Robot Skills for Transformable Manufacturing Systems

Pedersen, Mikkel Rath

DOI (link to publication from Publisher):
10.5278/vbn.phd.engsci.00018

Publication date:
2015

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Pedersen, M. R. (2015). Robot Skills for Transformable Manufacturing Systems. Aalborg Universitetsforlag. (Ph.d.-serien for Det Teknisk-Naturvidenskabelige Fakultet, Aalborg Universitet). DOI: 10.5278/vbn.phd.engsci.00018

ROBOT SKILLS FOR TRANSFORMABLE MANUFACTURING SYSTEMS

BY
MIKKEL RATH PEDERSEN

DISSERTATION SUBMITTED 2014

AALBORG UNIVERSITY
DENMARK

Robot Skills for Transformable Manufacturing Systems

a dissertation
presented to
the academic faculty

by

Mikkel Rath Pedersen

Robotics, Vision and Machine Intelligence Laboratory
Department of Mechanical and Manufacturing Engineering
The Faculty of Engineering and Science
Aalborg University Copenhagen
November 2014

Thesis submitted: November 27th, 2014

PhD supervisor: Professor Volker Krüger
Aalborg University Copenhagen

PhD committee: Associate Professor Manfred Jaeger, Aalborg University

Research Fellow Ron Petrick, University of Edinburgh

Associate Professor Klas Nilsson, Lund University

PhD Series: Faculty of Engineering and Science, Aalborg University

ISSN: 2246-1248
ISBN: 978-87-7112-193-3

Published by:
Aalborg University Press
Skjernvej 4A, 2nd floor
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2014

For my two girls

"There is a single light of science, and to brighten it anywhere is to brighten it everywhere."
- Isaac Asimov

Curriculum Vitae

Mikkel Rath Pedersen

Mikkel Rath Pedersen is with the Robotics, Vision and Machine Intelligence Lab (RVMI) at the Department of Mechanical and Manufacturing Engineering, at the Copenhagen campus of Aalborg University (AAU) in Denmark. During his PhD, he has been involved in three research projects funded by the European Commission, namely the two FP7 projects TAPAS and STAMINA, a well as the GISA experiment within the ECHORD project.

He holds a M.Sc.Eng. in Manufacturing Technology (2011), from Aalborg University, where he did his master thesis on the development, construction and testing of the mobile manipulator Little Helper Plus. Previous to this, he holds a B.Sc.Eng. (2009) in Industrial Engineering, where he did his bachelor thesis on injection molding of wood-polymer composites.

Committee memberships

Program Committee, IROS 2013 workshop "Cognitive Robotics Systems: Replicating Human Actions and Activities (CRS 2013)."

Reviewer

IEEE International Conference on Robotics and Automation (ICRA 2015)
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)

Professional memberships

Member of the IEEE, the IEEE Robotics and Automation Society (RAS) and the IEEE RAS Technical Committee on Mobile Manipulation.

Abstract

Nowadays, consumers are demanding customized products, contrary to mass produced ones. This demand needs to be satisfied directly at the manufacturing level, and requires transformable manufacturing systems. The pinnacle of flexibility in manufacturing is human labor; however, this is not economically viable in high-wage countries. The alternative is the advancement of robot-based automation technologies, towards robots that are flexible and easily repurposed or reconfigured. We argue that this can be achieved through the use of general, object-centered *robot skills*.

This dissertation focuses on the implementation and uses of a robot programming paradigm, focused on robot skills, that facilitates intuitive and explicit task-level programming by laymen, such as factory workers, as well as ad-hoc task planning in the skill domain.

We start by giving an overview of the uses of skills, and why the skill-based programming paradigm is an important next step in industrial robotics. We present a conceptual model of a robot skill, that incorporates both sensing and action operations, by leveraging appropriate *primitives* that perform these operations. A robot skill is parametric in its execution; able to estimate if it can be executed in a particular instant; and able to verify whether or not it was executed correctly. This means that the model assumes the important property of self-sustainability, meaning that the skill can "stand alone," as well as be concatenated with other skills to form more complex *tasks*, or robot programs.

We show how the skills can be implemented on a robot by leveraging state of the art primitives available from the robotics community, as well as ad-hoc implemented ones. Since the skills are combinations of primitives, we show that by maintaining the interfaces to these primitives within the skill, they can be transferred to different robots with relative ease. We also show that it is sufficient to implement skill-relevant world knowledge, e.g. knowledge of those objects in the scenario that can be used as input for the skills.

Since a task is a sequence of skills, using the skills for task programming is a matter of specifying both the sequence and the input parameters for each skill. We demonstrate how this can be achieved through a combination of a touchpad GUI and human gesture recognition. The sequence is created in the GUI, and the parameters are specified simply by pointing at objects to manipulate. Experiments reveal that this approach is intuitive, even for robot novices, and programming a complete transportation task can be done in as little as 2 minutes by inexperienced users.

The preprogrammed task might be sufficient for some scenarios, or even desirable when complete control over the skill sequence is necessary. However, for most tasks it is sufficient to have the robot create tasks ad hoc through task-level planning, given a desired goal. We demonstrate how the robot is capable of doing exactly this, through a formal representation of the skills in a planning language and the associated world model. In this case, a task specification simply becomes the desired setting of certain state variables, and the system plans a sequence of parameterized skills that accomplish the necessary changes in the world. Since the planning is carried out on the current state of the world, we also show that execution errors and discrepancies between the world model and the physical world can be overcome simply be replanning.

Overall, this dissertation shows that the use of robot skills is a valid solution to applying transformable robot systems in factories of the future. Through the use of general robot skills, the robot can handle a great variety of tasks in the relatively limited scenarios in the industry. Since the skills are intuitive, even for robotics novices, they can effectively be used for task programming, when reprogramming of the robot is necessary. And since the skills contain the necessary information for task planning, i.e. preconditions, predicted outcome, and skill-relevant world knowledge, task programming can even be automated, when it is desirable.

It is the firm belief of this researcher that industrial robotics need to go in a direction towards what is outlined in this dissertation, both in academia and in the industry. In order for manufacturing companies to remain competitive in Western countries, where it is not possible to compete on salary, robotics is the definite way to go – and current industrial robot systems are plainly too difficult to program for new tasks on a regular basis.

Resume

Dansk Titel: Robot-færdigheder til transformerbare produktionssystemer

Nu om dage forlanger forbrugere skræddersyede, i modsætning til masseproducerede, produkter. Det er nødvendigt at dette behov bliver dækket på produktions-niveau, hvilket kræver transformerbare produktionssystemer. Toppunktet af fleksibilitet er menneskelig arbejdskraft; dog er dette ikke en realistisk løsning ud fra en økonomisk betragtning i højtlønnede lande. Alternativet er udviklingen af robot-baserede automatiseringsteknologier, hen imod robotter der er fleksible of nemt rekonfigurerbare. Vores argument er, at dette kan opnås gennem brugen af generelle objekt-fokuserede *robot-færdigheder*.

Denne afhandling fokuserer på implementeringen og brugen af et robot-programmeringsparadigme, der er fokuseret på robot-færdigheder, og som muliggør intuitiv og eksplicit programmering på robotters opgave-niveau, *og* ad hoc planlægning af opgaver, baseret på disse færdigheder.

Vi begynder med at give et overblik over brugen af færdigheder, og hvorfor det færdigheds-baserede programmeringsparadigme er et vigtigt næste skridt inden for industriel robotteknik. Vi præsenterer en konceptuel model af en robot-færdighed, der indeholder både sansning- og handlingsprocesser, ved at udnytte passende *primitiver* der udfører disse processer. En robot-færdighed er parametrisk i dens eksekvering, i stand til at estimere om den kan eksekveres korrekt på et givent tidspunkt, og i stand til at verificere om den blev eksekveret korrekt. Dette betyder at modellen antager den vigtige egenskab at den er selvbærende, hvilket betyder at færdigheden kan "stå alene," og desuden kan kombineres med andre færdigheder, og derved danne mere komplekse *opgaver* eller robotprogrammer.

Vi viser hvordan færdighederne kan implementeres på en robot ved at udnytte avancerede primitiver, både tilgængelige fra robotsamfundet og ad hoc udviklede primitiver. Siden færdighederne er kombinationer af primitiver, viser vi at ved at bibeholde interfacene til disse primitiver inden i hver færdighed, kan de overføres til andre robotter relativt omkostningsfrit. Desuden viser vi at det er tilstrækkeligt at implementere færdigheds-relevant kendskab til verden, f.eks. kendskab til de objekter i miljøet der kan bruges som input til færdighederne.

Siden en opgave er en sekvens af færdigheder kan færdighederne bruges til at programmere opgaver ved at specificere både sekvensen af færdigheder, og deres parametre. Vi demonstrerer hvordan dette kan opnås ved at bruge en kombination af et touchpad-interface og genkendelse af operatørens positurer. Sekvensen specificeres i interfacet, og parametrene specificeres enkelt ved at pege på objekter der skal håndteres. Eksperimenter afslører at denne fremgangsmåde er intuitiv, selv for nybegyndere i robotteknik, og programmering af en komplet transport-opgave kan udføres på blot 2 minutter af uerfarne brugere.

En forudprogrammeret opgave kan være tilstrækkelig i nogle situationer, eller sågar ønskelig når komplet kontrol over sekvensen af færdigheder er nødvendig. For de fleste opgaver er det dog tilstrækkeligt at få robotten til at *planlægge* opgaven, givet et ønsket mål. Vi viser hvordan robotten er i stand til at gøre præcist dette, gennem en formulering af færdighederne i et formelt planlægningssprog og den tilknyttede kendskab til miljøet omkring robotten. I dette tilfælde bliver en opgavespecifikation udelukkende en specifikation af værdien af bestemte variable, og systemet planlægger selv sekvensen af færdigheder, og deres parametre, der udfører de nødvendige ændringer i den virkelige verden. Siden planlægningen udføres baseret på den nuværende tilstand af området omkring robotten, kan robotten også håndtere eksekveringsfejl og uoverensstemmelser mellem den modelerede og den virkelige verden, ved at lave en ny plan.

Alt i alt viser denne afhandling at brugen af robot-færdigheder er en gangbar løsning til at implementere transformerbare robotsystemer i fremtidens fabrikker. Gennem brugen af generelle robot-færdigheder kan robotten håndtere en lang række opgaver i det relativt begrænsede scenarie i industrien. Fordi færdighederne er intuitive, selv for nybegyndere i robotteknik, kan de effektivt bruges til programmering af opgaver, når robotten skal omprogrammeres. Og siden færdighederne indeholder den nødvendige information for at kunne planlægge opgaver, dvs. formelle betingelser før og efter eksekvering, samt færdigheds-relevant viden om miljøet omkring robotten, kan programmeringen af opgaver automatiseres, når det er ønskværdigt.

Det er denne forskers faste overbevisning at industriel robotteknik bliver nødt til at bevæge sig imod hovedtrækkene i denne afhandling, både inden for den akademiske verden, og den industrielle. For at produktionsselskaber kan forblive konkurrencedygtige i vestilige lande, hvor det ikke er muligt at konkurrere på lønninger, er robotteknik vejen frem. Og nuværende industrirobotter er ganske simpelt for vanskelige at omprogrammere jævnligt til at udføre nye opgaver.

Preface

The present dissertation is submitted in partial fulfillment of the requirements for the degree Doctor of Philosophy at the Faculty of Engineering and Science, Aalborg University. The dissertation is a culmination of three very interesting years of my life, work and learning, from 2011 to 2014.

This thesis has been written based on research and experiments conducted at the Robotics, Vision and Machine Intelligence (RVMI) Laboratory, at the Department of Mechanical and Manufacturing Engineering at Aalborg University Copenhagen. The work has been under excellent academic supervision by Professor Volker Krüger, Aalborg University Copenhagen.

Funding for this work was partly obtained from the Department of Mechanical and Manufacturing Engineering, Aalborg University, and the European Commission under grant agreement numbers FP7-260026 (TAPAS) and FP7-610917 (STAMINA).

Mikkel Rath Pedersen
Aalborg University, November 27, 2014

Acknowledgements

I owe a huge thanks to a lot of people who have inspired me throughout my time as a PhD student. I would like to acknowledge everyone for their support and discussions.

First and foremost I would like to thank my supervisor, Volker Krüger, for always being a great discussion partner, for constantly giving me valuable feedback on my progress and work, as well as constantly showing a very high level of commitment. Volker has been standing on the border between supervisor and advisor, always going to the side where he thought I needed him the most. Besides this, I appreciate all our discussions about whisky, music, parenthood and countless other equally non-academic topics.

A huge thanks also goes out to the rest of the team at the Robotics, Vision and Machine Intelligence Lab, and the Robotics and Automation Group. They are a joy to work with, and I wish them all the best in the future. A special thanks goes out to my friend Lazaros Nalpantidis, who has always been a great discussion partner in all matters, academic or otherwise, for being an excellent academic mentor, and for always being a really nice guy.

Aaron Bobick from the Computational Perception Laboratory at Georgia Tech functioned as my acting supervisor during my 5-months stay there. He was always extremely helpful, and always found new layers to my work that I had not noticed myself. He always left me thinking after our meetings, which I appreciate a lot.

From Georgia Tech I would also like to thank Tucker Hermans and Brian Goldfain for the great company they provided during my stay. I deeply appreciate the immediately welcoming attitude from them, and the friendship we developed over the course of only 5 months.

Finally I am deeply grateful to my fiancée Signe and our daughter Astrid for always being patient, supportive and interested in my work throughout the last 3 years. You both bring so much more joy into my life than I thought possible.

Contents

List of Figures

Thesis Details

Thesis Title: Robot Skills for Transformable Manufacturing Systems
Ph.D. Student: Mikkel Rath Pedersen
Supervisor: Professor Volker Krüger, Aalborg University

The main body of this thesis consist of the following papers:

[A] Mikkel Rath Pedersen, Lazaros Nalpantidis, Rasmus Skovgaard Andersen, Casper Schou, Simon Bøgh, Volker Krüger, Ole Madsen, "Robot Skills for Manufacturing: from Concept to Industrial Deployment," Submitted for review November 1st 2014 to *Robotics and Computer Integrated Manufacturing*.

[B] Mikkel Rath Pedersen, Lazaros Nalpantidis, Aaron Bobick, Volker Krüger, "On the Integration of Hardware-Abstracted Robot Skills for use in Industrial Scenarios," *Proceedings of the 2nd International IROS Workshop on Cognitive Robotics Systems (CRS): Replicating Human Actions and Activities* (peer reviewed), Tokyo, Japan, November 2013.

[C] Mikkel Rath Pedersen, Dennis Herzog, Volker Krüger, "Intuitive Skill-Level Programming of Industrial Handling Tasks on a Mobile Manipulator," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4523–4530, Chicago, USA, September 2014.

[D] Mikkel Rath Pedersen, Volker Krüger, "Planning of Industrial Logistic Tasks in the Robot Skill Domain," Submitted for review October 1st 2014 to *2015 IEEE International Conference on Robotics and Automation (ICRA)*.

In addition to the main papers, the following publications have also been made:

[I] Bjarne Großmann, Mikkel Rath Pedersen, Juris Klonovs, Dennis Herzog, Lazaros Nalpantidis, Volker Krüger, "Communicating unknown objects to robots through pointing gestures," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8717 LNAI, pp. 209–220, 2014.

[II] Mikkel Rath Pedersen, Volker Krüger, “Robot Skills for Teaching by Demonstration on an Industrial Robot,” Revised version submitted November 25th, 2014 in *Cognitive Robotics Systems: Concepts and Applications*. [Special issue] of *Journal of Intelligent & Robotic Systems*.

[III] Ole Madsen, Simon Bøgh, Casper Schou, Rasmus Skovgaard Andersen, Jens Skov Damgaard, Mikkel Rath Pedersen, Volker Krüger, “Integration of Mobile Manipulators in an Industrial Production,” Accepted November 10th, 2014 in *Industrial Robot: An International Journal*.

[IV] Simon Bøgh, Oluf Skov Nielsen, Mikkel Rath Pedersen, Volker Krüger, Ole Madsen, “Does your Robot have Skills?,” *Proceedings of the 43rd International Symposium on Robotics (ISR)*, Taipei, Taiwan, August 2012.

[V] Mikkel Rath Pedersen, Carsten Høilund, Volker Krüger, “Using Human Gestures and Generic Skills to Instruct a Mobile Robot Arm in a Feeder Filling Scenario,” *IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 243–248, Chengdu, China, August 2012.

This thesis has been submitted for assessment in partial fulfillment of the PhD degree. The thesis is based on the submitted or published scientific papers which are listed above. Parts of the papers are used directly or indirectly in the extended summary of the thesis. As part of the assessment, co-author statements have been made available to the assessment committee and are also available at the Faculty. The thesis is not in its present form acceptable for open publication but only in limited and closed circulation as copyright may not be ensured.

Part I.

Summary

Introduction

Nowadays, consumers are demanding customized products, contrary to mass produced ones; for instance, it is possible to completely configure and order a car in your web browser. This demand needs to be satisfied directly at the manufacturing level – which raises the need for transformable manufacturing systems that can be reconfigured, or even repurposed, when necessary. The only way to satisfy this demand is currently by having a high degree of human labor, which is not cost efficient, especially in high-wage countries. In these countries, manufacturing companies must instead rely on, and implement, complex automation equipment, such as robots. Furthermore, in nearly all production facilities there are humdrum, trivial tasks, like transportation, inspection or part feeding, that are still handled by human workers – simply because they are too difficult to automate. This project aims to do something about that.

1. Background

Manufacturing in Western countries is facing serious challenges, and has been for quite some time. Todays manufacturing paradigm is shifting from mass production to mass customization, requiring *transformable* production facilities that can rapidly accommodate changes in production volume and product variety due to volatile global markets. Hartmann [1–3] defines transformability as

- fast, active adaptation of structures to *unforeseen* changes in tasks and
- the possibility to evolve structures responding to *foreseeable* and constantly changing requirements.

Currently, the use of automation technologies such as robots can not fully satisfy this production paradigm and sustain a transformable production. The alternative is human labor, which is easily scalable and reconfigurable – however, this is a too expensive alternative in high-wage countries. As a consequence, manufacturing companies are in many cases forced to outsource the production to low-wage countries, in order to remain competitive on todays global market.

Current automation lines, although highly efficient, are usually very inflexible, due to fixed layouts of the factory and locations of manufacturing equipment, such as robot cells, conveyors, etc. Even mobile robots or AGVs[1] in the industry are relatively inflexible, in that they usually follow fixed paths or markers on the factory floor, that are difficult to change. These automation technologies are thus difficult, time-consuming and expensive to adjust with respect to production volume and product variety. Furthermore, since they require expert knowledge to integrate and maintain in the manufacturing system, start-up manufacturers, or manufacturers with low throughput, are reluctant, and in some cases even afraid, to introduce robots in their factories.

