20th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Luxemburg, September 2015

A Skill-Based System for Object Perception and Manipulation for Automating Kitting Tasks

Dirk Holz, Angeliki Topalidou-Kyniazopoulou, Francesco Rovida,
Mikkel Rath Pedersen, Volker Krüger, and Sven Behnke

***Abstract*—The automation of kitting tasks—collecting a set of parts for one particular car into a kit—has a huge impact in the automotive industry. It considerably increases the automation levels of tasks typically conducted by human workers. Collecting the parts involves picking up objects from pallets and bins as well as placing them in the respective compartments of the kitting box. In this paper, we present a complete system for automated kitting with a mobile manipulator thereby focusing on depalletizing tasks and placing. In order to allow for low cycle times, we present particularly efficient solutions to object perception as well as motion planning and execution. For easy portability to different platforms, all components are integrated into a skill-based framework that is tightly coupled with a task planning component. We present results of experiments at both a research laboratory environment and at the industrial site of PSA Peugeot Citroën serving as a proof of concept for the overall system design and implementation.**

Keywords*—Object detection, grasping, motion planning, pre-computed trajectories, skill framework

I. Introduction

In the past decades, the paradigm of car production has shifted from mass production to increased customization of products (build-to-order). More customized products with increased assembly combinations implicitly means more components to store, transport and feed to the production line. Due to this variability of the production and to the diversity of suppliers and parts, part handling during the assembly stages in the automotive industry is the only task with automation levels below 30%.

Following this trend, *kitting type distribution* has developed massively in the automotive industry over the past few years. The main idea is to concentrate the value added on the production line and decentralize re-packing operations. Kitting operations are usually performed by operators called *pickers*. These pickers collect parts as needed from the respective containers, i.e., bins and pallets, and place them in kitting boxes with several compartments. Once complete, the kits are delivered to the production line and synchronized with the cars being produced. The full automation of such task will not only have a huge impact in the automotive industry but will also act as a cornerstone in the development of advanced mobile robotic manipulators capable of dealing with unstructured environments possibly shared with human coworkers, thus opening new possibilities for the automotive industry and mobile robots in the factory of the future in general.

D. Holz, A. Topalidou-Kyniazopoulou, and S. Behnke are with the Autonomous Intelligent Systems Group, University of Bonn, Bonn, Germany, {holz, topalido, behnke}@ais.uni-bonn.de.

F. Rovida, M. Rath Pederson, and V. Krüger are with the Robotics, Vision and Machine Intelligence (RVMI) Lab, Aalborg University, Copenhagen, Denmark, {francesco, mrp, vok}@rvmi.aau.dk.

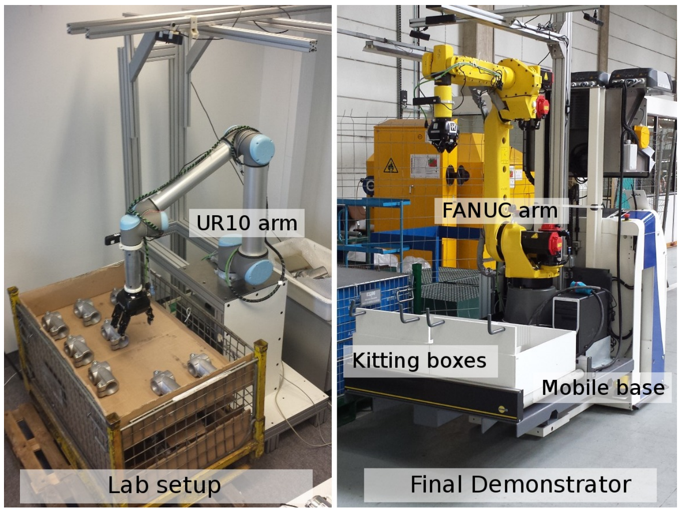


Fig. 1: Robot platforms and pallets, left: the lab setup used for development, right: demonstrator at the industrial end-user site. The platforms are equipped, respectively, with a Universal Robots UR10 and a FANUC M-20iA/35M arm, a Robotiq 3-finger gripper, and four RGB-D cameras for perceiving the workspace and objects in front of the gripper.

In the course of a larger project on kitting using mobile manipulators (Fig. 1), we have developed a system for automated grasping of parts from pallets and placing them in kitting boxes. This task comprises three major problem domains: 1) environment and object perception, 2) motion planning and execution, and 3) integrating all components into a reliable and easy-to-use system. We make the following contributions:

- We present a particularly efficient object perception pipeline that explicitly exploits the characteristics of depalletizing problems such as well-separated parts (under the assumption that intermediate packaging is removed). The pipeline allows detecting and localizing parts in as little as 300ms, thus allowing for low cycle times.
- In order to further decrease cycle times, we propose a mixture of motion planning and pre-computed trajectories between common poses, thus considerably decreasing the need for computationally expensive motion planning and making the overall system particularly efficient.
- In order to facilitate integration and portability between different robots and setups, we present a novel skill-based framework, in which the complete object perception and manipulation pipeline is integrated.

The system is based on a multi-layered architecture, where low level *primitives* interact directly with the robot hardware, *robot skills* coordinate the primitives to provide a higher-level functionality, and *tasks* aim to solve a high-level goal in the factory, by using the skills.

II. Related Work

A. Architectures

The complexity of a robot system increases drastically when including perception and advanced reasoning into the set of robot capabilities. In deployed systems, there is therefore a need of a rigorous software architecture to integrate the knowledge and manage teaching, learning and execution phases. Moreover, it is important to subdivide the software into modular parts, to increase the portability and the code reusability. A common approach is to split the software into several abstraction layers. For example, the KDMF framework [1] uses a classical component-based architecture, divides the process into components and uses semantic reasoning to automatically connect them. A more ambitious, but less practical, architectural pattern is the *5C-composition pattern* [2], that aims to create software blocks that internally can be composed of other blocks, without limiting the possible levels of abstraction. Our software framework, *SkiROS*, lies in the middle and is organized in four layers that gradually abstract the programming using device, primitive, skill and task layers.

