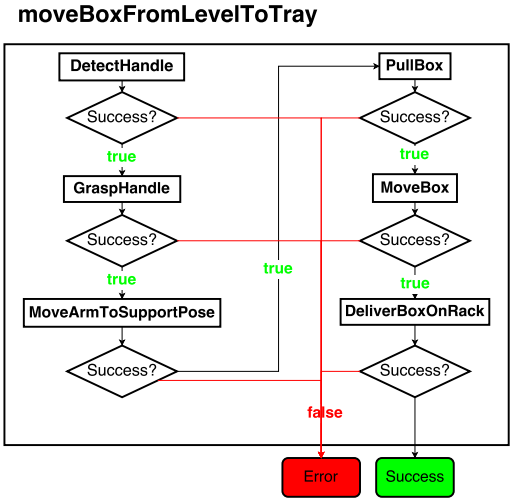


Figure 3: Decomposition of skill *moveBoxFromLevelToTray* into its primitives and how executive layer handles the execution.

4.3. Behavioral Control Layer

This layer holds all skill primitives. The primitives are platform-dependent and have to be re-programmed for specific robot platforms. The upper layers are immediately portable to other platforms. The primitives have a defined interface but their implementation is different for different platforms. Skill primitives can perform perception, manipulation, grasping tasks or any combination of those. Local recoveries are performed in this layer. If a recovery is impossible skill primitives report their error. A skill primitive can be used by different skills.
Figure 4 depict for instance the primitive of looking for items once the box is on the tray. For the box detection the point cloud of the top RGBD camera and the PCL implementation [16] of FPFH features [15] (initial detection) and ICP (fine alignment) are used. The control of the arms are realized using the MoveIt! framework [2]. Items are detected using the RGB cameras in the wrist. Figure 5a shows the inspection of a box while Figure 5a depict the internal representation of the situation.

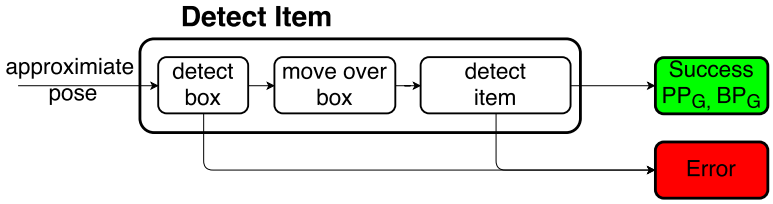
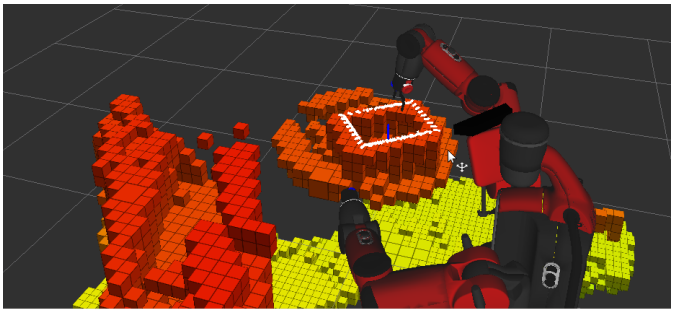


Figure 4: Detect Item primitive. PP_G represents the global item pose. BP_G represents the global box pose.

5. Results

The major result of this work is a working prototype implementation of the proposed order picking system based on the robot Baxter and standard software packages such as ROS or MoveIt!. But we are interested in particular in the dependability of the system. Therefore, we performed a detailed evaluation of the individual skill primitives.
For the evaluation we executed individual skill primitives multiple times (around 50 trials each) in

(a) Baxter inspecting the box with it's end-effector camera.



(b) Baxter inspecting the box visualized in RViz. White points indicate the detected box. Orange voxels represents the collision scene.

Figure 5: Realization of the primitive Item Detection.

given setups. Details about the evaluations can be found in [10]. In Figure 6 the results of the individual skill primitives are shown. The green bar indicates the successful execution rate, the gray bar marks the failure executions which are detected by the system and the red bar shows the failures which are not detected by the system. Even if the success rate of some primitive is not overwhelming, the system detects the failure and reacts to it. The recognition of failures is one fundamental ability of reliable systems. As soon as the errors are detected, the robot can react to it autonomously.