One metric for measuring the productivity of production systems is the Overall Equipment Efficiency (OEE). This 3-tier metric specifies the efficiency of a production system, with one tier focusing on availability of production equipment, calculated as planned uptime vs. unscheduled downtime of the equipment. If this metric is applied to current robot systems in factories that strives to be transformable, calculated as ideal runtime vs. programming time, it would result in a poor ratio, especially in periods where new products are introduced.

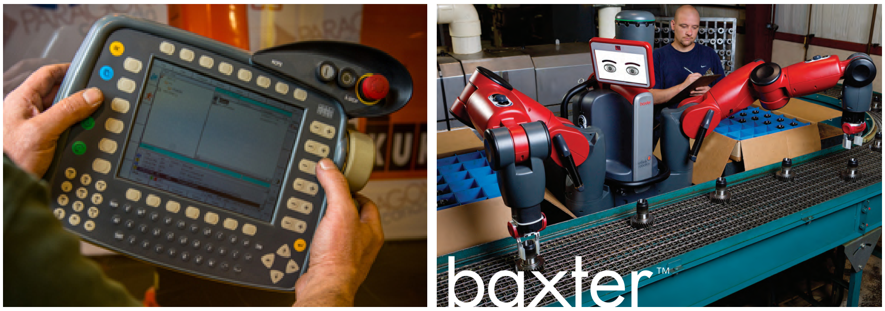
It is essential that the automation lines in factories of the future can easily be adjusted to accommodate changes in production demand. It is a clear need that an advancement of robot-based automation technologies is necessary, to enable manufacturing companies to overcome the issues presented above. This will enable production to remain in high-wage countries, while maintaining global competitiveness of the manufacturing companies. This need calls for robots that can be rapidly and effortlessly adjusted or even repurposed to solve new tasks in the factories – without having to stop the complete automation line, so an expert robot integrator can reprogram the robot. However, these robots are simply not available in todays markets.

2. Motivation

Currently, industrial robots are notoriously difficult to program. Usually the robot is equipped with a form of Teach Pendant (see Fig. 2.1a), which the expert robot integrator can use to program simple motions of the robot arm

[1]Automated Guided Vehicles

or input/output signals to external production equipment, usually in some proprietary programming language. It is only in recent years that industrial robot manufacturers have started addressing the issue of intuitive programming of industrial robots, the most prominent examples being the compliant robot arms from Universal Robots, or more recently the Baxter robot from Rethink Robotics (Fig. 2.1b). These are both examples of addressing the needs of industrial robots that are intuitive to use, and both are currently very popular, since they are somewhat satisfying the demands outlined in the previous section. However, they are still stationary robots, and it still takes significant training to use these systems.



(a) KUKA KCP2 [4] **(b)** The Baxter robot [5]

Fig. 2.1.: A standard proprietary teach pendant, the KUKA KCP2, and the Baxter Robot from Rethink Robotics.

Let us instead give a motivating example of the ideal robots for the factories of the future. Consider the scenario visualized in Fig. 2.2, showing two *mobile manipulators* working alongside factory workers. These are not typical industrial robots, performing the same sequence of motions repetitively in the same location, and requiring expert knowledge and experience to program. These particular robots are handling a variety of tasks, exactly when it is needed; tasks such as transporting parts to stations that need them, loading assembly machines and inspecting the quality of produced products. The robots are working alongside factory workers, that use the robots like any other tool at their disposal, and instruct the robots much like they would instruct a fellow worker, allowing the human workers to avoid mundane or even unhealthy tasks.

From a roboticist's point of view, there are several essential and interesting questions regarding the particular robots from the example:

- How can a robot be designed and programmed to handle such a variety

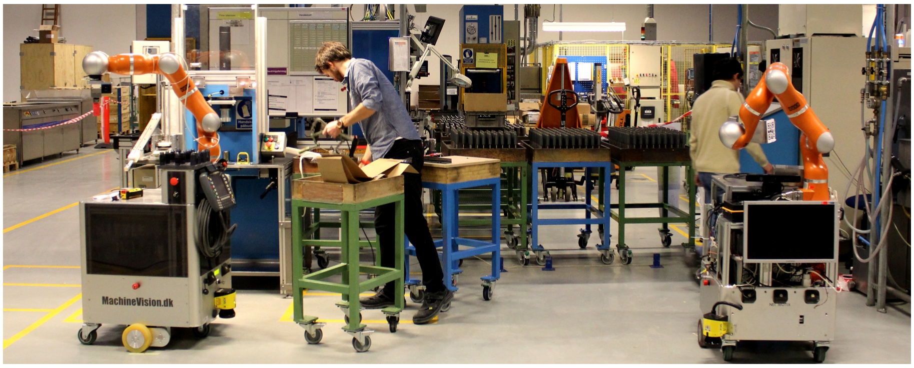


Fig. 2.2.: Two mobile manipulators working alongside factory workers

of tasks?

- How can a robot be instructed to complete new tasks, previously unknown to it?
- Who is able to instruct a robot to do these tasks?
- How is a robot able to act and react in a human-populated environment, which is prone to disturbances?

In this dissertation, we will focus on how to make it possible for laymen, inexperienced in robotics, to program tasks like the ones mentioned in the example, or to completely offload the programming effort by automated planning. This should be possible on, in principle, any robot, and not just specific robots with specific capabilities. We argue that the key to this is the integration and use of a small set of general *robot skills*, that can be concatenated to form a great variety of tasks.

Essentially, we see skills as the general building blocks that make up a robot program, where skills themselves can be compiled from simple motion and sensing operations. As skills contain sensing and motion, they are focused on *objects*, rather than e.g. 3D coordinates. This also makes them intuitive for the factory workers that will use the skill in everyday scenarios. If object-centered skills such as ***pick*** *object* or ***drive to*** *location* exists on the robot, they can be concatenated to form more complex behavior of the robot system. Complete robot programs, or *tasks*, then simply become sequences of skills, that aim to solve a given goal in the factory, e.g. *"bring pallet with part P_1 to machine A"* or *"insert part P_2 in machine C and start it"*. As an experimental platform we use a mobile manipulator built from standard industrial components; the Little Helper.

2.1. The Little Helper concept

Research in mobile manipulation has been carried out all over the world since the late 1980s, especially so during the past decade, and many prototypes have been built. Implementation of mobile manipulators in the industry is somewhat lacking, since these are all mainly prototypes, concentrated on specific scenarios, and as such have not been tested extensively in real-world industrial scenarios [6].

At Aalborg University, the mobile manipulator Little Helper (LH) has been developed in 2008 [7], and has been further improved since then. A new version, named Little Helper Plus (LHP), has been developed in 2011[2], incorporating the KUKA Light Weight Robot [8]. Both models have been tested in a real industrial scenario at a production facility at Grundfos A/S, a Danish pump manufacturer, and was one of the most advanced tests of mobile manipulators in a real industrial environment at the time [6]. The programming time for this demo was, however, very high, which emphasizes the need for robot programming paradigms that facilitate fast programming.

Some of the most advanced mobile manipulators (e.g. TUMs ROSIE, DLRs Justin, Fraunhofers Care-o-bot or Willow Garages PR2) are mainly humanoid service robots, which by far is the most prevailing research focus in mobile manipulation across the world. Although all of them are highly advanced systems, both regarding software and hardware, the problem regarding easy programming of them and interaction with them persists. This project aims towards easy programming of advanced mobile manipulator systems, so they can be useful in practical domains in the industry, and not just robotics research.

3. Related work

Skills and primitives To establish a fitting representation of robot skills for task-level programming has been the focus of research around the world [9–16]. The various concepts of skills is quite different, but most of them are composed of primitive, formal descriptions of sets of robot motions, called action or motion *primitives*. These primitives are simple, atomic robot movements that can be combined to form more complex behavior [17–22], which is often called a robot *skill*. As such, no single, unified definition of a skill in terms of robotics exists. Furthermore, none of these skill representations are focused on being easy to use and understand for laymen, but rather for experts in the

[2]Incidentally, by this researcher

field of robotics. We argue that this is an important aspect of a robot skill for industrial purposes, and pursue this argument in this dissertation.

The *primitives* that make up a skill [13, 14, 16] are rather well defined in the robotics community. Although several different descriptions exist, most of them loosely follow Mason's work on compliant controllers [23], which paved the ground for the Task Frame Formalism (TFF) introduced by Bruyninckx and De Schutter [24, 25]. Recent work has expanded upon this idea, for instance enabling the use of any sensor and controller in the same task frame (e.g. visual servoing combined with force control), and operating in a dynamic task frame [26–28].

Most recent work on motion primitives seem to utilize a variation of the Dynamic Movement Primitives (DMPs) described in [29–31]. The DMPs specify a single movement, and represents the desired kinematic state of a robot limb (or alternatively in the task space) as a mixture of non-linear differential equations. The DMPs have mainly been used for mapping human to robot motion in research in Teaching by Demonstration.

The concept of skills presented in this paper is comparable to the *manipulation primitive nets*, described best in [27]. These nets are sequences of manipulation primitives (MPs), with simple decisions based on the outcome of each single MP in the net. In these nets, however, each MP needs its own explicitly specified parameters, e.g. a specific force or velocity in a specific direction. This makes this particular implementation unsuitable for robotics novices in a factory hall. Instead, we propose a method for specifying parameters on a higher level, the skill level, and instead letting the system infer the parameters for the lower level primitives.

Another relevant skill-concept is the Action Recipes of RoboEarth [32, 33]. However, in RoboEarth, the skills (or *Action Recipes*) keep sensing and action separate. With its focus on knowledge sharing between robots, the Recipes (robot programs) in RoboEarth rather require some capabilities of the robot, such as object recognition and environment models, and the Recipes are only action based. There are additional features that this approach lacks; a formal description of which conditions the program will work under, i.e. preconditions, and a description of the effects of executing the program, i.e. the postconditions.

A solution to this could be to associate the program with a skill description template, as in SKORP [34] – an approach to skill-based robot programming using a graphical representation of skills, where the skill template serves as a communication tool between the skill developer and the task programmer. However, the pre- and postconditions need to be connected to the real robot, i.e. to be testable before they actually justify their existence.

Common for all the previous work, is that they have all, in one way or another, been insufficient to deploy in a real industrial setting, and require expert knowledge of the robot system to use and implement. This work will focus on the formulation of skills, and the use of these for task programming by laymen in industrial environments.

Human-Robot Interaction (HRI) for Industrial Robots HRI is a field, which, for industrial applications, is heavily influenced by tradition. As such, nearly all industrial robots are programmed in a native robot programming language, on a small teach pendant attached directly to the robot controller, with specific commands for motion, input/output etc.

The use of robot skills for intuitive programming in the industrial context is, quite interestingly, not gaining much attention from the research community. When it comes to programming of industrial robots, research focus seems to be on offline programming by using CAD models [35–37] or online programming using augmented reality (AR) [38–40]. Both of these approaches, however, do not address the issue of transformable robots, but rather methods for creating traditional robot programs that perform repetitive motions with limited sensing.

One method is to create a graphical programming interface (GPI) for the operator, where he/she can drag and drop predefined skills into a sequence, and specify the parameters for them. This has been carried out on an experimental state as part of the PACO-PLUS project [41], and will also be carried out in this project in a similar way.

Researchers have also combined skill primitives and speech recognition for programming grasping tasks by speech [42], with other work presented in [43, 44]. This work incorporates task programming using a combination of human tracking, a user interface, speech recognition and AR to program a packaging robot. However, the programming is a combination of primitives, with no higher-level skills between these and the task layer, and it is yet to be shown how well this system performs in real industrial scenarios.

To advance the field of HRI in the industrial context requires in part intuitive interfaces and methods for programming the specific skills, as well as for task-level programming. The problem is to design the interface so it is intuitive for the user, i.e. an engineer/specialist for skill-level programming and an operator for task-level programming.

Task Planning Research on task-planning has been around since industrial robots were first introduced - in fact it was not that long after the introduction of industrial robots that researchers realized this would be a necessary next step in robotics. The most famous of the earliest attempts on task planning was using the STRIPS planner [45], and many other algorithms has followed, based on the same ideas, in that they act on a set of actions that alters the current world state [46, 47]. This includes breaking down assembly tasks into some form of manipulation primitives the robot can interpret [48].

Task planning often uses motion primitives as the planning domain [46, 49, 50], where the focus seems to be on representing capabilities, or primitives, of the robot so they can be used for planning. The system then tries to plan a complete task from the primitives.

A layered structure of skills and tasks similar to ours is also proposed in [51]. Focus is still planning on the task level, and by utilizing somewhat general skills and skill primitives. The implementation of said skills is however at its current state rather simplistic.

The problem with modeling the world state, and maintaining this model, persists in these cases. One approach was to use logic programming to describe available robot actions and their results, using the situation calculus [52, 53]. In the situation calculus, the world state is only dependent on an initial state and the actions performed by the robot. This approach did however lack the interfacing to the actual robot, something GOLEX [54] overcame, although for a very simple scenario. Recent work however suggests to use logic programming for more advanced scenarios, somewhat resembling skills [55].

In this work have planning in mind from the beginning, which will result in a skill formalization and world knowledge that will immediately be usable for planning.

4. Research objectives

The main objective of this PhD project is to investigate intuitive methods of task-level programming and planning on a mobile manipulator. The approach to do this is by utilizing a set of general robot skills, which can be concatenated to form tasks, or complete robot programs.

The first part of this objective is to formulate and implement the skills that allows intuitive programming and task planning. These skills need to unify sensing and action - no specifications of coordinate frames or controller settings should be necessary, as this is not easily understandable by people inexperienced in robotics. Therefore, skills should not merely be macros or functions, that perform some operation on the robot system based on numeric input. Instead, the skill must internally handle the necessary computations that allow the skill to be applied on *objects* instead.

Secondly, a clear part of the objective is to investigate methods for robot programming by using *a)* a set or sequence of motion primitives to create skills, and *b)* the robot skills to create tasks, where the main emphasis will be on the latter. The goal is to provide an intuitive method for specifying the sequence of skills, and their parameters, to solve a task in a factory. The keyword here is *intuitive*, so that the tasks can be programmed by unskilled shop-floor workers.

The final part of the objective is to use the skills for task planning. Since humans are programming the skills, we are effectively abstracting away the lower level aspects of traditional task-level planning (which is usually done on some formal representation of motion primitives). This results in a lower dimension of the planning space, since we are doing task-level planning on the (human-specified) skills instead. Thus, we expect that task-level planning could be implemented in a useful manner, if the planning is done on the skills, which is also to be investigated.

The above aspects of the objective can be summarized in the following scientifically important research question:

> *How can a robot programming paradigm focused on general, object-centered robot skills be modeled, implemented and used for intuitive task-level programming* and *ad-hoc task-level planning of full robot programs/tasks?*

The research question can be expanded into the following intertwined research objectives:

- Determine the appropriate abstraction level of the implementation of skills, that facilitates both an interface between the low-level robot motion primitives, and the higher levels of task planning and programming, error handling and ease of use.

- Investigate methods of interacting with the robot during task programming, i.e. intuitive specification of new tasks, utilizing the robot skills.

- Describe and maintain the skill-centered world model in the robot, i.e. in a manner that facilitates planning on the skill level.

- Investigate the use of skills for task-level planning, based on the above world model, so that only a desired goal setting of the world state is necessary to specify a task to the robot.

In approaching the above objectives, we are not only interested in whether or not the proposed methods work, but also how well they perform. Therefore, we ask the additional scientifically important questions:

- To what extent can the programming of robot skills be simplified, in order to easily expand the usability of the robot system?
- To what extent can a finite sequence of parametric skills be used to describe a robot program, i.e. a task?
- To what extent is it possible to automate the programming of tasks on the robot?

5. Research methodology

With respect to the project objectives, several steps need to be carried out, many of which are highly focused on implementation, as is often the case in robotics. Implementation will be carried out bottom-up, i.e. by starting with a clear formalization but simple implementation of the skills, composed of lower-level primitives. These skills will later be expanded to include the capabilities that are necessary for task-level programming and planning. This leads to an overall sequential work methodology, yet iterative in the form of expanding the findings in each point as the complexity of the problem increases. The work methodology is visualized in Fig. 5.1.

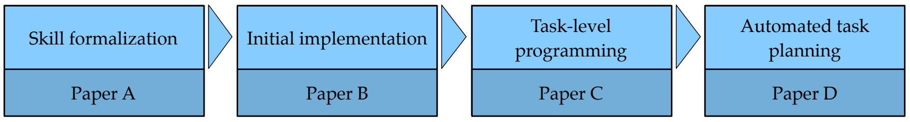


Fig. 5.1.: Progression of work during the project. Lighter colors show the order of addressed topics, and darker colors show the output. After the initial skill implementation, the implementation is revised and updated in each of the following blocks.

The skill formalization should already from the beginning contain the aspects that make it possible to use them for task programming and planning. However, the *implementation* of skills on the robot will be expanded iteratively, to include these aspects when they are needed in the process. This includes pre- and postconditions, but also intuitive parameter inputs.

At first, it will be sufficient to make a simple implementation of skills, such as a pick and place task without complex contact mechanisms in the lab. This will provide a good starting point for learning how to define parametrized skills and primitives and to implement them on the robot. Later this can be expanded to more advanced cases, such as serving a feeder, which is essentially a pick and place task, where the robot has to drive between two locations. Finally, the implementation of the necessary skills to solve a majority of logistic tasks can be tested. This will be done as a mock-up in the lab, prior to experiments at a real production line.

Experiments will need to be carried out in the lab, in order to ensure they work in a controlled mock-up of the scenario at hand. This is also beneficial in revealing aspects that are not thought of beforehand, but are required for the function of the skills. However, it is crucial that the skills are verified and tested in real environments. As part of the project, actual demonstrations at real production facilities are also carried out, which will test the setup under realistic conditions.

Summary of contributions

This chapter summarizes the papers that make up the main body of the thesis. In each section we will summarize the main contributions of the specific paper, and highlight the process that connects it to the following papers. Finally, we will give a short summary of the principal results of the body of work as a whole.

6. Paper A

Robot Skills for Manufacturing: from Concept to Industrial Deployment

In this work we thoroughly introduce the reader to the problem at hand, and why we believe robot skills is part of the solution to this problem. The reader is introduced to the skill concept, and how skills can be used to construct higher level tasks. The paper summarizes most of our experiments with skills and task programming, tested both in lab settings and real production facilities. The key points in this paper is the motivation for using skills, and the introduction of the skill model and architecture of primitives, skills and tasks.

We summarize the current challenge in manufacturing; to have a production facility that is highly responsive to change in product variety or volume. We argue that having a robot fleet equipped with general skills is the answer to this challenge. Skill-equipped robots can handle a variety of tasks that are currently performed by humans, simply because they are too difficult to automate. This conclusion is reached from an analysis of task instructions given to the factory workers. These are in the form of Standard Operating Procedures (SOPs), and we show that all of these SOPs can be decomposed into sequences of 13 reoccurring, object-centered operations, which we call skills.