Numerous middlewares have been proposed to support the implementation of robotics applications, such as the Robot Operating System (ROS)[1] and OROCOS[2]. These middlewares hide the complexity of the robot system from the developer and highly simplify the development of a distributed software. Robotic software frameworks nowadays are usually built on top of these middlewares. Our framework, *SkiROS*, is build on top of ROS. Other ROS projects in a similar direction include *MoveIt!* [3], *ROSCo*[3] and *SMACH*[4]. The former is focused on industrial manipulators and embeds many useful mechanism, like collision-detection, arm motion planning and trajectory execution, etc., suitable for the low-level layer we define as *primitives*. ROSco and SMACH work on a higher level of abstraction with respect to MoveIt!, and are comparable to our *skill* layer. These are frameworks for rapidly creating complex robot behaviors, under the form of a hierarchical finite state machine. Both frameworks only allow static compositions of behaviors and cannot adapt those to new situations during execution.

More adaptive solutions that can be used at a skill level are the Reactive Action Packages (RAP) [4], specifically designed to create short-term reactive robot behaviors, Object Action Complexes (OAC) [5], or the manipulation primitive nets [6]. Other approaches that try to integrate skill programming with learning capabilities are the works presented in [7] and [8]. The latter starts from an initial set of predefined primitives and generates artificial neural-networks to connect them.

At task level, research is focused on planning. One of the earliest developments in autonomous robots, Shakey, uses symbolic representations of tasks and the STRIPS planner to reason about which actions may lead to accomplishing a goal [9]. Dozens of other algorithms have followed, based on the same ideas, in which they act on a set of actions that alter the current world state. Planning algorithms require a description of the world state on which they can reason about. A widely used approach to create this knowledge is to formalize it under the form of an ontology. The KnowRob framework [10] is a prominent example of that way of thinking in the area of service robotics. It is based on SWI-Prolog and its Semantic Web library which allows loading and accessing ontologies represented in the Web Ontology Language (OWL). A similar approach, which differs in the implementation of the semantic database, based in this case on openRDF, is the one presented in [11], [12], [13] for the Rosetta project. Both KnowRob and Rosetta use the semantic web standard, and exploit the ontologies not only to define the world model, but also to formalize the robot actions, the devices and the robot self-description. Another solution is to define a world model using a Domain Specific Language (DSL) definition tool, like it has been done by [14].

B. Perception for Depalletizing and Bin-Picking

Using vision sensors for object detection and pose estimation in industrial robotics applications has been focus of research within the last decades. Early work used very specific geometric features of objects which are found in intensity images. Rothwell et al. [15] use invariants of such geometric features under perspective transformation for fast indexing and matching of observed edges of planar shapes to a model database. Other works such as [16], use singulated geometric shapes such as circular or polygonal structures on the objects as landmark features to find the pose of the object. Such features have also been extracted from range images—one prominent early example being the 3DPO system [17].

More recently, Papazov et al. [18] use depth images of a Microsoft Kinect sensor to find objects in a scene for picking and placing by a robot. Their recognition approach compares surfel-pair matches between scene point cloud and object model in a RANSAC scheme for detection and pose estimation. Drost et al. [19] propose to use Hough voting with surfel-pair matches instead. This approach has been extended by Choi et al. [20] with oriented points on contours of the object. They acquire point clouds of small objects in a transport box, and grasp them with a high success rate with an industrial robot arm. Skotheim et al. [21] also propose a voting scheme based on pairs of oriented points in 3D point clouds. Their approach mounts a laser triangulation sensor directly at the wrist of the robot such that the robot could scan the object from arbitrary view points. Our approach finds a highly accurate pose of the object through segmentation and dense model alignment by exploiting that in a depalletizing scenario, a coarse initial guess of the object orientation on the palette is typically known.

Pretto et al. [22] use a monocular vision system in a bin picking scenario, i.e., they find objects using intensity images. The approach is based on matching contours in the scene with the object model, assuming planar surfaces on the objects. They report cycle times of up to 7s in a statically mounted robot setup. Our approach does not make such

[1]ROS: http://ros.org
[2]OROCOS: http://www.orocos.org/rtt
[3]ROSCo: https://www.hsi.gatech.edu/hrl/project_rosco.shtml
[4]SMACH: http://wiki.ros.org/smachs

strong assumptions on object shape. Brachmann et al. [23] use a learned object representation combining dense 3D object coordinate labeling and dense class labeling for textured and texture-less objects. They achieve high detection accuracies and runtimes around $500\,\mathrm{ms}$, but only for single objects as opposed to scenes containing multiple instances of the same object.

In own previous work [24], we have developed an approach to mobile robot bin picking where objects are modeled using geometric primitives. Compounds of both 2D contour and 3D shape primitives are found by graph matching of primitives detected in 3D point clouds. Our new approach is less restrictive in the sense that objects need not to be composed from geometric primitives. Instead, dense registration of acquired color and depth information is used to accurately localize parts. In addition, the registration confidence is used to verify that the found part does not deviate from the known object model, e.g., if the container contains a wrong part.

III. Skill framework

The background of this work is a European Project on shop floor logistics and mobile manipulation in industrial settings and automating kitting tasks in particular. One of the cornerstones of the project is vertical integration: A so-called *logistic planner* coordinates the enterprise information, enterprise resource and manufacturing execution systems (MES) of the end-user and connects it to a central *mission planner*. The mission planner coordinates a fleet of mobile manipulators, by generating and delegating kitting orders. The kitting orders are processed by the individual mobile manipulators. Each mobile manipulation robot processes two kitting orders at a time.