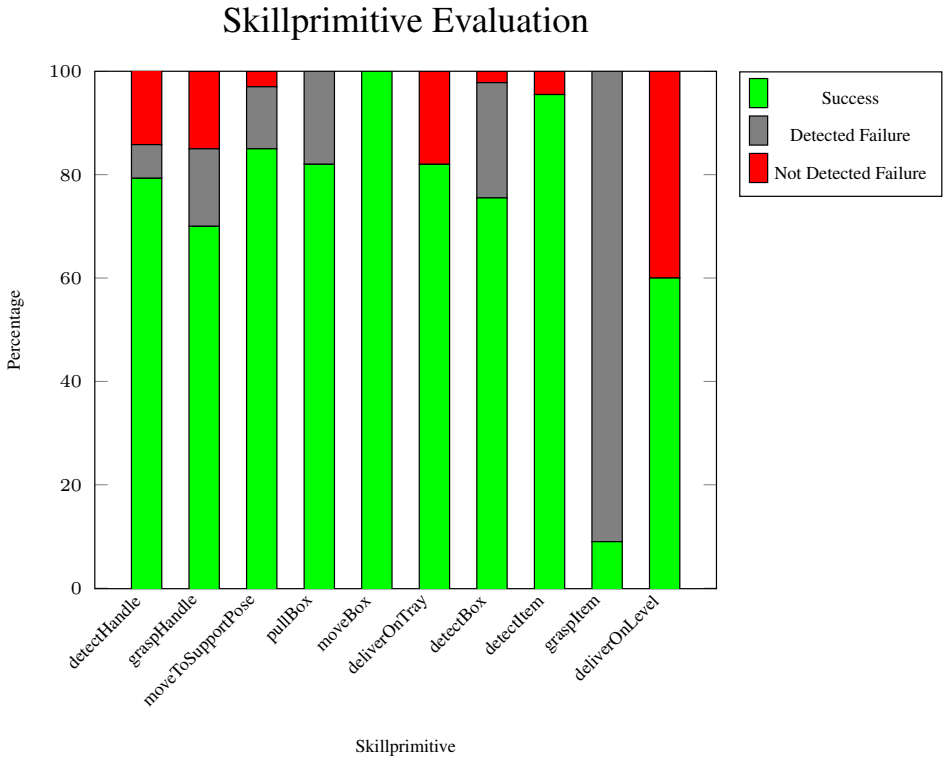


Figure 6: Skillprimitive evaluation

6. Conclusion

In this paper an autonomous order picking system was presented. In order to keep the system modular and portable a 3-TIER architecture was developed. The planning layer utilized an AI planner, which uses a PDDL description of the planning problem. The planner received the description of system skills, as well as start and goal state and provided a list of planned skills. Each skill is composed of skill primitives. These skill primitives are needed to address manipulation of the box, grasping items and perceptual tasks. The decomposition of skills into primitives enriched with monitoring and recovering capabilities contribute to the dependability of the system. The proposed system was implemented as a prototype using the robot Baxter and standard robotics software libraries.
Using this prototype implementation the concept of the skill primitives was evaluated. Although most primitives worked quite well, the evaluation pointed out some problems of this proof-of-concept system. Within most primitives, the major problem was that the execution of planned trajectories was aborted because Baxter was not able to execute them precisely enough. However these errors were detected by our system and reported to the high-level controller. For future work a more reliable execution of arm motions by Baxter needs to be addressed.

References

[1] S. Chitta, E.G. Jones, M. Ciocarlie, and K. Hsiao. Mobile manipulation in unstructured environments: Perception, planning, and execution. *IEEE Robotics & Automation Magazine*, 19(2):58–71, June 2012.

[2] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit! *IEEE Robotics & Automation Magazine*, 1(19):18–19, 2012.

[3] Matei Ciocarlie, Kaijen Hsiao, Edward Gil Jones, Sachin Chitta, Radu Bogdan Rusu, and Ioan A Şucan. Towards reliable grasping and manipulation in household environments. In *Experimental Robotics*, pages 241–252. Springer, 2014.

[4] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning: theory & practice*. Elsevier, 2004.

[5] Chih-Wei Hsu, Benjamin W Wah, Ruoyun Huang, and Yixin Chen. Handling soft constraints and goals preferences in SGPlan. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2006.

[6] J. Huckaby, S. Vassos, and H.I. Christensen. Planning with a task modeling framework in manufacturing robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5787–5794, Nov 2013.

[7] D. Mcdermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL - The Planning Domain Definition Language. Technical Report TR-98-003, Yale Center for Computational Vision and Control, 1998.

[8] Robin Murphy. *Introduction to AI robotics*. MIT press, 2000.

[9] Matthias Nieuwenhuisen, David Droeschel, Dirk Holz, Jorg Stuckler, Alexander Berner, Jun Li, Reinhard Klein, and Sven Behnke. Mobile bin picking with an anthropomorphic service robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2327–2334. IEEE, 2013.

[10] Julia Nitsch. Industrial Grasping. Master's thesis, Faculty for Computer Science and Biomedical Engineering, Graz University of Technology, Graz, Austria, 2016.

[11] K. Okada, Y. Kakiuchi, H. Azuma, H. Mikita, K. Murase, and M. Inaba. Task compiler : Transferring high-level task description to behavior state machine with failure recovery mechanism. In *IEEE International Conference on Robotic and Automation (ICRA)*, May 2013.

[12] Mikkel Rath Pedersen, Lazaros Nalpantidis, Aaron Bobick, and Volker Kruger. On the integration of hardware-abstracted robot skills for use in industrial scenarios. In *2nd International IROS Workshop on Cognitive Robotics Systems: Replicating Human Actions and Activities, (Tokyo, Japan)*, Nov 2013.

[13] M.R. Pedersen, D.L. Herzog, and V. Kruger. Intuitive skill-level programming of industrial handling tasks on a mobile manipulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4523–4530, Sept 2014.

[14] Francesco Rovida, Casper Schou, Jens Skovand Dimitris Chrysostomou Andersen, Rasmus Skovgaardand Damgaard, Simon Bøgh, Mikkel Rathand Bjarne Grossmann Pedersen, Ole Madsen, and Volker Kruger. Skiros: A four tiered architecture for task-level programming of industrial mobile manipulators. In *International Workshop on Intelligent Robot Assistants, (Padova, Italy)*, 2014.

[15] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3217. IEEE, 2009.

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