A conceptual model of a *robot* skill is then presented, showing a skill that is parametrizable, and changes the internal representation of the state of the world, either through sensing or action. Skills need to contain both sensing and action in order to be applied on abstract parameters like objects, rather than 3D coordinates. Furthermore, the skill model contains both a formal description of preconditions and predicted outcome, and methods that verify these, either based on the world model or ad hoc sensing. This is important both for robustness, and to allow the future extension of task planning on the skill domain.

The conceptual model is shown in Fig. 6.1. This particular model is more in-depth than the models included in Papers B, C and D, and is the final version of the *visual* representation of a robot skill, incorporating all of the above aspects.

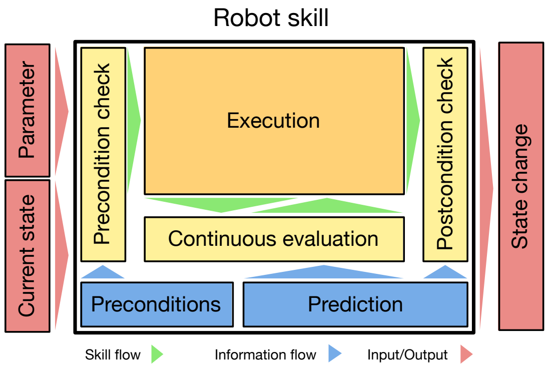


Fig. 6.1.: Conceptual model of a robot skill. The preconditions and prediction (blue blocks) are informative aspects of the skill. The checking procedures (yellow) verifies the informative aspects before, during and after execution (orange). Execution is based on the input parameter and the world state (red), and the skill effectuates a change in the world state (red).

We present a three-layered architecture for a skill-equipped robot, containing a *primitive*, *skill* and *task* layer. Primitives are the fundamental sensing or motion operations that the robot can perform – such as opening or closing a gripper or detecting objects in the current field of view. Skills are composed of these primitives. Tasks are sequences of skills, that aim to solve a specific goal in the factory, and are composed of skills, either through direct task programming or through task planning, both of which is the focus of the following work.

We show several experiments with skills as self-contained components, as well as experiments with task programming using e.g. kinesthetic or gesture-based teaching. The experiments are presented in summary form, to show the wide array of use cases where the skills can be used. These experiments verify the generality of skills, that they can in fact be used for a variety of tasks, and are useful for task programming by laymen.

Furthermore, we show an array of use cases in real production environments, where complete robot systems are integrated in running production lines, using the implementation of skills and task programming methods outlined in the previous experiments.

Progression of work The purpose of this paper has been to formally introduce the background and rationale of the skill concept to the wider academic community. The paper thoroughly explains the skill concept, and an architecture that permits task-level programming using these skills. The experimental section is in the form of highlights of a range of experiments, including highlights of the following work. In the following we will give additional details on the work carried out within skill implementation, task programming and task-level planning, as well as present more advanced experiments in these areas of the project.

7. Paper B

On the Integration of Hardware-Abstracted Robot Skills for use in Industrial Scenarios

This paper presents the initial skill implementation, according to the skill concept. The purpose of the work is to investigate the implementation of skills, following the model presented in the previous paper, as well as a skill-centered world model, and hardware abstraction of robot skills.

The focus is on hardware transferability, and we demonstrate how two skills implemented on one robot, a PR2, can be transferred with relative ease to a very different robot, the Little Helper. The skill implementation contains world knowledge, pre- and postcondition checks, as well as object-centered execution of the skill. What is still lacking in this work is the prediction of the skills. The two robots used for experiments are shown in Figure 7.1.

We have implemented the ***pick up*** *object* and ***place at*** *table* (x, y)*-position* skills. These are very complex skills, in that they require a high degree of both sensing and motion to be sufficiently general, and we give a detailed outline of the contents of these important skills.

The implementation leverages sensing primitives readily available from the robotics community, that can detect common household objects, and shows a skill-centered, database-like world model, containing previous observations of these objects. Furthermore, we have implemented a Task-Frame Formalism (TFF) [24, 25] controller primitive for accurate motions with contact-stop mechanisms for the placing skill. This particular primitive is only running on the PR2 robot, and we go on to demonstrate how the skills regardless can be transferred to a different robot.

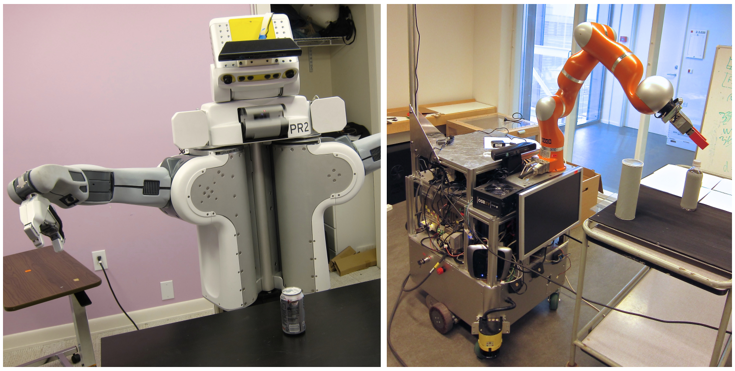


Fig. 7.1.: The two robots used for experiments: (a) the PR2 robot and (b) the Little Helper.

By implementing the same skill primitives on both robots, the skills are directly transferable. However, we have found that this is not always possible, as different robots have different capabilities, as was the case in these experiments. Instead, we have maintained the *interfaces* to the skill primitives. For instance, for controlling robot arm movements, we simply give the corresponding primitive a desired pose of the gripper. The PR2 then uses motion planning to find a collision-free joint trajectory that moves the arm into the desired position, while the Little Helper system simply relays the coordinate to the robot controller, which essentially performs a blind movement of the robot arm.

The paper shows that skills can be implemented with relative ease on a robot that already has the necessary primitives. This was the case for the PR2 robot, where drivers for all mechanisms on the robot are readily available. Since the initial work on the PR2 was related to the skill implementation itself, little attention was paid to hardware transferability. However, since the fundamental primitives are the same (e.g. moving the arm or opening the gripper), and require the same parameters (e.g. a joint/Cartesian pose or a desired grasp width), transferring the skills to a completely different robot like the Little Helper proved effortless.

Progression of work This work was highly focused on getting the skills to work, and has showed how skills can be implemented. In fact, the method by which the skills were implemented in this work forms the basis of the skills used in all the following work. There are, however, some key differences. First and foremost, since the focus of the following work is to investigate the use of skills for task programming, we restrict the generality, and thus complexity, of the skills. The reason for this is simple – we want skills that are highly robust. The second is that, out of necessity of having readable code, the skills have a more well-defined interface to the primitives. Since this is not the focus of the following work, however, this will not be discussed further in this dissertation.

8. Paper C

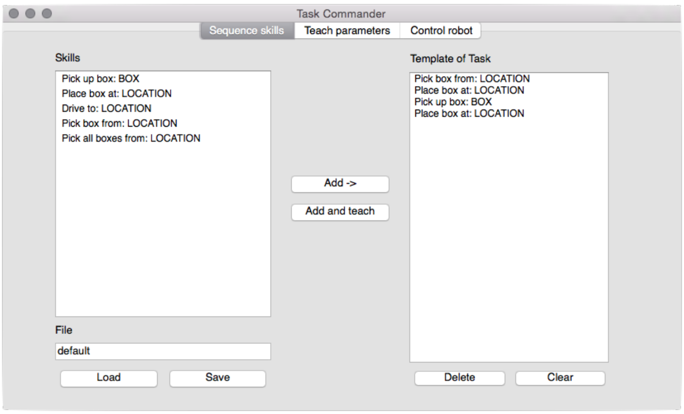
Intuitive Skill-Level Programming of Industrial Handling Tasks on a Mobile Manipulator

In this paper, we present an intuitive method for task programming using a combination of gestures and a touchpad-based graphical user interface. The purpose of this is to investigate the use of skills for direct task programming, and how intuitive this approach is for people inexperienced in robotics.

A total of three skills has been implemented, following the implementation method outline in Paper B; ***pick up*** *object*, ***place at*** *location* and ***drive to*** *location*. The implementation of the skills are not the focus of this work, and we restrict the skill to manipulate small-load-carriers (SLCs), where the SLCs and valid placing locations are marked by QR codes. The world model is a similar skill-centric world model as in the previous paper, now containing previous detections of QR codes. Whenever the robot detects a QR code, its location and data is stored, or updated if the QR code has been previously detected near the prior coordinate. We add advanced querying to the world model, such as querying objects within a specific area, or of a specific type. This ensures a robust and simple world model and set of skills, so the focus can remain on the task programming method.

The task programming is divided into two steps; *sequencing* and *teaching*. In the sequencing step, the outline of the task is constructed offline, on the touchpad. The user has access to a list of the skills available on the robot, and can drag and drop skills into the desired task sequence. The sequencing tab in the GUI is shown in Fig. 8.1a. The teaching phase takes place online, where the robot recognizes distinct gestures performed by the task programmer. This makes it possible to specify parameters for each skill in the task sequence, by simply pointing at objects or locations.

The teaching phase is modeled by a finite state machine (FSM), where the performed gesture determines the state transition. During teaching, the user has the possibility to cancel or acknowledge saved parameters, simply by performing specific gestures. Furthermore, the user can instruct the robot to follow him/her around the environment, when necessary. During teaching, the user is given feedback on the touchpad, showing the available gestures and the current state of the teaching FSM (see Fig. 8.1b), as well as on a monitor mounted on the robot, showing the status of gesture recognition.

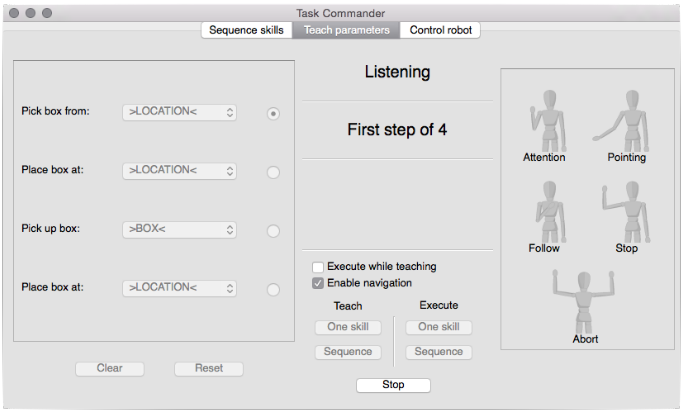


Skills
Pick up box: BOX
Place box at: LOCATION
Drive to: LOCATION
Pick box from: LOCATION
Pick all boxes from: LOCATION

Template of Task
Pick box from: LOCATION
Place box at: LOCATION
Pick up box: BOX
Place box at: LOCATION

File: default

(a)



Pick box from: >LOCATION<
Place box at: >LOCATION<
Pick up box: >BOX<
Place box at: >LOCATION<

Listening

First step of 4

☐ Execute while teaching
☑ Enable navigation

(b)

Fig. 8.1.: Two of the GUIs available to the task programmer: (a) The *sequencing* tab of the touchpad GUI, where skills are dragged to the desired sequence in the task, and (b) the *teaching* tab, where the user is given feedback on the current state and available gestures.

The approach has been verified through experiments with a total of 17 people, in two different environments. All users received a 10 minute introduction to the complete system, and programmed a task of moving SLCs between two discrete locations – a mock-up of a machine tending task, where an SLC is filled with parts and needs to be exchanged with an empty one.

The sequencing step was immediately intuitive for all participants, showing that the skills themselves are also intuitive. The teaching phase was performed three times by all participants, to investigate the learning curve of the approach. The teaching of the parameters, including navigating the robot to the different locations, by having it follow the participant, was equally intuitive. The teaching phase only took between 2 and 7.5 minutes across participants, the mean time of all attempts being a little less than 3 minutes. Teaching was notably reduced on the third attempt, showing a low learning curve for the system. Furthermore, there were no major difference in programming time between male/female participants or participants experienced/inexperienced in robotics. The sequencing time and teaching times across three attempts for all participants are shown in Figure 8.2.

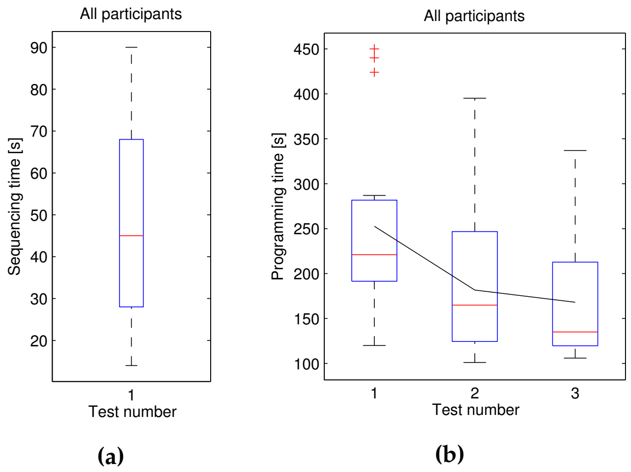


Fig. 8.2.: Box charts showing sequencing time (a) and teaching times (b) for all participants. The black curve in (b) shows the mean teaching time for each attempt.

Progression of work This paper investigated how skills can be used to directly program a task on a skill-equipped robot. This might be sufficient for robots working in smaller, semi-static environments, and perhaps even desirable for tasks where the sequence of skills and their parameters is always the same. But what about unstructured environments? What happens in the case of only minor task-level disturbances or variability, when applying the approach outlined above, e.g. when there is suddenly two SLCs at a location instead of one, or no SLCs at all? The explicitly defined task is simply unable to deal with this, as the necessary pre- or postconditions will not be satisfied.

The following work aims to overcome this problem by introducing task planning, based on the skills. This does not only have the benefit of being able to deal with disturbances. It also allows the specification, or programming, of a task to be limited to specifying the desired goal state, where the exact sequence of skills and their parameters is determined online.

9. Paper D

Planning of Industrial Logistic Tasks in the Robot Skill Domain

This paper connects the execution of individual skills in a task with online task planning, enabling the task programming to be limited to specifying a desired goal setting of the (partial) environment. This enables much easier and faster specification of a task, and the possibility for the robot to handle disturbances at runtime, simply by replanning.

The key to ad hoc, online task planning on the skill level is not the skills themselves. Since the skills already contains STRIPS-like information, i.e. input parameters, preconditions and effects on the world, they are immediately useful for planning. However, for *online* planning, the robot needs a model of the state of the world, and to concurrently update this model. Furthermore, it needs to be possible to translate this world model into a formal language like PDDL[3].

We expand the scenario from the previous paper, to include a mock-up of a feeder at a production line, and several distinct locations. We add the ability of the robot to empty an SLC in such a feeder, which is the ***unload in*** *container* skill. Furthermore, we attach a small rack on the chassis of the robot, where two SLCs can be placed at any time, enabling the robot to carry a total of three SLCs (including one in the gripper). SLCs can be picked from the rack or placed on it using the same two skills as for any other location. The scenario is visualized in Fig. 9.1, and it should be clear that explicit programming of advanced tasks in this scenario is no longer feasible.

Since we have already equipped the robot with *a)* a model of the world, described in Papers B and C, and *b)* a set of skills, that loosely follow the STRIPS formulation, we have the ingredients for applying online task-planning. In order to specify the planning domain, we manually formulate the 4 skills in PDDL, by specifying their valid input parameters, and preconditions and ef-

[3]Planning Domain Definition Language; the de facto language for defining planning domains and problems

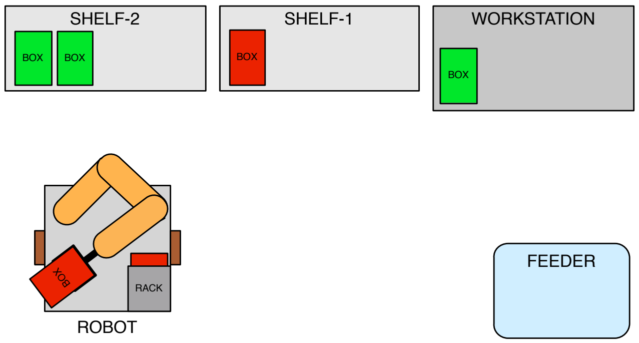


Fig. 9.1.: Example logistic task of sorting boxes and tending to a feeder for a production line. Parts are produced at the workstation and placed in boxes. Full boxes are green and empty boxes are red. The robot can carry 1 box in the gripper, and 2 in a rack mounted on the robot.

fects each expressed by a set of predicates, also formulated in PDDL. Only 9 predicates are necessary to model the (discretized) state of the world and the internal state of the robot.

By utilizing three core components, we can now perform online task planning and execution. The first component is the *world model parser*, which translates the current, continuous world model and a manually defined desired goal state into a PDDL problem specification. This problem is sent to a *planner*, along with the planning domain, which is the set of skills mentioned above. The planner outputs a sequence of skills and their exact parameters, which transfers the current world state to the desired goal state – if a valid plan can be found. The final component is responsible for making this happen on the robot. This is the *plan execution monitor*, which handles the initialization of each skill with the correct parameters, and in the right order. Furthermore, the plan execution monitor will, in the case of errors, request a new plan from the planner, and execute this instead. Finally, it verifies that the desired goal state is satisfied after the last skill is executed.

We have tested the above approach by passing different goals to the robot, always in the same initial state in a mock-up environment of a section of a production line. The environment is shown in Figure 9.2. Furthermore, we have introduced a number of disturbances during execution, such as removing an SLC from the gripper when the robot is driving between locations – in all cases the disturbances can be handled by simple replanning, as the world model is updated continuously by the skills.

The experiments reveal that planning using skills is possible based on the

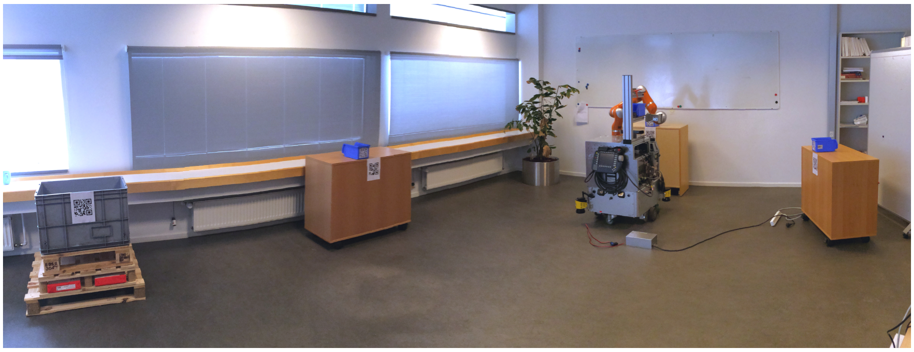


Fig. 9.2.: Environment used for planning experiments, with the feeder on the left, the robot in front of the workstation, the two shelves and four boxes.

current world state. It is also obvious that this can be an effective way to instruct tasks to the robot, be it by a factory worker or from the MES[4] at the factory. Finally, both physical disturbances and discrepancies between the world model and physical world can in most cases be overcome by simple replanning.