On each robot, a *task manager* constitutes the connection point with the logistic planner. The task manager provides to the logistic planner a list of available robot skills and receives from the logistic planner 1) the kitting orders, with preliminary sequences of skills required to complete them, and 2) a portion of the world instance, that is injected into the local memory of the robot and used to reason about the environment. The task manager schedules the execution of skills by connecting to the *skill managers* and quickly re-plans the skill sequence in case of failures during execution. An overview of these interactions is shown in Fig 2a. In the following, we focus on the skill manager, the underlying world model, and the developed skills for efficient part localization and manipulation, using the use-case of picking parts from pallets.

A. System Overview

In this paper, we neglect the vertical integration to the logistic planner and assume that the robot's task manager has received a preliminary plan of collecting a number of parts for a kit. This plan comprises a sequence of high-level robot *skills* and their parameters, e.g. '*pick up alternator model X*'. The robot system sequentially executes the robot skills in the plan through the skill framework and the skill model explained in the following section.

As a demonstrator for this work, we use two similar robot systems, equipped with a robot arm mounted on a frame and a mobile base. Referring to Fig. 1, we use a total of four RGB-D cameras on each system for perceiving the workspace around the robot, the kitting boxes at the front of the robot, and objects in front of the gripper. For manipulation, we use a Universal Robots UR10 and a Fanuc M-20iA/35M robot arm. Both arms are equipped with a Robotiq 3-finger underactuated robot hand. The mobile base—based on an autonomous forklift—has its own available skills, namely autonomous navigation, but they will not be discussed in this paper. Note that only the final demonstrator used at the end-user site features the mobile base. In the lab setup, we only use the manipulation part.

B. Skills, Framework and Managers

The core idea of our robot skills is that they are fundamental software building blocks causing a modification on the world state. The set of available skills and the encoded modifications of the world state define the input to the mission and task planners.

The conceptual model of a complete robot skill is shown in Fig. 2b. The input parameters to skills are *objects* from the world model, i.e. the skills are object-centric and do not require complex parameters to be executed. This does not only simplify the planning problem to only deal with classical task planning, but also enables shop floor workers to explicitly program the robot on the skill-level, as we have previously shown in [25]. However, the execution routine within the skill needs to be more advanced than traditional robot macros. We will give an example of a skill implementation in Section IV.

An integral part of the skills concept is concatenation, i.e., skills can be combined to form a complete robot program or task. To ensure both concatenation and verbosity and robustness of the skills they include pre- and postcondition checks, as well as continuous evaluation of their performance. This allows to provide information of skill failures to the higher-level components, as well as ensuring proper concatenation.

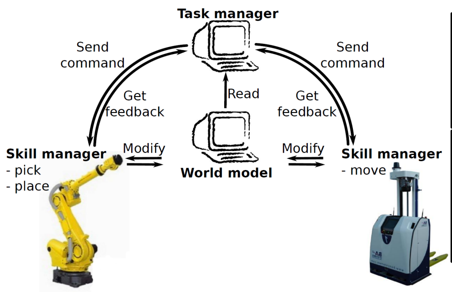
We will not give a further, comprehensive introduction to the robot skills in this paper, but instead refer to our previous works [26], [27]. For this work, it suffices to say that the manipulation skills presented below are implemented according to the conceptual model shown in Fig. 2b in our skill framework, *SkiROS*.

As illustrated in Fig. 2c, SkiROS is divided into 4 layers to organize the software processes. The abstraction layers of SkiROS are:

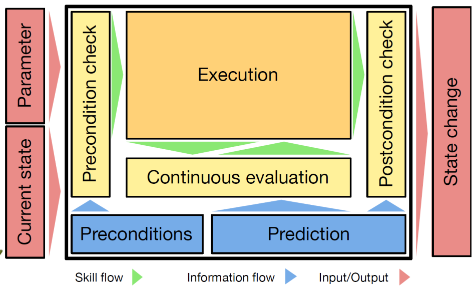
1) Device layer: realizes the hardware abstraction and presents an interface to the devices that make up the system.

2) Primitive layer: embeds and coordinates *a)* motion primitives, i.e. software blocks that realize a movement controlled with a multi-sensor feedback, and *b)* services, i.e. software modules that realize a generic computation. An integral part of this layer are the *primitive managers*. The common characteristic of the modules in this layer is that they do not influence the world state in the shared world model, but only the intrinsic robot state, which is not needed by the higher-level layers.

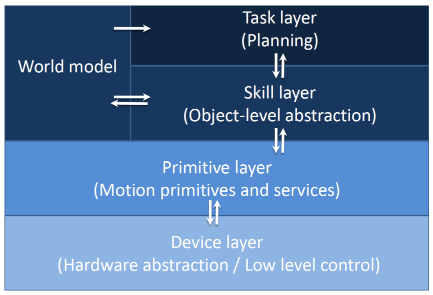
3) Skill layer: embeds and coordinates skills. An integral part of this layer are the *skill managers* that are associated to every sub-system of a single robot, e.g., the mobile base and the manipulator in our platform, The purpose of the skill



(a) Interaction of components



(b) Model of a robot skill


World model
Task layer
(Planning)
Skill layer
(Object-level abstraction)
Primitive layer
(Motion primitives and services)
Device layer
(Hardware abstraction / Low level control)

(c) Skill-based Architecture *SkiROS*

Fig. 2: (a) The task manager reads the world state and operates on it by sending commands to the skill managers. (b) Each skill is modeled by an abstract description with pre-conditions and predictions of the world state change as well as functional blocks. Their execution is always coupled with pre-condition and post-condition checks. (c) The framework is divided into four layers: the *device layer*, the *primitive layer*, the *skill layer*, and the *task layer*.

manager is to execute the skills, and concurrently supervise their status and execution, as well as relay information about available skills to the upper layers.

4) Task layer: the higher level embeds task-level programming and planning capabilities, as well as execution of saved tasks. Each complete robot system has an associated *task manager*. This connects to the various *skill managers* in the system and collects the complete list of available skills. The task manager is designed to accept from outside two possible inputs: a skill sequence or a desired goal state in the world model. In the former, a skill sequence, including parameters, is specified by e.g. the mission planner or a human operator. In the latter, the task manager's integrated planner automatically finds a skill sequence for the given goal state.

The common characteristic of the managers in every layer is that they import the defined modules as ROS plugins[5] and present the set of plugins to the upper layer over the ROS network. As the managers communicate over the ROS network they can be executed on different machines, creating a flexible distributed network. Our structure of our architecture reduces the ROS network communication by having each manager collecting internally, on a single process with a shared-memory communication, sets of related modules. The purpose of the modular abstraction layers presented in this section is to 1) organize and give a clear structure to the code and the ROS nodes, 2) create an intuitive workspace on every level of abstraction, and 3) support autonomous run-time selection of the best module to use for the problem at hand. To realize the latter, a support from a knowledge integration framework is fundamental. It is provided from our world model server, presented in the next section.

C. World Model

The world model plays a central role in all advanced knowledge-driven systems, as every planning system relies on a world model to organize the knowledge about the objects in the world, their properties and their relations. Starting from this information, the planner can apply logic rules to discover a correct sequence of actions (skills), in order to reach a desired goal state from the initial state. Constructing a knowledge base for robot application is a challenging task since robotic applications have very specific demands. Generally, knowledge is hard-coded in the task planner in a suitable way but this solution is a problem when there is necessity to modify the knowledge and propagate it over different software modules. In our system we decided to embed a world model server to store and organize a-priori and learned knowledge. The server is shared between all software processes, that have shared access to read or modify it. Our world model ontology is conceptually defined using the Web Ontology Language (OWL) standard, which is a good solution for knowledge sharing and maintainability, and the possibility of automated inference of new concepts. The knowledge encoded in the OWL format is automatically imported into our system and loaded into an Oracle database using the Redland C library. Based on rules imposed by the OWL ontology, we generate a model for the spatial 3D environment, that we call generically *world model*. This is modeled as an asymmetric, not-weighted and acyclic graph, inspired by the scene graphs used in computer games. An example of a world instance is presented in Fig. 3.

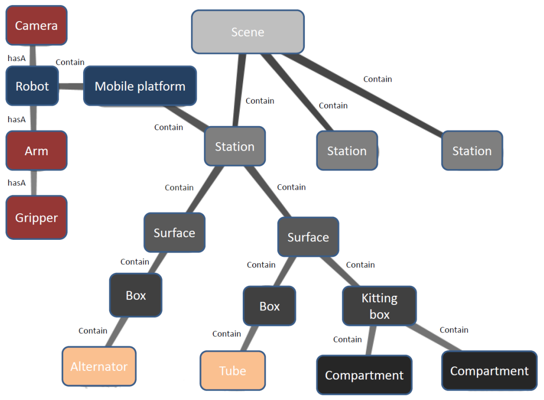


Fig. 3: Example of a world model instance

The graph nodes represent world objects, and we refer to them as *elements*. The graph arcs represent qualitative spatial relations. Qualitative means that they are not precise metric

[5]Using `pluginlib`: http://wiki.ros.org/pluginlib

positions, but instead they give a semantic specification of the position in accordance to human knowledge (e.g. *hasA*, *contains*). This is usually sufficient information for a planning algorithm and can be quickly and easily specified by a human. It is important, nevertheless, to create a mapping between qualitative definitions and more precise metric information. For a robot to grasp an object it is not sufficient to know that the object is ”over” the table, but it is required a specific geometric pose of the object in the space with respect to the robot body. Consequently, the skill requires more precise *discrete* information besides the *semantic* information. It also requires to share this information with other skills. The elements in our world model are mapped into classes and characterized by the following data: *type*, *id*, *label* and a flexible *list of attributes*. All skills operate and share information by means of this data structure. As input, the skills get one or more elements, operate on them, and update the world model accordingly. The methods to reason on the data of the elements during this process are collected in special classes called *discrete reasoners*. The role of *discrete reasoners* is two-fold: 1) convert the sensor data to data in elements and vice-versa, and 2) offer an abstract API to the semantic level in order to reason about the data. Once an element is filled up with data from a reasoner it is marked and associated to that particular reasoner in order to inform other software modules about which reasoners are possible to apply on that particular element. To summarize, the SkiROS world model server is now used in two ways:

1) maintain a world instance, to a) parameterize our skills and b) share information between the different software modules, as well as to
2) maintain a knowledge integration framework, that supports a-priori knowledge definition and teaching phases.

In the close future we will use the world instance to support the task planning. We also foresee a clear path to gradually reduce the ad-hoc code inside our skills and increase the artificial reasoning and learning, by using the world model server to manage the concepts and the modular structure idea presented in the previous section to map the concepts to software structures.

IV. Picking skill implementation

This section will go into details with explaining the processing pipeline for the picking skill (Fig. 4), integrated into the SkiROS framework explained in the previous section.

A. Part Perception and Grasping Pipeline

Referring to the overview in Fig. 4, our object perception and grasping pipeline comprises the following steps.