10. Summary of results

The previous sections outline the approach of formalizing, implementing, and using skills for task-level programming and planning. In Paper A we introduce the reader to the skill concept, and the rationale behind this. We show conceptual examples on the uses of skills for industrial robots, including skill implementation and the use of skills for task programming. Furthermore, we show several deployments of skill-equipped robot systems in running production facilities.

The initial implementation of skills is presented in Paper B, where we also demonstrate how the skills can be transferred between different robots. This implementation concerns only two skills, and includes simple skill-centered world knowledge, which is sufficient for the skills to function.

In Paper C we expand the skill library slightly, and demonstrate an intuitive approach for direct task programming, using the skills as building blocks, and a combination of a touchpad GUI and human gestures to specify their sequence and parameters. This makes it possible for laymen to instruct even

[4]Manufacturing Execution System – central real-time system that relays information of the current state of a production facility

complex tasks to the robot system, simply by pointing at e.g. objects to manipulate.

Finally, in Paper D we argue that in some cases it is not desirable to completely specify the task, but rather specify the desired goal setting. We show how the skills can be used for automatic task planning, execution and error handling by replanning, by utilizing the skills' STRIPS-like information, and the associated skill-centered world model.

In Paper A, we argue that the skills effectuate a change in the world state, either through sensing or action, or a combination of the two. It is thus necessary for the robot to maintain a representation of the world state. In all the work, we show that a skill-centered world model is sufficient. In fact, in Papers B, C and D, the same ad hoc, database like world model is used, with only minor variations. A skill-centered world model contain only the information that is directly necessary for the skills to function, and depends on the environment where the robot is deployed. In a fully deployed robot system in a factory, the skill-centered world model would thus contain information about the objects which the robot can manipulate in the factory, as well as information about the factory layout.

Conclusion

We will now discuss the work presented in the previous chapter and the included papers, as well as conclude on the contribution of the dissertation and known limitations. We will describe the impact of the dissertation on the academic and industrial community, as well as outline the contribution. Finally, we will give recommendations on the future work needed in this particular field of research, as a motivator for both the academic community and for this researcher.

11. Discussion

Referring to Section 1, we will now discuss how the presented challenges in the industry today can be overcome, in the light of the research presented in the included papers.

We have shown a skill model that is sufficiently general to be used on a variety of robots, providing an effective abstraction layer between the capabilities, or primitives, of the robot, and higher-level tasks. The initial skill implementations are also somewhat general, in that they can handle a variety of objects from a database. This generality, and specifically object-centeredness, is exactly the foundation for introducing robots that are easy to reprogram for unskilled workers. The main reason for this is that skills are object-centered. If you tell almost any person: "Here is a robot – it has a skill called '***pick up** object*', and you only have to tell it which object to pick up," that person will immediately have an idea of what the robot will do with that skill.

Since the skills are so intuitive, they can easily be used as building blocks to construct complete robot programs, or tasks. This means that the robots can be reprogrammed on the fly, by the shop floor workers. Industrial robots in the future factories will no longer have to be programmed to do simple, blind movements by an expert systems integrator – often stopping the entire section of the production line where the robot is working. Robot experts can instead focus on the robot skill level, by improving or adding to the available skills of the entire robot fleet, and offload the task-level programming to the shop floor workers.

In some cases, it is not even necessary to program the task, but instead have the robot plan the sequence of skills that accomplishes a desired goal setting. As the goal setting is usually related to the surrounding factory environment, the goal could even be specified by the central Manufacturing Execution System. This will greatly offload the factory workers to more meaningful tasks, such as maintenance of the production equipment (including the robot fleet). However, there will still be some cases where a task needs to be programmed manually, when a desired skill sequence is essential, and where the planner is not able to deal with this.

It is the firm belief of this researcher that industrial robotics need to go in a direction towards what has been outline above, both in academia and in the industry. Considering the high ratio of programming time vs. uptime for current industrial robots, the methods proposed in this paper can help to reduce this notably. In order for manufacturing companies to remain competitive in Western countries, where it is not possible to compete on salary, robotics is the definite way to go – and current industrial robot systems are plainly too difficult to program for new tasks on a regular basis.

11.1. Known limitations

We realize of course that there are a number of limitations with the work presented in this dissertation. First and foremost, we are abstracting the generality of skills away in both Papers C and D, by having a set of skills that can only manipulate a single object type. This essentially boils down to the problems of object recognition and grasp detection – both very active research areas in the robotics community. However, due to the modular nature of the skills, this can easily be added for the objects that are relevant for a particular robot, while preserving the task programming and planning capabilities presented in these papers. Even the more general skill implementation presented in Paper B requires a database of objects with associated valid grasp poses. For a robot deployed in a factory setting, the same approach might be valid, since information of the product or parts to manipulate is readily available, in most cases with a finite feature variation.

There is also the issue of scalability to different task domains. In this dissertation we have focused the experiments on logistic tasks, such as transportation and part feeding. However, what if the skills were to be applied on domains requiring more complex object interactions, such as assembly? We argue that, again due to the modular structure of the skills, this can be achieved by incorporating more advanced primitives, from which the skills are composed. At the highest level of generality, this would result in e.g. assembly skills like ***place in*** *sub-assembly-hole-3*, where the necessary primitives are executed to place whatever is in the gripper in a hole in some subassembly of a final product. However, this might not be feasible from an integration viewpoint, as there is the risk that skills will become unnecessarily complex. Instead, one could easily imagine skills on different layers, where e.g. ***insert in hole*** or ***screw in*** are abstractions of the ***place*** skill.

12. Conclusion

Relating to the objectives presented in Section 4, we will now outline the contribution of this dissertation.

We have presented an intuitive conceptual model of general, robust and object-centered robot skills. This model is an abstraction of lower-level capabilities of the robot. We show that by implementing skills according to this model, composed of primitive sensing and action operations, it is indeed possible to apply the skills on objects, rather than e.g. 3D locations. This enables the skills to function as a middle layer between e.g. traditional robot motion commands and complete robot programs, or tasks.

Due to the intuitiveness of the skills, they can be used for task-level programming, even by laymen. We have demonstrated this through the use of simple methods for specifying the sequence of skills that make up a task, and their parameters. Finally, since the skill model adheres to a STRIPS-like notation, we have demonstrated that task planning can be performed using the skills as the planning domain, contrary to traditional task planning using primitive motions. This enables the automatic construction of more advanced tasks, which would not be feasible to specify manually. We also demonstrate that a simple world model, focused on the valid parameters for the available skills, is sufficient for the approach to be realized.

Overall, this dissertation shows that the use of robot skills can be a valid solution to applying flexible robots in factories of the future. Through the use of general robot skills, the robot can handle a great variety of tasks in the relatively limited scenarios in the industry. Since the skills are intuitive, even for robotics novices, they can effectively be used for task programming, even by factory workers, whenever reprogramming of the robot is necessary. And since the skills contain the necessary information for task planning, i.e. preconditions and predicted outcome, the task programming can even be automated, when it is desirable.

Hopefully, this dissertation will serve as an eye-opener for both the academic and industrial communities, as to the direction research and development in industrial robotics needs to go. The dissertation shows valuable proofs-of-concept for improved human-robot interaction for industrial robots – an aspect that has largely been overlooked for many years by industrial robot developers.

13. Future work

The purpose of this final section is to act as a primer for future research within the field of robot skills, both for the wider academic community and for this researcher.

In the work presented in this dissertation, there is no formalized software structure for the skill implementation, nor for the complete system. We believe it is a natural next step to investigate how primitives, skills, and methods for task programming and task planning can be integrated in a complete framework, so that it can directly be used by system integrators on a wide variety of robots. A natural part of this is to expand the capabilities of the skills, to verify that the skills can be used in different task domains, including e.g. assembly tasks.

We also believe it is necessary to look into improved task programming capabilities. This could even be a combination of the proposed method for task programming and task planning, focusing on the specification of the desired goal state through intuitive methods such as pointing gestures or speech recognition.

An interesting research topic from the task planning point of view is also the link between robot skills and the planning domain. In Paper D, the domain was specified manually – which is sufficient for a low number of static skills. It would be interesting instead to automatically extract the available skill information on a given robot, and directly use this domain for planning. In this way, task planning will become possible for any skill-equipped robot.

In future research in this area, it is imperative that the methods are tested extensively in industrial settings, with real people, products and production equipment. This will not only be great experiments from a pure research point of view, but also help bridge the gap between academia and industry – a gap that is currently wide, unfortunate, and seemingly continues to grow.

References

[1] M. Hartmann, *DYNAPRO: Erfolgreich produzieren in turbulenten Märkten*. Logis Verlag, 1996.

[2] ——, *DYNAPRO II: Erfolgreich produzieren in turbulenten Märkten*. Logis Verlag, 1997.

[3] ——, *DYNAPRO III: Erfolgreich produzieren in turbulenten Märkten*. Logis Verlag, 1998.

[4] Paragon Technologies, "Teach pendants | paragon technologies," Oct. 2014. [Online]. Available: http://www.paragontech.com/teach-pendant-repair.html

[5] Rethink Robotics, "Baxter datasheet 5.13," Oct. 2014. [Online]. Available: http://cdn-staging.rethinkrobotics.com/wp-content/uploads/2014/08/Baxter_datasheet_5.13.pdf

[6] M. Hvilshøj, S. Bøgh, O. S. Nielsen, and O. Madsen, "Autonomous industrial mobile manipulation (AIMM): past, present and future," *Industrial Robot: An International Journal*, vol. 39, no. 2, pp. 120–135, Feb. 2012.

[7] S. Bøgh, M. Hvilshøj, C. Myrhøj, and J. Stepping, "Fremtidens produktionsmedarbejder - udvikling af mobilrobotten "lille hjælper"," Master's Thesis, Aalborg University, Aalborg, Denmark, 2008.

[8] M. R. Pedersen, "Integration of the KUKA light weight robot in a mobile manipulator," Master's Thesis, Aalborg University, Aalborg, Denmark, 2011.

[9] R. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, no. 1, pp. 14 – 23, Mar. 1986.

[10] C. Breazeal and B. Scassellati, "Robots that imitate humans," *Trends in Cognitive Sciences*, vol. 6, no. 11, pp. 481–487, 2002.

[11] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2, pp. 109–116, 2004.

[12] M. Lopes and J. Santos-Victor, "A developmental roadmap for learning by imitation in robots," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 37, pp. 308–321, Apr. 2007.

[13] S. Schaal, “Is imitation learning the route to humanoid robots?” *TRENDS IN COGNITIVE SCIENCES*, vol. 3, no. 6, p. 10, 1999.

[14] A. Björkelund, L. Edstrom, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. Robertz, D. Storkle, A. Blomdell, R. Johansson, M. Linderoth, A. Nilsson, A. Robertsson, A. Stolt, and H. Bruyninckx, “On the integration of skilled robot motions for productivity in manufacturing,” in *Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on*, May 2011, pp. 1 –9.

[15] C. C. Archibald, “A computational model for skills-oriented robot programming,” PhD Thesis, University of Ottawa, Ottawa, Canada, 1995.

[16] N. Krüger, J. Piater, F. Wörgötter, C. Geib, R. Petrick, M. Steedman, A. Ude, T. Asfour, D. Kraft, D. Omrcen *et al.*, “A formal definition of object-action complexes and examples at different levels of the processing hierarchy,” Technical report, 2009. [Online]. Available: http://www.paco-plus.org

[17] T. B. Moeslund, A. Hilton, V. Krüger, and L. Sigal, *Visual Analysis of Humans: Looking at People*, 1st ed. Springer-Verlag New York Inc, 2011.

[18] V. Krüger, D. Kragic, A. Ude, and C. Geib, “The meaning of action: a review on action recognition and mapping,” *Advanced Robotics*, vol. 21, no. 13, pp. 1473–1501, 2007.

[19] A. F. Bobick, “Movement, activity and action: the role of knowledge in the perception of motion.” *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 352, no. 1358, pp. 1257–1265, Aug. 1997.

[20] D. Newtson, G. A. Engquist, and J. Bois, “The objective basis of behavior units,” *Journal of Personality and Social Psychology*, vol. 35, no. 12, pp. 847–862, Dec. 1977.

[21] H.-H. Nagel, “From image sequences towards conceptual descriptions,” *Image Vision Comput.*, vol. 6, no. 2, pp. 59–74, May 1988.

[22] G. Rizzolatti, L. Fadiga, V. Gallese, and L. Fogassi, “Premotor cortex and the recognition of motor actions,” *Brain Research. Cognitive Brain Research*, vol. 3, no. 2, pp. 131–141, Mar. 1996.

[23] M. T. Mason, “Compliance and force control for computer controlled manipulators,” *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 11, no. 6, pp. 418 –432, Jun. 1981.

[24] H. Bruyninckx and J. De Schutter, “Specification of force-controlled actions in the "task frame formalism"-a synthesis,” *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 581 –589, Aug. 1996.

[25] J. De Schutter and H. Van Brussel, “Compliant robot motion i. a formalism for specifying compliant motion tasks,” *The International Journal of Robotics Research*, vol. 7, no. 4, pp. 3–17, Aug. 1988.

[26] T. Kröger, B. Finkemeyer, and F. Wahl, “Manipulation primitives — a universal interface between sensor-based motion control and robot programming,” in *Robotic Systems for Handling and Assembly*, ser. Springer Tracts in Advanced Robotics, D. Schütz and F. Wahl, Eds. Springer Berlin / Heidelberg, 2011, vol. 67, pp. 293–313.

[27] 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, Jun. 2005.

[28] T. Kröger, B. Finkemeyer, S. Winkelbach, L.-O. Eble, S. Molkenstruck, and F. Wahl, “A manipulator plays jenga,” *Robotics Automation Magazine, IEEE*, vol. 15, no. 3, pp. 79 –84, Sep. 2008.

[29] A. J. Ijspeert, J. Nakanishi, and S. Schaal, “Trajectory formation for imitation with nonlinear dynamical systems,” in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 2, 2001, pp. 752–757.

[30] S. Schaal, “Dynamic movement primitives -a framework for motor control in humans and humanoid robotics,” in *Adaptive Motion of Animals and Machines*, H. Kimura, K. Tsuchiya, A. Ishiguro, and H. Witte, Eds. Springer Tokyo, Jan. 2006, pp. 261–280.

[31] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, “Learning movement primitives,” in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, P. Dario and R. Chatila, Eds. Springer Berlin Heidelberg, Jan. 2005, no. 15, pp. 561–572.

[32] M. Waibel, M. Beetz, J. Civera, R. D’Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. M. M. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, “RoboEarth,” *IEEE Robotics Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.

[33] M. Tenorth, A. Perzylo, R. Lafrenz, and M. Beetz, "Representation and exchange of knowledge about actions, objects, and environments in the RoboEarth framework," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 643–651, 2013.

[34] C. Archibald and E. Petriu, "Skills-oriented robot programming," in *Intelligent Autinomous Systems III*, vol. 3, Pittsburgh, 1993, pp. 104–113.

[35] P. Neto and N. Mendes, "Direct off-line robot programming via a common CAD package," *Robotics and Autonomous Systems*, vol. 61, no. 8, pp. 896–910, Aug. 2013.

[36] H. Chen and W. Sheng, "Transformative CAD based industrial robot program generation," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 5, pp. 942–948, 2011.

[37] S. Mitsi, K.-D. Bouzakis, G. Mansour, D. Sagris, and G. Maliaris, "Off-line programming of an industrial robot for manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 26, no. 3, pp. 262–267, 2005.

[38] D. Gadensgaard and D. Bourne, "Human/robot multi-initiative setups for assembly cells," in *ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems*, 2011, pp. 1–6.

[39] Y. H. Jen, Z. Taha, and L. J. Vui, "VR-based robot programming and simulation system for an industrial robot," *International Journal of Industrial Engineering: Theory, Applications and Practice*, vol. 15, no. 3, pp. 314–322, 2008.

[40] R. Bischoff and A. Kazi, "Perspectives on augmented reality based human-robot interaction with industrial robots," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 4, 2004, pp. 3226–3231.

[41] S. Ekvall, D. Aarno, and D. Kragic, "Task learning using graphical programming and human demonstrations," in *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, 2006, pp. 398–403.

[42] M. Ralph and M. A. Moussa, "Toward a natural language interface for transferring grasping skills to robots," *Robotics, IEEE Transactions on*, vol. 24, no. 2, pp. 468–475, 2008.

[43] J. Blume, A. Bannat, G. Rigoll, M. Rooker, A. Angerer, and C. Lenz, "Programming concept for an industrial HRI packaging cell," in *2013 IEEE RO-MAN*, Aug. 2013, pp. 93–98.

[44] J. Blume, "iProgram: Intuitive programming of an industrial HRI cell," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Mar. 2013, pp. 85–86.

[45] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1972.

[46] S. Gottschlich, C. Ramos, and D. Lyons, "Assembly and task planning: A taxonomy," *Robotics & Automation Magazine, IEEE*, vol. 1, no. 3, pp. 4–12, 1994.

[47] G. Biggs and B. MacDonald, "A survey of robot programming systems," in *Proceedings of the Australasian conference on robotics and automation*, 2003, pp. 1–3.

[48] H. Mosemann and F. Wahl, "Automatic decomposition of planned assembly sequences into skill primitives," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 5, pp. 709 –718, Oct. 2001.

[49] S. Ekvall and D. Kragic, "Robot learning from demonstration: A task-level planning approach," *International Journal of Advanced Robotic Systems*, vol. 5, no. 3, pp. 223–234, 2008.

[50] C. Galindo, J.-A. Fernández-Madrigal, J. González, and A. Saffiotti, "Robot task planning using semantic maps," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 955–966, Nov. 2008.

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

[52] Y. Lespérance, H. J. Levesque, F. Lin, D. Marcu, R. Reiter, and R. B. Scherl, "A logical approach to high-level robot programming–a progress report," in *Control of the Physical World by Intelligent Systems: Papers from the 1994 AAAI Fall Symposium*, 1994, pp. 79–85.

[53] H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. B. Scherl, "GOLOG - a logic programming language for dynamic domains," *The Journal of Logic Programming*, vol. 31, no. 1-3, pp. 59–83, 1997.

[54] D. Hähnel, W. Burgard, and G. Lakemeyer, "GOLEX—bridging the gap between logic (GOLOG) and a real robot," *KI-98: Advances in Artificial Intelligence*, pp. 165–176, 1998.

[55] A. Ferrein and G. Lakemeyer, "Logic-based robot control in highly dynamic domains," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 980–991, Nov. 2008.

Part II.

Papers

Paper A

Robot Skills for Manufacturing:
from Concept to Industrial Deployment

Mikkel Rath Pedersen, Lazaros Nalpantidis,
Rasmus Skovgaard Andersen, Casper Schou, Simon Bøgh,
Volker Krüger, Ole Madsen

The paper has been submitted for review November 1st 2014 to
Robotics and Computer Integrated Manufacturing.

The layout has been revised.