1) Using the respective workspace camera (to the left or right side of the robot), we detect the pallet and object candidates. If no object is found (e.g., when the pallet is cleared) the robot stops and reports to the operator.
2) The wrist camera is positioned on top of the object candidate being closest to the pallet center.
3) Using the wrist camera, we recognize and localize the part. The quality of the found matching is used for object verification. Poor matching quality indicates that a wrong object was found. In case of a wrong object, the robot stops, reports the error and waits for an operator instruction to continue its operation.
4) A grasp is selected from a set of predefined grasps and the robot plans a motion to reach it.
5) The robot grasps the object and plans a motion to an intermediate pose for for subsequent placement in the kitting box.

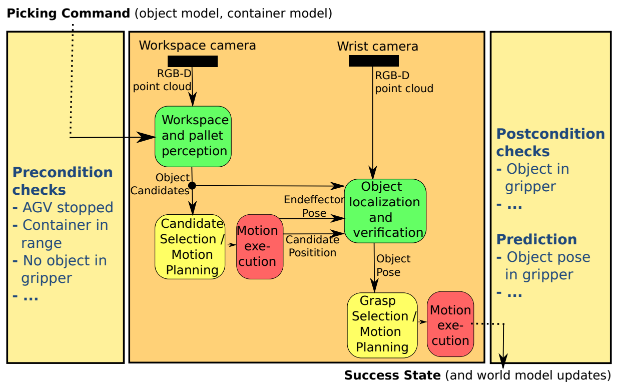


Fig. 4: Execution block with precondition and postcondition checks of the picking skill with the perception pipeline and the internal data flow.

B. Initial Part Detection

The task of picking an object from the pallet starts, respectively, when navigation has already taken place and the robot is positioned in the vicinity of the pallet. In order to compensate for potential misalignments or inaccuracies in the estimated poses of robot and pallet, we first use the workspace camera to find the pallet and to get a first estimate of where to find potential object candidates. Assuming that we know on which side of the robot the pallet is located, we acquire images of the corresponding workspace camera and search for horizontal support surfaces above the ground plane. In order to achieve real-time performance, we efficiently compute local surface normals using integral images, extract points whose normals point along the gravity vector, and fit planes perpendicular to the normals of the extracted points [28].

Referring to Fig. 5, we restrict the extracted (horizontal) planes to lie in the region where we expect to find the pallet, e.g., not outside the robot’s reachable workspace, and neglect others such as the ground plane. In order to find potential object candidates, we then select the most dominant support plane, compute both convex hull and minimum area bounding box, and select all RGB-D measurements lying within these polygons and above the extracted support plane. Thereby, we slightly shrink the limiting polygons in order to neglect measurements caused by the exterior walls of the pallet. The selected points are clustered (to obtain object candidates), and the cluster being closest to the center of the pallet is selected to be approached first.

After approaching the selected object candidate with the end effector, the same procedure is repeated with the wrist camera in order to separate potential objects from the support surface. Using the centroid of the extracted cluster as well as the main axes (as derived from principal component analysis),

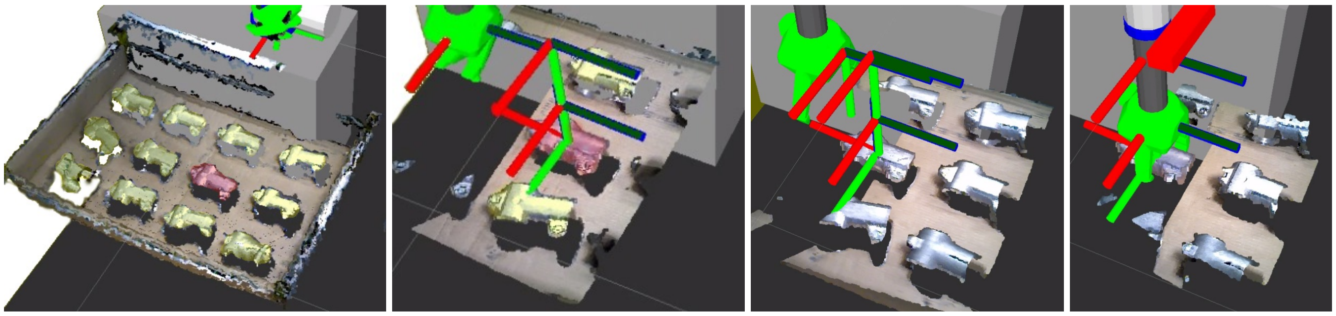


Fig. 5: Typical results of object detection and localization. From left to right: workspace camera point cloud with extracted object candidates (yellow) and selected object (red), and wrist camera point clouds during localization, approach and grasping.

we obtain a rough initial guess of the object pose. With the subsequent registration stage, it does not matter when objects are not well segmented (connected in a single cluster) or when the initial pose estimate is inaccurate.

C. *Object Pose Refinement*

The initial part detection only provides a rough estimate of the position of the object candidate. In order to accurately determine both position and orientation of the part, we apply a dense registration of a close-range depth image acquired with the wrist camera with a pre-trained model of the part. We use multi-resolution surfel maps (MRSMaps, [29]) as a concise dense representation of the RGB-D measurements on an object. In a training phase, we collect one to several views on the object whose view poses can be optimized using pose graph optimization techniques. Our pose refinement approach is closely related to our soft-assignment surfel registration approach in [30] for registering sparse 3D point clouds.

Instead of considering each point individually, we map the RGB-D image acquired by the wrist camera into an MRSMap and match surfels. This needs several orders of magnitudes less map elements for registration. Optimization of the surfel matches (and the underlying joint data-likelihood) yields the rigid 6 degree-of-freedom (DoF) transformation from scene to model, i.e., the pose of the object in the coordinate frame of the camera. The actual optimization is done through expectation-maximization [31].

D. *Object Verification*

After pose refinement, we verify that the observed segment fits to the object model for the estimated pose. We can thus find wrong registration results, e.g., if the observed object and the known object model do not match or if a wrong object has been placed on the pallet. In such cases the robot stops immediately and reports to the operator (a special requirement of the end-user).

For the actual verification, we establish surfel associations between segment and object model map, and determine the observation likelihood similar as in the object pose refinement. In addition to the surfel observation likelihood, we also consider occlusions by model surfels of the observed RGB-D image as highly unlikely. Such occlusions can be efficiently determined by projecting model surfels into the RGB-D image given the estimated alignment pose and determining the difference in depth at the projected pixel position. The resulting segment observation likelihood is compared with a baseline likelihood of observing the model MRSMap by itself, in order to avoid the calculation of the partition function of the joint data likelihood. We determine a detection confidence from the rescaled ratio of both log likelihoods thresholded between 0 and 1. For more details on the part detection and localization pipeline we refer to [32].

E. *Motion Planning and Execution*

In each pipeline cycle, we need a series of individual motions in order to localize, pick, and place an object. Since planning every single motion is a time consuming task, we use pre-computed motions whenever the arm is moved between predefined poses. Pre-computed trajectories are planned and stored only once per platform, and simply retrieved whenever the robot needs to follow the resulting motion. For being able to use pre-computed motions, we define several fixed poses, e.g., the initial pose of the robot (used while navigating with the mobile base). In addition, we define rectangular grids (Fig. 6) of poses in different heights to the sides of the robot and above the kitting boxes. For every pose in the grid, we pre-compute a joint trajectory from the initial pose to the grid pose as well as from the grid pose to an intermediate pose and the placement pose above the right compartment of the kitting box.

Whenever possible, the robot uses pre-computed trajectories to reach its goal, or approaches the closest pose reachable with pre-computed trajectories and only plans the residual motion from that pose on. By this means, we only need to plan short trajectories that are computed faster and only check if collisions exist between the robot and the environment for every pre-computed motion before execution, which is not as time consuming as motion planning. In an average picking cycle, following this scheme reduces motion planning time by more than 25 %. For both motion planning and execution we use *MoveIt!* [3]. We use standard components where available such as the standard drivers for arm and gripper as made available within the ROS Industrial initiative[6].

V. Experiments and Results

For validating our approach, we have run two series of experiments, one with the lab setup focusing on the individual

[6]For more information on ROS industrial, see http://rosindustrial.org.

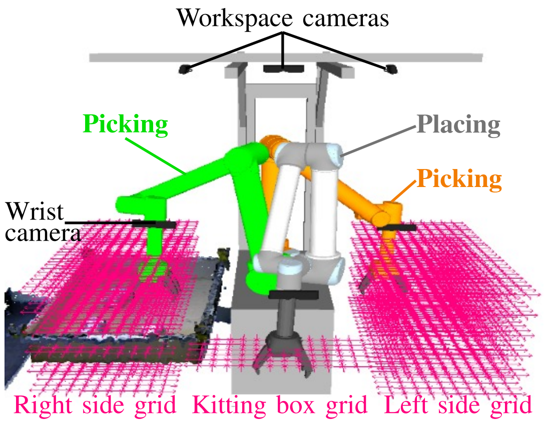


Fig. 6: Lab setup and examples of end-effector poses in the respective pose grids for picking parts to the left and right of the robot as well as placing parts in the kitting boxes.

components of the pipeline and another series with the final demonstrator platform at an industrial end-user site in order to show a proof-of-concept for the portability of the system.

A. *Performance Evaluation at the Research Site*

As evaluation criteria, we focused on the success rates, the execution times of the individual components, and the overall cycle times (for picking an object from the pallet) of the integrated system. In all experiments, we use the same platform and lab setup in order to be able to fully control the conditions under which the tests are executed, e.g., testing under different lighting conditions. The lab platform consists of a Universal Robots UR10 6 degree-of-freedom arm, a Robotiq 3-finger gripper, an ASUS RGB-D camera (workspace camera), and a PrimeSense RGB-D camera (wrist camera). Gripper and wrist camera are mounted on a linear linkage extending the robot's reachable workspace for being able to reach into corners of deeper boxes and lower layers of pallets.

In the experiments, the pallet is equipped with a total of 12 objects: 10 being the right part to pick, and two wrong objects (a very similar one and a very different one). A typical experiment setup is depicted in Fig. 7a. Instead of waiting for a particular order, the robot is repeatedly asked to pick an object from the pallet. The robot is expected to clear the pallet by grasping all (correct) objects, stop and report when it has found a wrong object, and to report when the pallet is empty. In case of failure (wrong object, empty pallet, etc.), the robot waits for commands from an operator, e.g., to continue operation. The latter is a special requirement by the industrial partner.

We present detailed results in Fig. 7. Only a single of the 100 grasps fails due to a collision. The robot reports the error and successfully grasps the object after being commanded to continue.

B. *Experiments at the Industrial End-User Site*

Lab environments can be particularly helpful for the initial research and development in order to build a prototype implementation. However, they often cannot adequately simulate the conditions of an industrial end-user site. In a final series of experiments, we applied the developed system at the industrial end-user site of PSA Peugeot Citroën. Furthermore, to provide a proof-of-concept for the portability of the system by means of the skill-based architecture, we have ported the complete pipeline to the actual robot setup as being used for automated kitting tasks at the end-user site.

Porting the system to the new platform essentially required: 1) Modeling the platform in terms of the Unified Robot Description Format (URDF), based on a CAD model of the platform and existing URDF models, e.g., of the FANUC arm. 2) Interfacing the new hardware, i.e., executing trajectories on the arm and retrieving joint states. 3) Pre-computing trajectories for the setup (including the adaptation of joint limits and pose grids for the new platform). The only problems experienced were related to the physical platform deviating from the provided CAD model. The problems could be fixed on site. We experienced no other problems.