Abstract

Due to a general shift in manufacturing paradigm from mass production towards mass customization, reconfigurable automation technologies, such as robots, are required. However, current industrial robot solutions are notoriously difficult to program, leading to high changeover times when new products are introduced by manufacturers. In order to compete on global markets, the factories of tomorrow need complete production lines, including automation technologies that can effortlessly be reconfigured or repurposed, when the need arises. In this paper we present the concept of general, self-asserting robot skills for manufacturing. We show how a relatively small set of skills are derived from current factory worker instructions, and how these can be transferred to industrial mobile manipulators. General robot skills can not only be implemented on these robots, but also be intuitively concatenated to program the robots to perform a variety of tasks, through the use of simple task-level programming methods. We demonstrate various approaches to this, extensively tested with several people inexperienced in robotics. We validate our findings through several deployments of the complete robot system in running production facilities at an industrial partner. It follows from these experiments that the use of robot skills, and associated task-level programming framework, is a viable solution to introducing robots that can intuitively and on the fly be programmed to perform new tasks by factory workers.

1. Introduction

In order to remain competitive in a globalized environment, manufacturing companies need to constantly evolve their production systems and accommodate the changing demands of markets. Currently, production is experiencing a paradigm shift from mass production to mass customization of products. The impact of this trend on production systems is that they should adapt to handle more product variation, smaller life cycles, and smaller batch sizes – ideally batch size 1. Today, robot-based production is an essential part of the industrial manufacturing backbone. However, the concept of an industrial robot statically placed in a cell and continuously repeating a carefully predefined sequence of actions has remained practically unchanged for many decades. Not surprisingly, typical industrial robots are not flexible and thus such a degree of *transformable production* [1–3] is beyond the capabilities of current systems.

A simplified illustration of the situation is shown in Figure A.1. Traditional manufacturing systems are automated to a large degree, but can only be reconfigured with great difficulty. On the other hand, traditional manual labor is very flexible, but not economically viable for large scale production, especially so in high-wage countries. In the future, mass customization will make it necessary to combine high reconfigurability with a large degree of automation. This goal can be achieved in two distinct ways: Workers can either be equipped with better automation tools, such as intuitive on-the-fly programming of robots, thereby increasing their productivity, or the reconfigurability of traditional automated production lines can be improved, e.g. through the use of multipurpose robots.

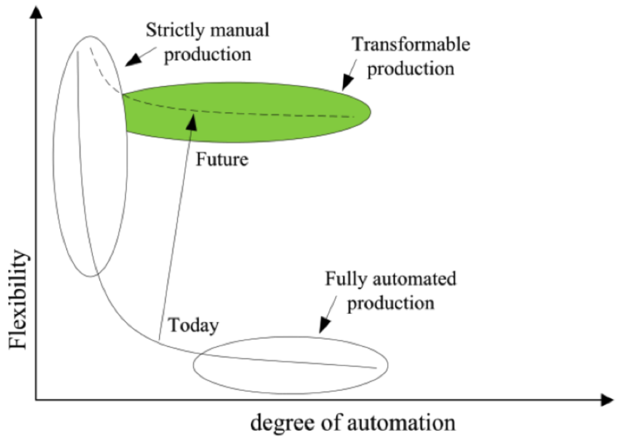


Fig. A.1.: A clear requirement for the factories of tomorrow is that of a *transformable* production system, showing a high degree of flexibility, while still applying a high degree of automation.

Robotics is expected to be one of the main enablers of this transition to the transformable factory of tomorrow. To reach the demanded level of flexibility robots, or more generally mobile manipulators, need to be able to move autonomously, cope with uncertainty in interactions with humans and partially known environments, handle a variety of different tasks, and be able to be reprogrammed fast by non robot experts when a new task in the factory arises.

So, the question emerges: What should be the characteristics of a framework that would allow robots to be flexible enough to handle uncertainty and changing production tasks, while being intuitive enough to be used and re-programmed by non-robot experts? We argue that the answer lies in the tight integration of sensing and action within a small set of modular and parametrizable *"robot skills"*. Contrary to traditional robot macros, our skills are characterized as being general, in that they can handle a variety of objects, and self-asserting, in that they contain pre- and postcondition checks. Finally, this set of skills for a given industry can be naturally extracted from a careful analysis of industrial standard operating procedures (SOP).

In this work we present a conceptual model of the *"skills"* and how this concept can be implemented and successfully applied in real-world industrial scenarios. The conducted experiments show that a small set of skills can be intuitively parametrized by either kinesthetic teaching, pointing gestures, or automatic task planners to solve different industrial tasks, such as part feeding, transportation, quality control, assembly, and machine tending.

The main contributions of this paper are that:

- We propose a conceptual model of robot skills and show how this differs from traditional macros,
- we show how this approach can enable non-experts to utilize advanced robotic systems by encapsulating expert knowledge in each individual skill, and
- concrete applications of the approach are presented, in which advanced robot systems have been applied in several real industrial scenarios.

The rest of this paper is organized as follows: in Section 2 we discuss how the list of skills can be extracted for a given industry and how the concept of skills can be formalized. In Section 3 we present a series of experiments were robots powered by skill-based software architecture were deployed in real industrial settings. We discuss the outcomes of our work and possible future directions in Section 4 and finally we conclude with Section 5.

1.1. Related work

The goal of achieving easy and quick reprogramming of robots by non experts has been consistently pursued using the task-level programming paradigm. Task-level programming constitutes a higher abstraction level than traditional robot programming. It specifies a given task as a sequence of actions, but avoids describing each action in full detail. The sequence of actions leading to the fulfillment of this task can either be planned or specified by an operator. Such an approach is similar to the symbolic representation of tasks of the Shakey robot that used a STRIPS planner to reason about which actions may lead to accomplishing a goal [4], or the concept of Object-Action Complexes (OACs) [5] that allowed cognitive robots to identify possible actions based on object affordances.

Task-level programming is based on lower level entities, usually called *robot skills*, that instantiate actions. Various representations of robot skills that would be suitable for task-level programming have been pursued during the last decades [6–14]. What most of the aforementioned attempts have in common, is that robot skills are, in turn, composed of primitive sets of robot motions, called action or motion *primitives*. These skill primitives are simple, atomic robot movements that can be combined to form more complex behaviors [15–21]. A systematic survey of the rich relevant literature reveals how fragmented the concept of skills is, lacking a widely acceptable, strict definition.

In contrast to the skills themselves, the *skill primitives* [10, 11, 13] are rather well defined in the robotics community; although several different descriptions exist, most of them loosely follow Mason's work on compliant controllers [22], which paved the ground for the Task Frame Formalism (TFF) introduced by Bruyninckx and De Schutter [23, 24]. Recent work has expanded upon this idea, for instance enabling the use of any sensor and controller in the same task frame (e.g. visual servoing combined with force control), and operating in a dynamic task frame [25–27].

Our definition of skills integrates pre- and post-conditions checks in the execution, which makes the interconnection of skills into tasks an intuitive and controlled process. In that sense, our approach is comparable to the *manipulation primitive nets*, described best in [26]. However, this approach requires low-level parametrization of the manipulation primitives, making it accessible only to robot experts. Another concept, similar to our notion of skills, is that of Action Recipes as defined in the RoboEarth project [28, 29]. Different to our skills though, Action Recipes keep sensing and acting separate and mainly focus on knowledge sharing between robots with similar capabilities. All of the aforementioned approaches require a robot expert to program and use the system. This is also one of the reasons reasons why these works, in contrast to ours, were never deployed in a real industrial setting.

Easy programming of industrial robots requires, apart from the underlying concepts, intuitive Human-Robot Interaction (HRI) mechanisms. HRI in industrial settings focuses on offline programming by using CAD models [30–32] or online programming using augmented reality (AR) [33–35]. The apparent limitation of such methods is that they assume the presence of an expert for reprogramming, which makes them unsuitable for the envisioned transformable factories. In contrast to the majority of publish works and similar to our approach, the authors of [36] combined skill primitives and speech recognition for programming grasping tasks by speech, with other work presented in [37, 38]. This work incorporates task-level programming, however, the programming is a combination of primitives, with no higher-level skills between these and the task layer. In general, the literature of HRI in industrial settings is rather limited. One possible reason is that industrial robotics applications traditionally focused on accuracy and speed without the need for external sensors. However, lately this trend is giving way to more intuitive and user-friendly interfaces that largely rely on sensing [39].

Apart from HRI that allows human instructions to define a task, another possibility is to consider use of automated task-planers. Industrial robots have been endowed with task-planning capabilities, starting with the STRIPS planner [4], and many other algorithms that followed, based on the same ideas, in that they act on a set of actions that alters the current world state [40, 41]. This includes breaking down assembly tasks into some form of manipulation primitives the robot can interpret [42]. Simple robotic scenarios, as in GOLEX [43], have been succeeded recently by more advanced ones [44], resembling to some of the capabilities of our work presented in the paper at hand.

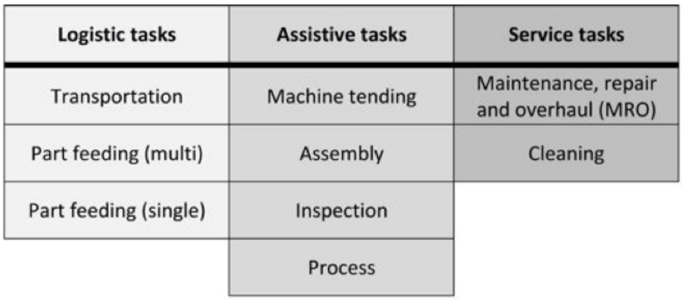
2. Finding and formalizing robot skills

2.1. Finding skills

As mentioned earlier, mobile manipulators can be one of the enablers in robotics to accommodate the higher demand for flexibility in future industrial production. We argue that the use of robot skills is the key to making this happen. We will begin by showing how robot skills have been identified from the current work procedures in the industry.

As a starting point, we have investigated applicable task domains for mobile manipulator applications. A range of general industrial applications suitable for mobile manipulators have been identified through an extensive study in real-world production environments [45]. The scope of the study included the analysis of 566 industrial tasks, carried out by human workers, since the tasks were either too complex or low volume to justify automation solutions [45, 46]. We have grouped these tasks into three different task domains, with distinct task categories in each domain, as shown in Figure A.2.

Within the logistic and assistive task domains, we have analyzed the set of skills required to solve a majority of these tasks. This analysis is based on the workflow of the factory workers, in order to detect reoccurring, fundamental actions, or skills, which can be implemented on a robot, so that they are still



Logistic tasks	Assistive tasks	Service tasks
Transportation	Machine tending	Maintenance, repair and overhaul (MRO)
Part feeding (multi)	Assembly	Cleaning
Part feeding (single)	Inspection	
	Process	

Fig. A.2.: Nine general task categories in three domains, suitable for mobile manipulators.

intuitive to use by the factory workers for constructing higher-level tasks. We base the analysis on a set of Standard Operating Procedures (SOPs), which formally specify in writing the task descriptions for factory workers, accompanied by images. Each step of the SOPs is mapped to respective skills. For example, a step in a machine tending task is described as “pick up the shaft, insert it in the assembly machine, and press the start button on the machine,” and leads to the conclusion that this particular step consists of the three skills ***pick up*** *object*, ***place in*** *machine* and ***press*** *button*. This analysis leads to a total of only 13 of these general skills across the two analyzed task domains [47].

One important aspect of the analysis is that it is based on the natural language and communication between the people who work in the production, and the language used in the SOPs. This has the effect that the skill concept, and specifically the outcome of instructing a robot to execute a particular skill, is immediately intuitive for the factory workers, who will use the skills directly.

Based on the findings from the analysis of the SOPs, we will now go on to formalize what exactly robot skills are, and how they can be used for task-level programming.

2.2. Skill formalization

In this section we will explain our conceptual approach to robot skills. The section will thoroughly introduce the reader to the concept of robot skills, as envisioned by these authors, as well as the three-layer architecture of skills, tasks and primitives. We will finally summarize the advantages of using skills for manufacturing.

2.2.1. Conceptual model of a skill

In a nutshell, we see robot skills as the high-level building blocks from which a complete robot program, or task, can be composed, and that allow the robot to complete the task. In other words, skills are intuitive object-centered robot abilities, which can easily be parameterized by a non-expert.

Our focus is especially on improved human-robot interaction, and on the reusability of skills within the domain of tasks that are frequently required by the manufacturing industry.

One core property and main justification for using skills in this context is their object-centeredness, i.e. a skill incorporates both sensing and action performed on the object. This means that skills are not applied on 3D locations but rather on physical entities the shop floor worker can relate to, which is exactly the objects that are present in the factory where the robot is deployed.

We define a task as a fully instantiated sequence of skills, with specified parameters, and it is characterized as solving a specific goal in the environment where the robot is deployed, e.g. loading an assembly machine with the necessary parts to create a certain product or subassembly.

In order for the skills to be useful for task-level programming, they must both change the robots notion of the state of the world through sensing or action, and be self-sustained, so they can be used in any task. Self-sustainability implies that each skill should be

- parametric in its execution, so it will perform the same basic operation regardless of input parameter,
- able to estimate if the skill can be executed based on the input parameter and world state, and
- able to verify whether or not it was executed successfully.

Our conceptual model of a robot skill, which incorporates all of the above points, is shown in Figure A.3.

A skill relies on a defined action and sensing sequence, but the motions themselves are adapted to a specific task by a set of parameters, thus making the skills generic within a certain scope. Each skill has one or more intuitive parameters as input, which is all the factory worker has to relate to. These parameters are the objects that are relevant for the skills, and the execution of a skill is based on only this parameter.

During the execution phase, the robot performs the pre-programmed sensing and action operations to accomplish the skill, based on the input parameter. These operations are specified by the skill integrator, and should ideally

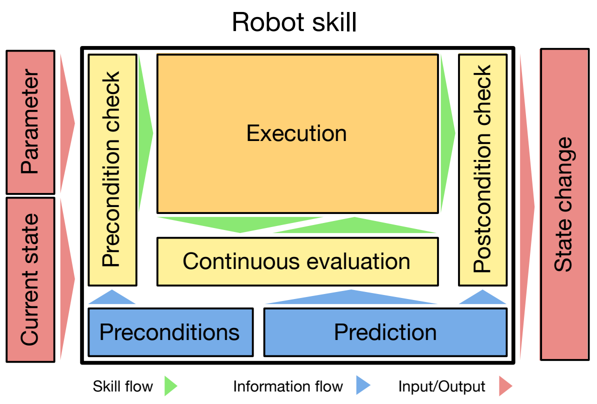


Fig. A.3.: Conceptual model of a robot skill. The preconditions and prediction (blue blocks) are informative aspects of the skill. The checking procedures (yellow) verifies the informative aspects before, during and after execution (orange). Execution is based on the input parameter and the world state (red), and the skill effectuates a change in the world state (red).

be implemented in such a way that the skill can handle all parameters that are relevant in the factory. The operator is never exposed to the individual operations within the skill, but only has to relate to the complete skill and its intuitive parameters.

Each skill has associated a set of preconditions and a prediction of the outcome of its execution in the real world, which are used for the checking procedures before and after execution. These can either be verified from the associated world model or through ad-hoc sensing operations, depending on the condition. Furthermore, the preconditions and prediction enables task planning on the skill domain, as the skills conform to a STRIPS-like formulation [4]. In this case, only the goal state is required for task specification, given that the robot has an associated world model, from which the current state can be extracted for the planning problem.

The prediction specifies formally what the expected effect of executing the skill is. During ongoing execution, continuous evaluation of the performance of the skill ensures is is executed as expected. When the execution has finished, a final postcondition check determines if the current world state is as predicted.

Modeling the world state Since the skills need to access prior knowledge of objects and locations in the factory, a notion of the state of the world need to be modeled.

Skills are effectuating a change in a set of state variables, describing the knowledge the robot has of its surroundings. State variables can be either measured with vanishing uncertainty by dedicated sensors, e.g, by those that are built into the manufacturing systems, or by sensors on the robot, such as vision, torque or tactile sensors.

For the skill approach to work, the world model must contain information that is directly relevant to the skills. Therefore, at the very least, the world model must contain information about all the aspects of the environment where the robot is deployed, that can be used as parameters for the skill working on the robot. Since skills are object-centered, the world model therefore must contain prior knowledge, such as global 3D coordinates, about e.g. objects, locations and production equipment that is relevant for the skills. In fact, in most cases *only* an ad-hoc, skill-relevant world model is sufficient, containing only information that is relevant parameters for the skills currently implemented on the robot, and the scenario where the robot is deployed.

2.2.2. Architecture of tasks, skills and primitives

The approach of task-level programming using skills is divided into three layers, inspired by [15] and [11]; the *primitive*, *skill* and *task* layers. The architecture is shown in Figure A.4, and we will explain how the layers work together in the following. We will not give details here on the implementation of each layer, nor describe a complete framework for skill-based programming, as this is ongoing work.

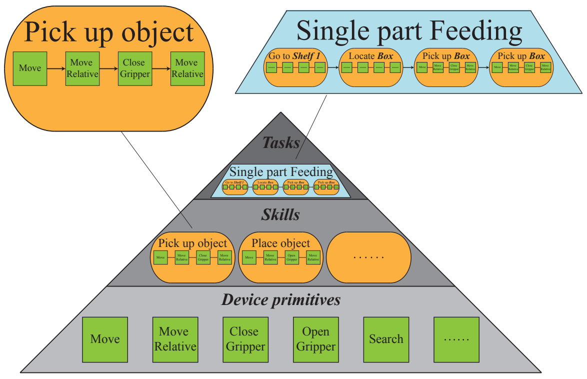


Fig. A.4.: The three layers of primitives, skills and tasks. The components in each layer is essentially a combination of components from the lower layer. Skills are predefined by the skill programmer, and tasks are programmed by the operators in the factory, or planned online.

At the lowest level, which is the most hardware-specific, we have the *primitive* layer. This layer contains the real-time control loops of the robot, as well as the necessary sensing operations that are necessary for the skills to work. This layer is similar to the level of programming that typically is carried out on industrial robots, but with the addition that movements can be interrupted and altered in real-time, based on feedback from sensors. The primitives are characterized as performing a single operation related to the robot system. Examples of primitives include opening the gripper, planning a motion of the robot arm to a certain configuration, and calibrating the position of the robot with respect to a specific production machine.

The *skill* layer is the layer that is exposed to the end user, i.e. the operator, when programming new tasks on the robot. Currently much robot programming is carried out by engineers or robotic system integrators, and even though many applications for stationary industrial robots have similarities, tasks are usually programmed from scratch. For flexible robots, it is necessary to minimize that effort, and adding functionality in the building blocks is a way to do that. However, subprograms, as skills can be considered, can only be reused if they are both generic and robust for the possible set of applications.

Skills are combinations of primitives, where the exact choice of which primitives to execute is determined online. Skills are implemented by utilizing the motion primitives for creating advanced, yet versatile, program building blocks, and establishing the prediction and the pre- and postcondition checks. For example, the ***place** object* skill is a combination of the primitives that move the arm, open the gripper and detects surfaces, which can be specified by the skill programmer, but the exact motions that are carried out are determined at the skill runtime.