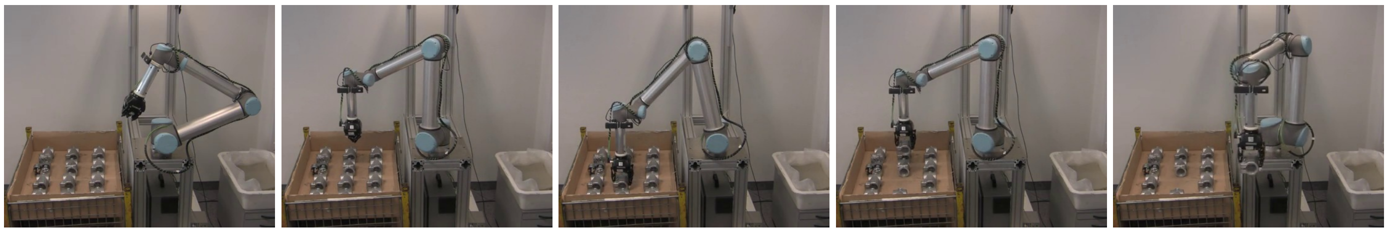
As a proof-of-concept for the overall system, we conducted integrated experiments involving both the higher level planning components and the skills for navigation, picking, and placing. The procedure of the experiments was as follows: using the higher level planning components kitting instructions were generated for retrieving compressors from two different pallets. The skill manager then called the respective skills for navigating to the pallet, picking up the part, and placing it in the kitting box. A photo sequence captured during one of these runs and the measured cycle times are presented in Fig. 8.

The measured cycle times are considerably longer than those measured in the lab environment. This is primarily caused by the main computer on the platform (an older dual core Xeon processor), which is considerably slower than the newer Intel Core i7 processor used in the lab setup. During the experiments, the computer was running at full load causing several processes to run delayed and to take longer. Moreover, for safety reasons and regulations through the system integrator and the end-user site, the arm was operated at very low velocities compared to the UR10 arm in the lab setup (15 % of the maximum speed of the FANUC arm). It can be expected that using a more decent on-board computer and relaxing the strong constraints on the maximum velocities of the arm will considerably lower both the execution times of the individual components and the overall cycle time.

VI. Conclusions

In this paper, we have presented a complete system for picking parts in the context of automating kitting tasks in the automotive industry. The system is based on a skill framework that allows easy integration and replacement of components (both software and hardware) and easy porting to other robots and setups. It features an object perception and manipulation pipeline specifically developed for efficiency. The pipeline comprises real-time pallet and initial part detection, near real-time object localization and verification, and pre-computing trajectories to drastically minimize motion planning time. Since objects and grasp poses are trained in a setup phase, the system can manipulate any object that can be detected with an RGB-D sensor, and be grasped with the used Robotiq gripper.

The performance of the system has been evaluated in terms of success rate and cycle times in a lab environment. The



(a) Photos taken during the picking experiments in the lab environment. Full videos at http://www.ais.uni-bonn.de/STAMINA.

	Execution times				Success Rate	
Component	Mean	Stdev	Min	Max	Successful / Total	
Initial part detection	26.3 ms	10.3 ms	15.2 ms	38.5 ms	120 / 120	(100 %)
Part localization & verification	532.7 ms	98.2 ms	297.0 ms	800.1 ms	100 / 100	(100 %)
Grasping a found part	7.80 s	0.56 s	6.90 s	10.12 s	99 / 100	(99 %)
Overall cycle time picking (incl. release)	34.57 s	3.01 s	29.53 s	49.52 s	99 / 100	(99 %)

(b) Execution times and success rates per component (measured over 10 complete runs, i.e., 100 grasps)

Fig. 7: Detailed results of the lab experiments: in a total of 10 runs, the robot clears the pallet each containing 10 correct parts and 2 wrong parts (a). It correctly detects the objects on the pallet and detects, localizes, and verifies parts with high success rates and low execution times (b). Wrong objects and empty pallets are correctly detected (both without false positives). Overall, we achieve cycle times for grasping objects of approx. 13 s and approx. 35 s for grasping a part, placing it in the placeholder kitting box, and returning to the initial pose.

(a) Snapshots of an integrated run with navigation, picking and placing. Full videos at http://www.ais.uni-bonn.de/STAMINA.

	Execution times			
Component	Mean	Stdev	Min	Max
Overall skill execution time picking	74.650 s	3.745 s	68.330 s	83.536 s
Overall skill execution time placing	44.325 s	1.334 s	42.559 s	46.307 s

(b) Cycle times for picking and placing. Skill execution times include moving back to the initial pose.

Fig. 8: Results of integrated runs: navigating to two pallets and, at each pallet, picking up a part and placing it in the kitting box. Shown are snapshots of a video that was captured using a GoPro attached to the upper rack (a), and the cycle times for picking and placing (b). Note that the arm was operated with only 15 % of its maximum speed for safety reasons.

system only showed a single grasp failure in 100 runs (the part was successfully grasped in a second attempt immediately after) and particularly low cycle times with all perception steps conducted in less than a second.

In a final series of experiments, the system was successfully integrated on a different mobile manipulator at an industrial end-user site. The main task here was to establish the low-level interfaces to the skill framework, after which the system could be set up exactly as for the lab demonstrator, thus demonstrating a certain degree of hardware abstraction. In the conducted experiments, the system could successfully navigate to pallets, pick up parts, and place them in the kitting box. It is a matter of ongoing and future work to extend the system to classic bin picking problems and to further improve the components on the different layers of the architecture to exploit the full potential of the vertical integration.

Acknowledgments

This project has received funding from the European Unions Seventh Framework Programme for research, technological development and demonstration under grant agreement no 610917 (STAMINA).