The tasks are directly related to solving specific goals in the factory, and must interface with end users, programming and initiating tasks, and systems controlling the production line, e.g. manufacturing execution systems (MES). A task is a predefined sequence of skills, with previously specified parameters, that can be executed whenever necessary. The skill sequence and parameters, that make up a task, can be programmed explicitly by an operator, or planned ad hoc based on the current modeled world state and a desired goal setting of the state variables.

The world model associated to the complete system can only be accessed from the skill and task layers. As previously stated, this world model is skill-centered, containing only the parameters that are necessary for the skills to function. This also means that it is the skill that alter the world state – a tasks can only access the state variables, but changes them through the skills. For instance, consider the task of removing all objects from a table. The task needs to know the initial amount of objects on the table, but it is the skills that update the world model when an object is removed.

2.3. Advantages of using skills

The main justification for using skills in this research is that of easily transformed production systems, specifically robot systems. The skills approach to this topic yields a number of advantages that we will outline in the following.

First and foremost, since skills ideally are general within the scenario where they are deployed, the robot system can deal with product variety in the factory. It should be effortless to enable a robot to deal with new products, when it is introduced in the factory, which is exactly the point of having skill-centered world knowledge. This results in a more effective and streamlined production, since a skill-equipped robot can react to changing products and production volumes – the key point of implementing transformable solutions in robotics.

Furthermore, the skills are so intuitive that it is possible for non-experts, i.e. the shop floor workers, to program new tasks on the robot, whenever the need arises. This stems partly from the fact that the skill are grounded in the SOPs, as explained in 2.1. This makes it easy for robotics novices to understand both what the robot will do, and the outcome of each skill, which is exactly what is needed to intuitively program a task as a sequence of skills.

As we have already mentioned briefly, we can even automate the task programming. Since the skills conform to the STRIPS formulation, in that they contain information about their input parameters, preconditions and predicted effects, planning on the domain of skills is possible. In this case, a task specification merely becomes a description of the desired goal state of the relevant state variables, and the task will never have to be programmed explicitly. The goal state can be specified by a factory worker, or even from the Manufacturing Execution System (MES), offloading the factory workers for other assignments in the factory.

From an integration viewpoint, an added benefit of using the three-layer architecture outline in 2.2.2 is hardware abstraction. A well-defined interface between primitives and skills facilitates substituting the primitives. By substituting primitives, it is not only possible to add e.g. improved motion planning primitives to the robot. Since a skill performs its sensing and action operations based on the primitives, it is even possible to have the exact same skills running on two seemingly very different robots, as long as the primitives perform similar, and maintain their interfaces to the skills.

Finally, when the skills approach is mature enough, it will help technology push from academia to industry. The reason for this is simple: If there is only a small number of skills, the focus of system integrators and developers will change towards development and improvement of primitives and skills, and this will eventually have to be grounded in state of the art research.

We will now go on to demonstrate a number of these advantages through a range of experiments, in both lab and industrial settings.

3. Experiments

In this section we show experiments in skill implementation and task programming, and conclude with showing a skill-equipped robot system integrated in a running production line, and showing a summary of the results in the end of the section. For all of these experiments, we use an in-house developed industrial mobile manipulator, called Little Helper.

The focus of the Little Helper concept is industrial applications, and for addressing the needs for flexible automation technologies. Although several iterations of the Little Helper exist, all are built from standard industrial components, and current versions (one of which is shown in Figure A.5) consists mainly of three hardware components. For manipulation we use a 7 DOF KUKA Light-Weight Robot (LWR) arm. The robot arm is mounted on a chassis which also contains the robot controller and a computer controlling the entire system. The chassis is mounted on a Neobotix MP-655 differential drive mobile platform. Unless otherwise stated in the following sections, we use a Schunk WSG-50 parallel gripper for grasping. The robots are equipped with different configurations of RGB-D cameras from Microsoft, ASUS or Primesense. All versions of the Little Helper are running ROS.

3.1. Skill implementation

This section will showcase our experiments into implementing skills on the Little Helper robot. These experiments form the backbone of the skill-based task programming paradigm, since they verify that skills can in fact be implemented according to the conceptual model shown in Figure A.3.

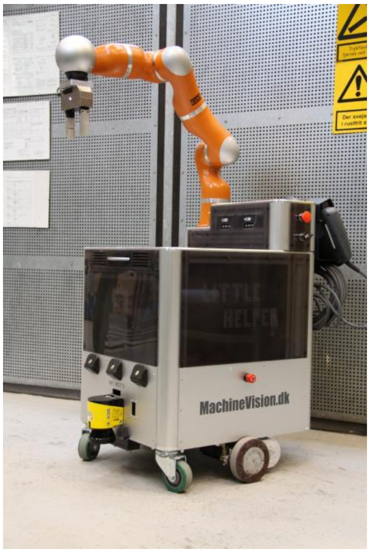


Fig. A.5.: One of two current versions of the Little Helper robot.

3.1.1. Basic manipulation skills

In our initial findings on skill implementation, we have implemented the basic manipulation skills ***pick*** *object* and ***place on*** *table*. Both of these skills are essential for mobile manipulation, and will be an integral part of a task-level robot programming paradigm, as they will both be used in almost any mobile manipulation task.

Referring to the skill model, the skills must contain the necessary sensing and motion operations that accomplish the goal of e.g. picking up an object. In [48], the ***pick*** *object* skill is presented, which allows the robot to identify and pick up objects on a flat surface. The experimental setup for this skill is shown in Figure A.6.

The skill was developed particularly for near-rotational symmetrical objects, and was tested on the objects shown in Figure A.7. All objects could be picked. Only the most deform cylinder in A.7c showed a high variability of the final pose in the gripper, and was subsequently placed (through blind motion, i.e. not with a placing skill) in a slightly different location than expected. This was caused by limits in the vision system and the gripper, and could thus be avoided by choosing different, more appropriate hardware.

We extend the ***pick*** *object* skill and include the ***place on*** *table* skill in [49]. In this work we leverage sensing primitives readily available from the commu-

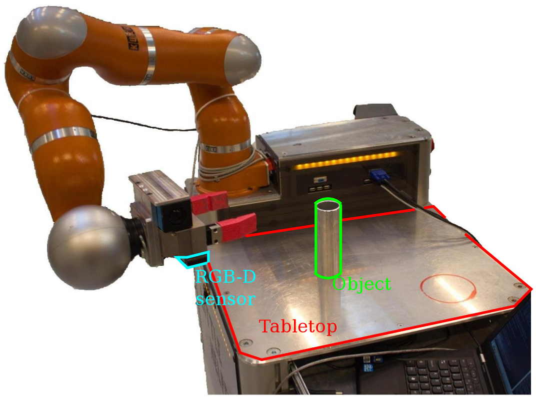
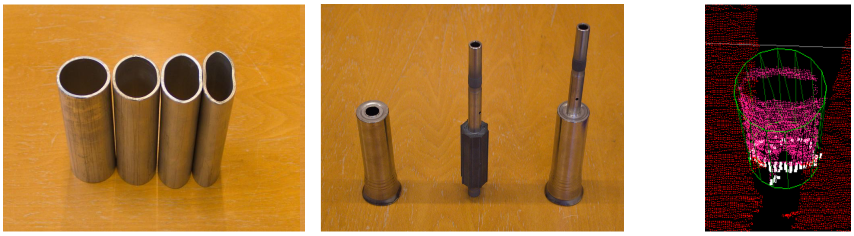


Fig. A.6.: The robot system consists of a robot arm, gripper, depth sensor, tabletop, and an object to pick.



(a) Cylindrical and deform objects

(b) Pump parts from an industrial production line

(c) Segmented pointcloud

Fig. A.7.: Objects that could successfully be grasped by the same skill. All objects could afterwards be placed using the same place skill. In A.7c, a deform cylinder has been detected and segmented and a cylindrical, containing model has been fitted.

nity, that can detect common household objects[1], and show a skill-focused, ad-hoc world model, containing previous observations of these objects.

The focus of the latter work is on hardware transferability, and we demonstrate how two skills implemented on one robot, a PR2, can be transferred with relative ease to a very different robot, the Little Helper. By implementing the same skill primitives on both robots, the skills are directly transferable. However, we have found that this is not always possible, as different robots have different capabilities, as was the case in these experiments. Instead, we have maintained the *interfaces* to the skill primitives. For instance, for controlling robot arm movements, we simply pass the desired pose of the gripper to

[1]Specifically, the `tabletop_object_detector` package in ROS Fuerte

the corresponding *move arm* primitive, and the robot performs the arm movement through either collision-free motion planning (for the PR2) or a blind, joint interpolated motion (for the Little Helper).

Since the purpose of these experiments was to show skill implementation and hardware abstraction, the highest layer of the architecture approached was the skill layer. However, since these skill implementations include world knowledge, pre- and postcondition checks, as well as the execution, they can easily be concatenated to complete tasks, as we will present later.

3.1.2. Calibration skills

For mobile robots, calibration relative to the environment is necessary whenever the robot arrives at a workstation. This enables the robot to know the locations of e.g. buttons and knobs of the workstation without detecting these explicitly, within the tolerances of the calibration routine. A calibration routine does not change the physical world, but rather updates the robot system's internal world model. Therefore, it can be classified as a skill.

Four different calibration skills have been developed in our skill framework:

1. Haptic calibration [50]: The robot uses moves along the robot's x, y, and z axes until physical contact with robust points in the workstation itself.
2. Visual high-precision [51]: Multiple images of a calibration plate are captured with a camera on the end-effector. The calibration plate is fixed relative to the workstation.
3. Visual high-speed [51]: One image of the same calibration plate is captured. Additionally, three distance measurements are made with a laser sensor, also mounted on the end-effector.
4. QR calibration [52]: An RGB-D camera on the robot platform captures one image and one depth image of a QR code, which is fixed relative to the workstation. This calibration skill is shown in Figure A.8.

Each of the methods have different advantages. Their execution time and precision is compared in Table A.1. It is clear that for very precise calibration, the high precision approach should be used. If less precision is necessary, QR calibration can be used instead, saving a significant amount of time. Additionally, if there is no room for a large marker close to the workstation, haptic calibration can be applied instead.

In the following section we will show our experiments conducted within task programming.

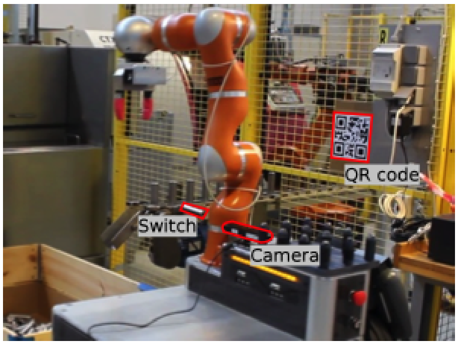


Fig. A.8.: Fast QR calibration using RGB-D camera. The camera detects the position and orientation of the fixed QR code, and subsequent movements with the robot are corrected accordingly.

Method	**Duration**	**Precision**
Haptic	$30 - 45s$	$\pm 1.0mm$
High speed	$10s$	$\pm 1.0mm$
High precision	$60s$	$\pm 0.1mm$
QR calibration	$< 1s$	$\pm 4.0mm$

Table A.1.: Comparison of calibration methods.

3.2. Task specification

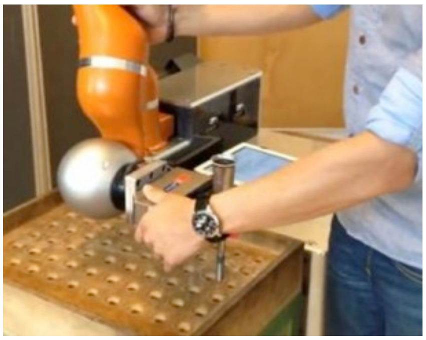
The key motivation for skills is to improve Human-Robot Interaction (HRI) capabilities of industrial robots, as described in section 2.2.1. Specifically, we focus on making task programming available to non-experts by establishing an intuitive task programming framework, compared to traditional robot programming methods. The skills forms the basis for creating a higher level of abstraction of robot programming, which better relates to the human operations in a given task. As a result, programming becomes more related to the operations in the SOPs.

3.2.1. Explicit task programming

In our initial work on task programming, we focus on the explicit programming of robot tasks, constructed from a sequence of skills. Task programming using skills can be divided into the two distinct steps of *sequencing* and a *teaching*. In the sequencing phase, the task programmer constructs the sequence of skills in the task, where each skill is subsequently parameterized in the teaching phase.

The sequencing phase is performed in a touchpad-based GUI on a tablet, in which the operator selects skills from the skill library to create a sequence matching the operations in the task. For the teaching phase, we have investigated the uses of both kinesthetic teaching and gesture-based input.

Kinesthetic teaching In one set of experiments, the operator uses kinesthetic teaching to directly interact with the manipulator [53]. The robot arm is piloted to target locations and input is given by pushing the end-effector in certain directions. The operator is guided during teaching via audiovisual instructions on the associated tablet. The kinesthetic teaching is illustrated in Figure A.9a.


(a) Kinesthetic teaching

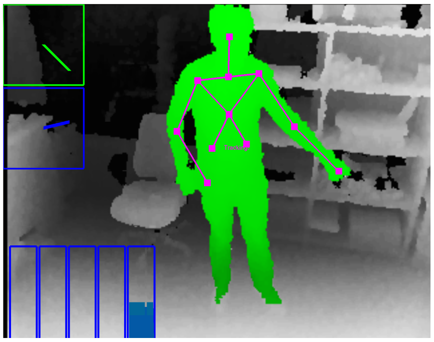

(b) Gesture recognition

Fig. A.9.: Teaching of parameters through direct interaction with the manipulator using kinesthetic teaching [53], and through the recognition of pointing gestures [54].

The approach was verified through experiments with a total of 18 people, many of which were inexperienced in robotics[2]. In these experiments participants programmed two pick and place tasks of varying complexity. Besides specifying coordinates through kinesthetic teaching, participants had to relate to specific parameters, such as robot speed, force-profiles, compliance, etc. The total programming time ranged from 2.6 to 12.5min, across all participants.

Gesture-based teaching In our experiments with gesture-based teaching [54], we have implemented a set of simpler manipulation skills, that only accept a single parameter as input. For instance, the ***pick*** *object* skill only requires the object to pick up as a parameter, where everything else is handled within

[2]Although only 9 are mentioned in [53], additional tests with 9 different participants were recently conducted within the EC-FP7 TAPAS research project (`www.tapas-project.eu`)

the skill. We add gesture recognition capabilities to the Little Helper, which has the effect that an operator can accomplish the teaching phase simply by pointing at the parameters for each skill in the sequence. Beside the pointing gesture, the operator has access to additional gestures, such as instructing the robot to follow him/her around the factory.

We have investigated the approach by having users program a simple task of exchanging two objects at two locations – a mock-up of a machine tending task, where a box is filled with parts and needs to be exchanged with an empty one. All users were given a single 10-minute introduction on how to use the system.

Experiments with 24 people[3], more than half of which were completely inexperienced with robots, showed that this is a feasible and intuitive method for task programming. The teaching of the parameters, including navigating the robot to the different locations, by having it follow the participant, was equally intuitive. The teaching phase only took between 2 and $7min$ across participants, the mean time being a little less than $3min$. Furthermore, there were no major difference in programming time between male/female participants or participants experienced/inexperienced in robotics.

In both the experiments with kinesthetic and gesture-based teaching, the sequencing step was immediately intuitive for all participants, proving that the skills themselves are intuitive. Users generally also expressed that the teaching phase was intuitive, although more so for the gesture-based programming, due to the skills being simpler and only requiring a single parameter as input. Based on the two experiments, and several minor experiments in the lab and at industrial manufacturing environments, the conclusion is, that the skill-based programming approach does in fact enable non-experts to *intuitively* and *quickly* program industrial relevant tasks.

3.2.2. Task-level planning

Although it is in many cases desirable to have complete control over the sequence of skills, in some cases, explicit task programming does not work. This is especially true for human-populated environments, like factories, that are prone to disturbances. We have investigated how robot skills instead can be used for planning, instead of traditional task planning on the motion primitive domain. By planning on the human-specified skills, the planning domain becomes much smaller, since there are fewer skills than motion primitives. Furthermore, a planner would not have to plan the reoccurring motion prim-

[3]Again, with additional experiments with 7 people conducted recently.

itives that e.g. pick up an object, since this is instead modeled within a single skill.

We have investigated how task planning on the robot skill domain can be used for multiple part feeding scenarios. The specific use case for these experiments is a production line as an industrial partner. In this production line, three feeders need to be regularly filled with parts for an assembly line. Parts are located at a warehouse in bulk in small containers, which the robot can manipulate. Empty containers need to be placed at a different warehouse.

Since the skills already conform to the STRIPS notation [4], in that they contain a notion of the preconditions and predicted outcome, they can be immediately specified in PDDL [55]. In these conceptual experiments, we have modeled the skills ***drive to*** *location*, ***pick up*** *object*, ***place at*** *location* and ***unload in*** *feeder* in PDDL. We model the robot Little Helper to be able to carry two out of the three necessary containers to fill all feeders. For the goal specification, we specify that all three feeders should be filled, and all empty containers should be at the appropriate warehouse.

Our experiments show that the skills can be immediately used for planning, a plan satisfying the goal can be found in the order of $0.1s$, and the output plan contains the sequence of skills, and the required parameters to run them. In contrast to a user-specified task, which would be a hardwired sequence of skills and parameters, the planned task can also be found based on the current world state. This has the advantage that the goal can simply be specified by the manufacturing system, and not by an operator – and since this is all that is required besides the world model on the robot, ad hoc task planning on the skill domain is indeed feasible.

3.3. Industrial use cases

Besides the conceptual experiments described in the previous sections, the skills have also been evaluated in real world industrial settings, through several experiments. The conceptual experiments presented above has demonstrated the feasibility of the skill concept and the possibility to program industrial tasks. However, real world industrial tasks often constitute other challenges; thus the motivation for the real world experiments has been to evaluate the applicability of the skill concept in an industrial setting.

3.3.1. Multiple Part Feeding

Some of our first industrial experiments were with multiple part feeding. In these experiments, a mobile robot took care of part feeding to a series of fully

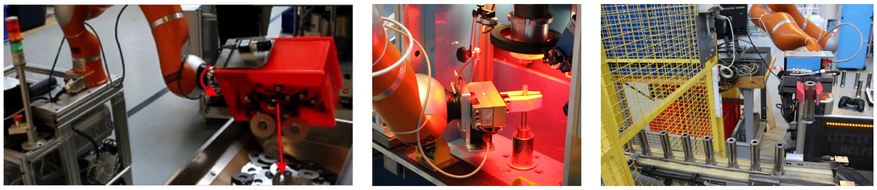

(a) Multiple part feeding **(b)** Machine tending **(c)** Part feeding

Fig. A.10.: Photos from real world industrial integration experiments where the skills were used for task programming and execution.

automated manufacturing cells. The task was to pick up a box from a warehouse, bring it to a feeding machine at the manufacturing cell, and empty the content into the feeding bowl. Afterwards, the empty box was brought back to the warehouse. The scenario was instructed using a total of 8 unique skills, and the task was subsequently executed continuously over a period of more than 1.5 hours. Figure A.10a shows the robot emptying the box into the feeding machine. A thorough description of this experiment is presented in [56].

3.3.2. Machine tending

As part of a larger industrial integration experiment, a complex machine tending task was instructed and afterwards executed using the task programming method described in Section 3.2.1. After task programming, the experiment reproduced a full 8-hour shift. This experiment was conducted using the real world manufacturing machinery and components on the shop floor at an industrial partner. In this task the operator would assemble 11 components in a fixture inside a hydraulic press, activate the press to join the components, and finally remove the finished sub-assembly. This task was automated using in total 11 unique skills combined into a sequence of 118 skills. This task included the use of compliant motions, force feedback, and communication with external equipment. Figure A.10b shows the robot performing the assembly task inside the press.

3.3.3. Logistic tasks

A logistic part feeding task was instructed as part of and executed in the same 8-hour long industrial integration experiment as the machine tending task. In this task the mobile robot arrived at a conveyor belt, where components exited a machining cell. These components ordinarily dropped into a bin. The mobile robot calibrated itself to the workstation using the methods described in Section 3.1.2, and using a tool mounted 2D vision camera it located a number of components. The components where then picked from the conveyor and placed in a fixture on the robot itself for transportation. This task was instructed using a total of 4 unique skills combined into a sequence of 26 steps. Figure A.10c shows the robot picking objects from the conveyor. A description of both the assembly task described above and the part feeding task is found in [57].

3.4. Summary of results

The results from the experiments show, that even for novices in robot programming, the concept of skills affords an applicable task programming framework. Instructing the task using robot skills results in drastically decreased programming time, compared to traditional robot programming methods.

All of the results mentioned in the above sections are summarized in Table A.2. It is clear that the same skills can be used in a variety of task domains. However, there is still some fragmentation of the skill library in the presented work, visible by the number of different skill abstractions (i.e. 4 different versions of the ***pick*** *object* skill). It is also clear that even with a low dimension of the skill library, it is possible to program even advanced tasks, such as the machine tending task described in Section 3.3.2.

	Logistic tasks			**Assistive tasks**
	Transp.	MPF	SPF	Mach.Tend.
Skills				
Pick (4)	✓	✓	✓	✓
Place (4)	✓	✓	✓	✓
Calibrate (3)	✓	✓	✓	✓
Activate (1)			✓	✓
Programming				
Kinesthetic			($4 \rightarrow 26$)	($11 \rightarrow 118$)
Gestures	($3 \rightarrow 8$)			
Planning	($3 \rightarrow 13$)	($4 \rightarrow 16$)		
Coding			($8 \rightarrow 23$)	

Table A.2.: Comparison of results, showing skill implementation and programming paradigms for different task domains (referring to Figure A.2). MPF/SPF is Multiple/Single Part Feeding tasks. The number of abstractions of skills of certain types are indicated in parenthesis. $x \rightarrow y$ denotes that x distinct skills were used to program a complete task, consisting of a total of y skills.

4. Discussion and Future Work

Robot skills have been defined on a conceptual model level, in this work. However, the transition from a conceptual model to its implementation on robotic systems and testing under real industrial conditions is not always straightforward. To the best of our knowledge, approaches similar to ours have never been successfully deployed in real industrial or other scenarios. Thus, we see the main value of our skill concept in its ability, as witnessed by our experiments, to operate robustly and solve real industrial problems.

The conducted experiments show how the same concept can be instantiated to implement seemingly very different types of skills, such as self-calibration or pick and place. Tasks can intuitively be taught by non expert users, as proven in our tests, or automatically generated by regular task-planners. The apparent advantage of our approach is that both users and task-planners do not need to consider the low level motion primitive domain, but rather consider only the limited set of available higher-level skills.

On the other hand, even if our tests have shown that teaching a task using skills can be performed very quickly by regular shop floor workers, the execution of the task by the robot is still very slow. A human performing the same task or a robot explicitly programmed to solve this single task is currently far more efficient. However, we are working in further developing and optimizing the underlaying perception algorithms, which is expected to boost the speed of execution.

Furthermore, we have experienced how tempting it is to over-specialize skills given a certain task; as an example, we ended up having 4 variations of the pick skill. This is exactly due to the fact that these 4 variations are not *general*. This should obviously be avoided, in order to achieve a robot programming framework that is not fragmented with task-specific skill instances. We clearly see that a strict list of general enough skills has to be defined according with the task *domain*, rather than according with specific tasks. Our presented analysis of SOPs and the extraction of a limited number of basic required skills out of them, show that this goal should be considered a requirement in order to avoid fragmentation and overlaps.

Finally, the embedded pre- and post-condition checks of skills allow for timely error handling in complex task, and trigger for replanning or human assistance. This potential has not yet fully explored, but we consider it as very promising research direction that can make skill execution even more robust and reliable.

5. Conclusion

This work has been focused on the developing robotic systems suitable for the envisioned transformable factories of tomorrow. We have presented a conceptual model for object-centered robot skills, that are similar to the abstraction level used when instructing tasks to human workers. We have shown how task-level programming can be combined with our notion of robot skills. This combination effectively acts as a higher abstraction layer, freeing the user from having to specify details such as cartesian coordinates, reference frames, or action specific parameters. The fact that skills are applied on objects, coupled with condition checks in our skills, makes it possible to use very intuitive HRI interfaces, such as kinesthetic teaching and gesture-based teaching for the definition of tasks. As a result, the notion of skills, the way we described and tested it in this work, allows non robot experts to intuitively interact with and program a complex robotic system, such as an industrial mobile manipulator, with only minor training. Finally, as any other system intended for industrial use, robots equipped with our skills have been deployed and tested in real industrial scenarios, showing their robustness and effectiveness. We believe that our robot skills constitute a significant step towards achieving transformable robots, and that such an approach can ultimately increase competitiveness of manufacturing companies.

Acknowledgments

The authors would like to thank Grundfos A/S for the possibility of conducting experiments in the factory. This work has partly been supported by the European Commission under grant agreement numbers FP7-260026 (TAPAS) and FP7-610917 (STAMINA).

References

[1] M. Hartmann, *DYNAPRO: Erfolgreich produzieren in turbulenten Märkten*. Logis Verlag, 1996.

[2] ——, *DYNAPRO II: Erfolgreich produzieren in turbulenten Märkten*. Logis Verlag, 1997.

[3] ——, *DYNAPRO III: Erfolgreich produzieren in turbulenten Märkten*. Logis Verlag, 1998.

[4] R. E. Fikes and N. J. Nilsson, “STRIPS: A new approach to the application of theorem proving to problem solving,” *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1972.

[5] C. Geib, K. Mourao, R. Petrick, N. Pugeault, M. Steedman, N. Krueger, and F. Wörgötter, “Object action complexes as an interface for planning and robot control,” in *IEEE RAS International Conference on Humanoid Robots*, 2006.

[6] R. Brooks, “A robust layered control system for a mobile robot,” *Robotics and Automation, IEEE Journal of*, vol. 2, no. 1, pp. 14 – 23, Mar. 1986.

[7] C. Breazeal and B. Scassellati, “Robots that imitate humans,” *Trends in Cognitive Sciences*, vol. 6, no. 11, pp. 481–487, 2002.

[8] R. Dillmann, “Teaching and learning of robot tasks via observation of human performance,” *Robotics and Autonomous Systems*, vol. 47, no. 2, pp. 109–116, 2004.

[9] M. Lopes and J. Santos-Victor, “A developmental roadmap for learning by imitation in robots,” *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 37, pp. 308–321, Apr. 2007.

[10] S. Schaal, “Is imitation learning the route to humanoid robots?” *TRENDS IN COGNITIVE SCIENCES*, vol. 3, no. 6, p. 10, 1999.

[11] A. Björkelund, L. Edstrom, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. Robertz, D. Storkle, A. Blomdell, R. Johansson, M. Linderoth, A. Nilsson, A. Robertsson, A. Stolt, and H. Bruyninckx, “On the integration of skilled robot motions for productivity in manufacturing,” in *Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on*, May 2011, pp. 1 –9.

[12] C. C. Archibald, “A computational model for skills-oriented robot programming,” PhD Thesis, University of Ottawa, Ottawa, Canada, 1995.

[13] N. Krüger, J. Piater, F. Wörgötter, C. Geib, R. Petrick, M. Steedman, A. Ude, T. Asfour, D. Kraft, D. Omrcen *et al.*, “A formal definition of object-action complexes and examples at different levels of the processing hierarchy,” Technical report, 2009. [Online]. Available: http://www.paco-plus.org

[14] A. Björkelund, H. Bruyninckx, J. Malec, K. Nilsson, and P. Nugues, “Knowledge for Intelligent Industrial Robots,” in *AAAI Spring Symposium on Designing Intelligent Robots: Reintegrating AI, 2012-03-26*. AAAI, 2012.

[15] E. Gat *et al.*, “On three-layer architectures,” *Artificial intelligence and mobile robots*, pp. 195–210, 1998.

[16] T. B. Moeslund, A. Hilton, V. Krüger, and L. Sigal, *Visual Analysis of Humans: Looking at People*, 1st ed. Springer-Verlag New York Inc, 2011.

[17] V. Krüger, D. Kragic, A. Ude, and C. Geib, “The meaning of action: a review on action recognition and mapping,” *Advanced Robotics*, vol. 21, no. 13, pp. 1473–1501, 2007.

[18] A. F. Bobick, “Movement, activity and action: the role of knowledge in the perception of motion.” *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 352, no. 1358, pp. 1257–1265, Aug. 1997.

[19] D. Newtson, G. A. Engquist, and J. Bois, “The objective basis of behavior units,” *Journal of Personality and Social Psychology*, vol. 35, no. 12, pp. 847–862, Dec. 1977.

[20] H.-H. Nagel, “From image sequences towards conceptual descriptions,” *Image Vision Comput.*, vol. 6, no. 2, pp. 59–74, May 1988.

[21] G. Rizzolatti, L. Fadiga, V. Gallese, and L. Fogassi, “Premotor cortex and the recognition of motor actions,” *Brain Research. Cognitive Brain Research*, vol. 3, no. 2, pp. 131–141, Mar. 1996.

[22] M. T. Mason, “Compliance and force control for computer controlled manipulators,” *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 11, no. 6, pp. 418 –432, Jun. 1981.

[23] H. Bruyninckx and J. De Schutter, “Specification of force-controlled actions in the "task frame formalism"-a synthesis,” *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 581 –589, Aug. 1996.

[24] J. De Schutter and H. Van Brussel, “Compliant robot motion i. a formalism for specifying compliant motion tasks,” *The International Journal of Robotics Research*, vol. 7, no. 4, pp. 3–17, Aug. 1988.

[25] T. Kröger, B. Finkemeyer, and F. Wahl, “Manipulation primitives — a universal interface between sensor-based motion control and robot programming,” in *Robotic Systems for Handling and Assembly*, ser. Springer

Tracts in Advanced Robotics, D. Schütz and F. Wahl, Eds. Springer Berlin / Heidelberg, 2011, vol. 67, pp. 293–313.

[26] 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, Jun. 2005.

[27] T. Kröger, B. Finkemeyer, S. Winkelbach, L.-O. Eble, S. Molkenstruck, and F. Wahl, "A manipulator plays jenga," *Robotics Automation Magazine, IEEE*, vol. 15, no. 3, pp. 79 –84, Sep. 2008.

[28] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. M. M. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, "RoboEarth," *IEEE Robotics Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.

[29] M. Tenorth, A. Perzylo, R. Lafrenz, and M. Beetz, "Representation and exchange of knowledge about actions, objects, and environments in the RoboEarth framework," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 643–651, 2013.

[30] P. Neto and N. Mendes, "Direct off-line robot programming via a common CAD package," *Robotics and Autonomous Systems*, vol. 61, no. 8, pp. 896–910, Aug. 2013.

[31] H. Chen and W. Sheng, "Transformative CAD based industrial robot program generation," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 5, pp. 942–948, 2011.

[32] S. Mitsi, K.-D. Bouzakis, G. Mansour, D. Sagris, and G. Maliaris, "Off-line programming of an industrial robot for manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 26, no. 3, pp. 262–267, 2005.

[33] D. Gadensgaard and D. Bourne, "Human/robot multi-initiative setups for assembly cells," in *ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems*, 2011, pp. 1–6.

[34] Y. H. Jen, Z. Taha, and L. J. Vui, "VR-based robot programming and simulation system for an industrial robot," *International Journal of Industrial Engineering: Theory, Applications and Practice*, vol. 15, no. 3, pp. 314–322, 2008.

[35] R. Bischoff and A. Kazi, “Perspectives on augmented reality based human-robot interaction with industrial robots,” in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 4, 2004, pp. 3226–3231.

[36] M. Ralph and M. A. Moussa, “Toward a natural language interface for transferring grasping skills to robots,” *Robotics, IEEE Transactions on*, vol. 24, no. 2, pp. 468–475, 2008.

[37] J. Blume, A. Bannat, G. Rigoll, M. Rooker, A. Angerer, and C. Lenz, “Programming concept for an industrial HRI packaging cell,” in *2013 IEEE RO-MAN*, Aug. 2013, pp. 93–98.

[38] J. Blume, “iProgram: Intuitive programming of an industrial HRI cell,” in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Mar. 2013, pp. 85–86.

[39] C. Fitzgerald, “Developing baxter,” in *IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, April 2013, pp. 1–6.

[40] S. Gottschlich, C. Ramos, and D. Lyons, “Assembly and task planning: A taxonomy,” *Robotics & Automation Magazine, IEEE*, vol. 1, no. 3, pp. 4–12, 1994.

[41] G. Biggs and B. MacDonald, “A survey of robot programming systems,” in *Proceedings of the Australasian conference on robotics and automation*, 2003, pp. 1–3.

[42] H. Mosemann and F. Wahl, “Automatic decomposition of planned assembly sequences into skill primitives,” *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 5, pp. 709 –718, Oct. 2001.

[43] D. Hähnel, W. Burgard, and G. Lakemeyer, “GOLEX—bridging the gap between logic (GOLOG) and a real robot,” *KI-98: Advances in Artificial Intelligence*, pp. 165–176, 1998.

[44] A. Ferrein and G. Lakemeyer, “Logic-based robot control in highly dynamic domains,” *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 980–991, Nov. 2008.

[45] S. Bøgh, M. Hvilshøj, M. Kristiansen, and O. Madsen, “Identifying and evaluating suitable tasks for autonomous industrial mobile manipulators (AIMM),” *The International Journal of Advanced Manufacturing Technology*, vol. 61, no. 5-8, pp. 713–726, Jul. 2012.

[46] M. Hvilshøj, S. Bøgh, O. S. Nielsen, and O. Madsen, "Autonomous industrial mobile manipulation (AIMM): past, present and future," *Industrial Robot: An International Journal*, vol. 39, no. 2, pp. 120–135, Feb. 2012.

[47] S. Bøgh, O. S. Nielsen, M. R. Pedersen, V. Krüger, and O. Madsen, "Does your robot have skills?" in *Proceedings of the 43rd International Symposium on Robotics (ISR)*, Taipei, Taiwan, Aug. 2012.

[48] R. Andersen, L. Nalpantidis, V. Krüger, O. Madsen, and T. Moeslund, "Using robot skills for flexible reprogramming of pick operations in industrial scenarios," in *VISAPP 2014 - Proceedings of the 9th International Conference on Computer Vision Theory and Applications*, vol. 3, 2014, pp. 678–685.

[49] M. R. Pedersen, L. Nalpantidis, A. Bobick, and V. Krüger, "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. [Online]. Available: http://renaud-detry.net/events/crs2013/papers/Pedersen.pdf

[50] M. R. Pedersen, "Integration of the KUKA light weight robot in a mobile manipulator," Master's Thesis, Aalborg University, Aalborg, Denmark, 2011.

[51] M. Hvilshøj, S. Bøgh, O. Madsen, and M. Kristiansen, "Calibration techniques for industrial mobile manipulators: Theoretical configurations and best practices," in *Proceedings for the joint conference of ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*. VDE Verlag GMBH, 2010.

[52] R. Andersen, J. Damgaard, O. Madsen, and T. Moeslund, "Fast calibration of industrial mobile robots to workstations using QR codes," in *Proceedings of the 44th International Symposium on Robotics ISR 2013*. IEEE Press, 2013.

[53] C. Schou, J. Damgaard, S. Bøgh, and O. Madsen, "Human-robot interface for instructing industrial tasks using kinesthetic teaching," in *44th International Symposium on Robotics (ISR)*. IEEE, 2013.

[54] M. R. Pedersen, D. Herzog, and V. Krüger, "Intuitive skill-level programming of industrial handling tasks on a mobile manipulator," *2014*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2014.

[55] M. Fox and D. Long, "PDDL2. 1: An extension to PDDL for expressing temporal planning domains." *J. Artif. Intell. Res.(JAIR)*, vol. 20, pp. 61–124, 2003.

[56] M. Hvilshøj, S. Bøgh, O. Nielsen, and O. Madsen, "Multiple part feeding – real-world application for mobile manipulators," *Assembly Automation*, vol. 32, no. 1, pp. 62–71, 2012.

[57] S. Bøgh, C. Schou, T. Rühr, Y. Kogan, A. Dömel, M. Brucker, C. Eberst, R. Tornese, C. Sprunk, G. Tipaldi, and T. Hennessy, "Integration and assessment of multiple mobile manipulators in a real-world industrial production facility," in *Proceedings for the joint conference of ISR 2014, 45th International Symposium on Robotics and Robotik 2014, 8th German Conference on Robotics*. VDE Verlag GMBH, 2014, pp. 305–312.

Paper B

On the Integration of Hardware-Abstracted Robot Skills for use in Industrial Scenarios

Mikkel Rath Pedersen, Lazaros Nalpantidis,
Aaron Bobick, Volker Krüger

The paper has been published online in the
Proceedings of the 2nd International IROS Workshop on Cognitive Robotics Systems: Replicating Human Actions and Activities, 2013.

The layout has been revised.

Abstract

In this paper, we present a method for programming robust, reusable and hardware-abstracted robot skills. The goal of this work is to supply mobile robot manipulators with a library of skills that incorporate both sensing and action, which permit robot novices to easily reprogram the robots to perform new tasks. Critical to the success of this approach is the notion of hardware abstraction, that separates the skill level from the primitive level on specific systems. Leveraging a previously proposed architecture, we construct two complex skills by instantiating the necessary skill primitives on two very different mobile manipulators. The skills are parameterized by task level variables, such as object labels and environment locations, making re-tasking the skills by operators feasible.