The authors would like to thank all project partners for the successful collaboration, integration and testing.

References

[1] K. Qian, X. Ma, X. Dai, and F. Fang, "Knowledge-enabled decision making for robotic system utilizing ambient service components," *Journal of Ambient Intelligence and Smart Systems*, vol. 6, pp. 5–19, 2014.

[2] D. Vanthienen, M. Klotzbücher, and H. Bruyninckx, "The 5C-based architectural Composition Pattern: lessons learned from re-developing the iTaSC framework for constraint-based robot programming," *Journal of Software Engineering for Robotics*, vol. 5, no. May, pp. 17–35, 2014.

[3] S. Chitta, I. A. Şucan, and S. Cousins, "MoveIt! [ROS Topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[4] R. Firby, "Adaptive execution in complex dynamic worlds," *Ph.D. Thesis*, 1989.

[5] N. Kruger, J. Piater, W. Florentin, C. Geib, R. Petrick, M. Steedman, T. Asfour, D. Kraft, D. Omr, B. Hommel, A. Agostini, D. Kragic, and J.-o. Eklundh, "A formal definition of object-action complexes and examples at different levels of the processing hierarchy," *Technical report, EU project PACO-PLUS (2009)*, pp. 1–39, 2009.

[6] B. Finkemeyer, T. Kröger, and F. M. Wahl, "Executing assembly tasks specified by manipulation primitive nets," *Advanced Robotics*, vol. 19, no. 5, pp. 591–611, 2005.

[7] A. Kirsch, "Robot learning language — integrating programming and learning for cognitive systems," *Robotics and Autonomous Systems Journal*, vol. 57, no. 9, pp. 943–954, 2009.

[8] L. Riano and T. McGinnity, "Autonomous Skills Creation and Integration in Robotics," *2012 AAAI Spring Symposium Series*, 2012.

[9] R. Fikes and N. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, 1972.

[10] M. Tenorth and M. Beetz, "KnowRob: A knowledge processing infrastructure for cognition-enabled robots," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 566–590, 2013.

[11] A. Björkelund, J. Malec, and K. Nilsson, "Knowledge for Intelligent Industrial Robots," *Proc. AAAI Spring Symposium*, 2012.

[12] M. Stenmark and J. Malec, "Knowledge-based industrial robotics," *Twelfth Scandinavian Conference on Artificial Intelligence*, 2013.

[13] A. Bjorkelund and L. Edstrom, "On the integration of skilled robot motions for productivity in manufacturing," *Proc. of IEEE International Symposium on Assembly in Manufacturing (ISAM)*, 2011.

[14] S. Blumenthal, H. Bruyninckx, W. Nowak, and E. Prassler, "A scene graph based shared 3D world model for robotic applications," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[15] C. A. Rothwell, A. Zisserman, D. Forsyth, and J. L. Mundy, "Planar object recognition using projective shape representation," *International Journal of Computer Vision*, vol. 16, pp. 57–99, 1995.

[16] K. Rahardja and A. Kosaka, "Vision-based bin-picking: recognition and localization of multiple complex objects using simple visual cues," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, vol. 3, 1996, pp. 1448–1457.

[17] R. C. Bolles and P. Horaud, "3DPO: A three-dimensional part orientation system," *Int. J. Rob. Res.*, vol. 5, no. 3, pp. 3–26, 1986.

[18] C. Papazov, S. Haddadin, S. Parusel, K. Krieger, and D. Burschka, "Rigid 3D geometry matching for grasping of known objects in cluttered scenes," *Int. Journal of Robotics Research*, vol. 31, no. 4, pp. 538–553, 2012.

[19] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2010.

[20] C. Choi, Y. Taguchi, O. Tuzel, M.-Y. Liu, and S. Ramalingam, "Voting-based pose estimation for robotic assembly using a 3D sensor," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

[21] O. Skotheim, M. Lind, P. Ystgaard, and S. Fjerdingen, "A flexible 3d object localization system for industrial part handling," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 3326–3333.

[22] A. Pretto, S. Tonello, and E. Menegatti, "Flexible 3d localization of planar objects for industrial bin-picking with monocamera vision system," in *Proc. of the IEEE Int. Conf. on Automation Science and Engineering (CASE)*, 2013, pp. 168–175.

[23] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6D object pose estimation using 3D object coordinates," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2014, pp. 536–551.

[24] D. Holz, M. Nieuwenhuisen, D. Droeschel, J. Stückler, A. Berner, J. Li, R. Klein, and S. Behnke, "Active recognition and manipulation for mobile robot bin picking," in *Gearing up and accelerating cross-fertilization between academic and industrial robotics research in Europe*, ser. Springer Tracts in Advanced Robotics, 2014, vol. 94.

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

[26] S. Bøgh, O. S. Nielsen, M. R. Pedersen, V. Krüger, and O. Madsen, "Does your robot have skills?" in *Intl. Symposium on Robotics (ISR)*, 2012.

[27] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. Schou, S. Bgh, V. Krger, and O. Madsen, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing*, 2015.

[28] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, "Real-Time Plane Segmentation using RGB-D Cameras," in *Proc. of the RoboCup Int. Symposium*, ser. Lecture Notes in Computer Science, vol. 7416. Springer, 2011, pp. 307–317.

[29] J. Stückler and S. Behnke, "Multi-resolution surfel maps for efficient dense 3d modeling and tracking," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 137–147, 2014.

[30] D. Droeschel, J. Stückler, and S. Behnke, "Local multiresolution representation for 6D motion estimation and mapping with a continuously rotating 3D laser scanner," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[31] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[32] D. Holz, A. Topalidou-Kyniazopoulou, J. Stückler, and S. Behnke, "Real-time object detection, localization and verification for fast robotic depalletizing," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.