1. Introduction

Manufacturing companies are currently experiencing a paradigm shift from mass production to mass customization; therefore, current production equipment and entire production lines need to be adjusted to follow this development, including industrial robots and automation equipment. For highly customized products, or product portfolios with large variations, a lot of manual labor is usually present. Human labor is the pinnacle of versatility when it comes to manufacturing, as a skilled worker can be shown new tasks and in most cases immediately reproduce them. In other words, if a human has a basic set of *skills*, then the repertoire of *tasks* the he/she can perform is impressively large. It is for this reason that we propose a similar structure for the mobile industrial manipulators of the future, as it is necessary to have methods for effortless reprogramming of robots at the *task level*, rather than the *robot level*. In task level programming, a robot program is a sequence of robot *skills* that specify a production-related goal.

To establish a fitting representation of robot skills for task-level programming has been the focus of research around the world [1–7]. The various concepts of skills is quite different, but most of them are composed of primitive, formal descriptions of sets of robot motions, called action or motion *primitives*. These primitives are simple, atomic robot movements that can be combined to form more complex behavior [8–13], which is often called a robot *skill*. As such, no single, unified definition of a skill in terms of robotics exists. In this paper, we present our model of a robot skill, that has been previously discussed in [14], and show how these skills can be implemented and used, using a few skills as examples.

In contrast to the skills themselves, the *skill primitives* [5–7] are rather well defined in the robotics community; although several different descriptions exist, most of them loosely follow Mason's work on compliant controllers [15], which paved the ground for the Task Frame Formalism (TFF) introduced by Bruyninckx and De Schutter [16, 17]. Recent work has expanded upon this idea, for instance enabling the use of any sensor and controller in the same task frame (e.g. visual servoing combined with force control), and operating in a dynamic task frame [18–20].

The concept of skills presented in this paper is comparable to the *manipulation primitive nets*, described best in [19]. These nets are sequences of *manipulation primitives*, with simple decisions based on the outcome of each single manipulation primitive in the net. In these nets, however, each manipulation primitive needs its own explicitly specified parameters, e.g. a specific force or velocity in a specific direction. This makes this particular implementation unsuitable for robotics novices in a factory hall. Instead, we propose a method for specifying parameters on a higher level, the skill level, and instead letting the system infer the parameters for the lower level primitives.

In this paper we will show our skill paradigm applied for two of the most important robot skills for manipulation - the *pick up* and *place* skills. These are very complex skills, in that they require a high degree of both sensing and motion to be sufficiently general. Being sufficiently general, they can also be used in almost any task, be it machine tending, part feeding, etc. In order to further show the capabilities of our approach, we show that the skills can easily be ported to a totally different robot, and can be sequenced to program a robot task. As our experimental platforms we use a commercially available PR2 robot and a custom industrial mobile manipulator, the Little Helper.

In section 2, we introduce the reader to the concept of skills used in this work, along with a brief explanation of the skill primitives. In section 3, we will present the outline of two skills, as well as the necessary skill primitives for these. After this, we present the implementation of these skills on two very different robots in section 4. We will briefly present the roadmap for future work and conclude this paper in sections 5 and 6.

2. Conceptual overview of skills and tasks

We will now introduce the definitions of skills and tasks used in this work. We present a layered structure, where tasks are the highest abstraction layer, and also the layer that is easiest to understand for robotics novices. Each task is composed of a sequence of skills, each of which is composed of a number of skill primitives, where the primitive layer is the most robot-oriented layer. See also Fig. B.1. We will start by explaining the concept of a task.

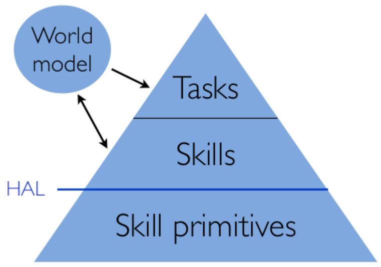


Fig. B.1.: The three layers of the skill model. The skills can access and alter information in the world model, and the tasks can only access the information, and alters it through the skills. The only hardware-specific parts are the skill primitives, so the hardware abstraction layer (HAL) is between the primitives and the skills.

We define tasks as sequences of robot skills, and the sequences can either be pre-programmed by an operator or generated by a task-level planner, either offline or ad hoc. Tasks are characterized by a relation to a production-related goal in the factory; for instance a machine tending task could be *insert part P in machine M, start the machine, remove part, place on workpiece tray*. The basis for a given task is a set of known state variables, such as locations of known objects, current state of production equipment, etc. The task uses skills to change these state variables to a desired goal setting.

We define a robot skill as a fundamental software building block, that incorporates both sensing and action. In order for the skills to be useful for task-level programming, they must both change the world state through sensing or action [7], and be self-sustained, so they can be used in any task. Self-sustainability implies that each skill should be

- parametric in its execution, so it will perform the same basic operation regardless of input parameter,
- able to estimate if the skill can be executed based on the input parameter and world state, and
- able to verify whether or not it was executed successfully.

A graphical representation of a skill, implementing all of these aspects, is shown in Fig. B.2.

The skill uses the current world state as input, along with a parameter, which is provided at task programming time. As an example, for the *pick*

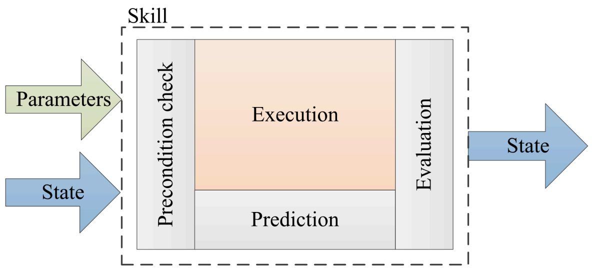


Fig. B.2.: Model of a robot skill

object skill the parameter is the specific object that needs to be picked up. The skill initially checks that all preconditions for executing the skill are satisfied, based on the inputs. If true, the skill executes the sequence of skill primitives that changes the world state to the desired goal setting. After execution, it is verified that the current, measured state variables are satisfactory. This is done by comparing them to the *prediction* of the outcome (see Fig. B.2), which is established from the parameter input and initial state setting.

Pre- and postconditions are required for both robustness and planning. There is no reason to attempt to execute a skill if the preconditions are not satisfied, and the skill is only correctly executed if the postconditions are satisfied. Furthermore, if there exists a task-level planner that is able to create tasks as sequences of skills, the same planner would be able to deal with precondition failures, by planning a sequence of skills that satisfies the preconditions. In the case of postcondition failures, however, the case is not as trivial, since a planner would still utilize the same library of skills, resulting in a similar task as the one currently being executed. In this case it might be feasible to simply retry the skill a few times, call an operator, or render the skill invalid for the next planning attempt.

Using these definitions of tasks and skills, and based on our findings in the FP7 project TAPAS and numerous experiments on the shop floor, a total of 566 manual tasks in a production facility have been analyzed [21], and we have shown that it is possible to break down most of these tasks into a set of only 13 distinct robot skills, and even less in simpler domains [14].

2.1. Skill primitives

In order for the robot skills to be robust, hardware-abstracted and easy to implement for robot skill developers, they must themselves be composed of lower-level skill primitives. We will not attempt to provide a strict definition of what a skill primitive is and what it is not. Instead, we see skill primitives as traditional macros, that perform a basic function directly which may be readily provided by the robot system. Comparing to skill primitives in humans, they are intuitively the motions and sensing we perform without thinking further about it. As such, a robot skill primitive could be as rudimentary as closing the gripper on the robot, or as advanced as planning and performing collision-free motion of a robot arm.

Being the only interface to the hardware in this model, these components are also effectively the hardware abstraction layer. Ideally, this means that any single skill primitive can be changed, as long as the same interfaces are maintained.

It is important to note that, for the sake of generality, we do not distinguish between skill primitives that purely perform motions and primitives that purely perform sensing. For example, both an object detector and a point-to-point robot arm motion are skill primitives.

We will now go on to present the implementation of the pick and place skills.

3. Implementation overview

This section will present the challenges in implementing the pick and place skills, which is the initial implementation of skills following the concepts mentioned above. As such, we divide this section into two subsections. First, we will briefly present the outline and contents of both skills, following the model introduced in 2. After this, we present a general overview of both skills, along with aspects of the implementation that is necessary for both. In this section we will also introduce the implementation of the skill primitives that are the same for both robots.

3.1. Skill descriptions

The pick skill aims to fulfill the goal of picking up a *known* object in the world model. The skill is designed to assume that the object is within reach of the robot, without the need of driving to the object location. Knowing that the object is within reach, the robot will re-detect the object, to get an accurate and up to date object pose. After this, it will pick up the object, and lift it away from the table. This ends the execution phase of the skill. It will then need to be verified that the execution was successful, which is done by *a)* checking if an object is in the gripper, and *b)* no object is at the previous location. The steps involved are shown in Table B.1, and the skill primitives used are marked with the abbreviation *SP*.

Parameter	Previously detected object
Preconditions	Object known in world model Object within reach Gripper empty
Execution	*SP:* Redetect and update pose of object *SP:* Open gripper Calculate pregrasp and grasp pose *SP:* Planned, collision-free arm motion to pregrasp pose *SP:* Motion into grasp pose *SP:* Close gripper *SP:* Motion away from table
Postconditions	Object in gripper Object not at previous location

Table B.1.: Pick skill

Similarly to the pick skill, the place skill assumes the position to place the object is within reach of the robot arm. The goal of the place skill in this implementation is to put down the object in the gripper at a specific location on a known surface. A robust and general place skill, that can handle advanced contact definitions, e.g. assembly tasks, is out of scope of this particular work. The robot will have to move the gripper to a position above the desired final position of the object, after which it will move down, normal to the table, until contact. After this, it will open the gripper and move it away from the object, which concludes the execution phase. Finally, it is verified that the gripper is empty, and the object is placed at the desired position. The steps involved in the place skill are shown in Table B.2.

3.2. General implementation

Certain prerequisites in the form of skill primitives need to be satisfied in order to implement the skills on the robots. Besides simple motion commands, such as gripper motions, the relatively advanced skill primitives necessary for this implementation are:

- Collision-free arm navigation
- Object detection, recognition and grasp planning

Parameter	(x, y) location on previously detected surface
Preconditions	Surface known in world model Place location within reach Object in gripper Place location empty
Execution	Calculate pre-place and place poses *SP:* Planned, collision-free arm motion to pregrasp pose *SP:* Motion down to contact with surface *SP:* Open gripper *SP:* Motion away from object
Postconditions	Gripper empty Object at correct location Gripper is away from object

Table B.2.: Place skill

- Cartesian motions with force feedback

Furthermore, it is necessary to have a suitable representation of the world state, which we have implemented as well. The world model in this initial implementation contains information about previously detected objects and surfaces, in global coordinates. This implementation also enables advanced querying (e.g. *return objects currently located on table 1*) and updating poses of surfaces and objects ad hoc. When starting up the robot, we perform an initial scan of the scene, and save information regarding objects and surfaces, or load a previously saved world state.

The current implementation relies heavily on available ROS packages for the primitives. However, due to the layered architecture, the primitive layer could in principle be any other robot middleware, with a package-like structure.

For the object and surface detection skill primitive, we use the ROS package `tabletop_object_detector`, which performs tabletop segmentation, object detection and object recognition based on measured point clouds of the environment, captured with a Kinect camera. This package is furthermore utilizing a database of known objects, with a set of simulated grasp poses for the PR2 parallel gripper for each object. For the purpose of example, this package is adequate, and can easily be exchanged with a more suitable primitive when needed.

In order for the robot to safely manipulate objects in semi-structured environments, such as production facilities, a skill primitive that performs collision-free arm motions is also required. In order to do this, we use the ROS stack `arm_navigation`, that is fully integrated to work with the PR2.

It is not always feasible to plan every motion of a robot arm, especially not during manipulation of parts for which we do not have a model we can input in the collision map. Therefore, a skill primitive that enables motions according to the Task Frame Formalism has been implemented. This primitive is a layer on top of a low level controller, that controls the joints of the robot arm directly. Therefore, the TFF primitive is not performing real-time control of the joints of the robot arm directly, and is therefore strictly not a TFF controller. However, this primitive does enable us to specify the desired position, velocity or applied force in any direction and in any known frame, until a certain stop criterion is met, e.g. a traveled distance, a velocity, or an external force in a particular direction.

We will give a more detailed description of the implementation in the next sections.

4. Implementation on two robots

We begin by presenting the implementation on the PR2 robot, which has essentially served as an initial prototyping platform for the skill concept introduced in 2. In this description, we will present how each skill primitive was implemented, and how it fits into the skill itself. Finally, we describe the necessary steps in order to transfer this implementation from the research platform, that the PR2 is, to an entirely different robot, the industrial mobile robot Little Helper.

The skills have all been implemented as classes in Python, that inherit from a base skill class. This base class contains methods for adding pre- or postconditions, as well as evaluating all conditions. It also contains an empty method for interfacing to the ROS topics and actions, that are used in the specific skill implementation, and an empty method for execution. When implementing a skill, the programmer must overwrite these two functions with skill-specific ones, as well as add pre- and postconditions to the skill. In this way, it is ensured that all skills maintain the same vocabulary for calling the functions within the skill.

4.1. Implementation on the PR2

The following two sections provide a detailed description of the implementation of the skills, according to the outline in Table B.1 and B.2, respectively.

4.1.1. Pick skill

As a unique ID identifies each object in the world model, this ID is used as the parameter for the pick skill. The world model is queried for the information regarding this particular object, e.g. the type of object, pose, and surface it is resting upon. The preconditions *object within reach* and *object known in world model* can be determined based on this query. The precondition check for the gripper being empty is tested on the internal joint state of the robot. If any of the preconditions fail, the skill terminates and outputs which precondition(s) failed.

In the execution phase, it is first necessary to redetect the scene by calling the object detector primitive, to accurately update the object pose in the robot coordinate system. This is due to the fact that the world model for a mobile robot will be populated with object poses in world coordinates, and current navigation and localization are not sufficiently precise for manipulation.

The grasp pose is selected based on a series of presimulated grasps, available for the specific object type in the database of objects. The chosen grasp is a selected as a tradeoff between the pose difference between the current gripper pose and the available grasp poses, and the success probability from grasp simulations. The pre-grasp pose is similar to the grasp pose, with an offset in the gripper frame, to ensure a collision-free approach to the object.

The end-effector is moved to the pre-grasp pose, by using the arm navigation primitive. The planner used is the SBL planner [22] from the Open Motion Planning Library [23].

In order to approach the object, the robot makes use of the TFF skill primitive, and performs a TFF motion into the grasp pose. This motion is specified as a velocity in the end-effector frame, towards the grasp pose, and the stop criterion is either the approach distance used to calculate the pre-grasp pose, or when sensing a contact with the object. The TFF primitive outputs desired positions to a cartesian position controller with force and velocity feedback, the `JTTaskController` in the `pr2_manipulation_controllers` package. A separate PI control of velocity and force has been implemented, where a desired pose is sent to the cartesian controller, based on the velocity or force feedback directly from the controller, respectively. After the motion reaches the stop criterion, the gripper is closed to grasp the object. The pose of the object in the world model is then updated to the current pose in the gripper frame, and a flag is set to indicate that this object is currently in the gripper. The collision model of the object is also attached to the arm, to enable future planned motions with the object in the gripper.

The TFF motion performing the lift is also a velocity command, with the stop criterion being a pre-programmed lifting height. This motion is carried out in a direction normal to the surface, to ensure there is no collision with the surface.

If any of the previous steps fails, the skill terminates and outputs at exactly which step it failed, and for what reason.

After the execution the postconditions are verified. Checking if the object is in the gripper is tested on the current gripper joint configuration, and comparing this to the known joint configuration for the chosen grasp. The detection primitive is called again, in order to ensure no objects are present at the location the object was picked from. If both postconditions are satisfied, the skill is concluded, and the robot returns to a waiting state.

4.1.2. Place skill

Similar to the objects, detected surfaces are identified in the world model with a unique ID. The parameter to the place skill is therefore the ID of the surface, and the (x, y) location to place the object on that surface. In this implementation, the coordinate is explicitly specified when calling the skill, but initial experiments with simple gesture recognition for parameter input show that this is a feasible approach [24]. The first precondition checks are whether or not the surface is known in the world model, and if the desired location is within the workspace of the robot, similar as the checks in the *pick* skill. The world model is also queried to obtain knowledge of the object in the gripper, based on the flag set after picking up the object. Calling the object detection primitive reveals whether or not there already is an object at the specified location, in which case that particular precondition fails.

The execution phase initially calculates the place pose from the (x, y) location in the table frame, specified in the parameter. This is the desired final pose of the object, and since the object pose in the gripper is known from the world model, the final pose of the gripper is easily deduced. The pre-place pose is an offset of the place pose in the direction normal to the surface.

We again use the arm navigation primitive to conduct a planned motion to the pre-place pose. In this way we can still avoid collision with both the arm and the object in the gripper, since the object was added to the collision model of the arm. Upon reaching the pre-place pose, the TFF primitive is used to move the grasped object toward the surface, in the direction of the surface normal. This motion is terminated once a contact is sensed with the table, and the gripper is opened. The collision model of the object is now detached from the arm, as it is no longer needed.

In order to avoid colliding with the object when moving away from it, this is also carried out as a TFF motion in the gripper frame, similar to the approach motion in the pick skill. The stop condition is a distance greater than the length of the gripper fingers, to ensure the gripper is clear from the object.

After the gripper is moved away, the pose of the object is updated to be the desired place pose. This is then verified as a post-condition by re-detecting the scene, and if the object is within an acceptable placing tolerance, the pose of the object is updated to be the actual, detected pose. The checks if the gripper is empty (fully open) and away from the object are measured based on the internal robot state.

4.2. Implementation on the Little Helper

The Little Helper (see Fig. B.3) relies on a KUKA Light Weight Robot (LWR) arm for manipulation, which is not fully compatible with ROS. Furthermore, the system consists of a Schunk WSG-50 parallel gripper for manipulation, and a Neobotix MP-L655 mobile differential drive platform for mobility. Both of these latter components are fully integrated with ROS.

The primary tasks in porting the skills to the Little Helper are to implement missing primitives, modifying the interfaces with the primitives, and to make adjustments in the various geometric calculations within the skill. This is primarily calculating gripper poses, but also mapping the joint configurations of a grasp with the PR2 gripper to that of the gripper on the Little Helper. Considering nearly no attention to hardware abstraction was taken when initially developing the skills on the PR2, the skills were working on the Little Helper within a very short time and with low effort. If the skills had initially been developed in a generic manner, it would have taken even less time and work.

On the Little Helper, we are using a skill primitive that sends cartesian or joint poses directly to the KUKA controller, which then handles inverse kinematics, path planning and joint interpolation. This has the drawback that it is not possible to perform collision-free arm navigation and TFF-like motions. It is, however, very important to again stress the fact that the interfaces to the different primitives are the same. So when the arm navigation and TFF skill primitives are working on the Little Helper, they can directly be used by the skills. This shows how easily skill primitives can be exchanged, as long as

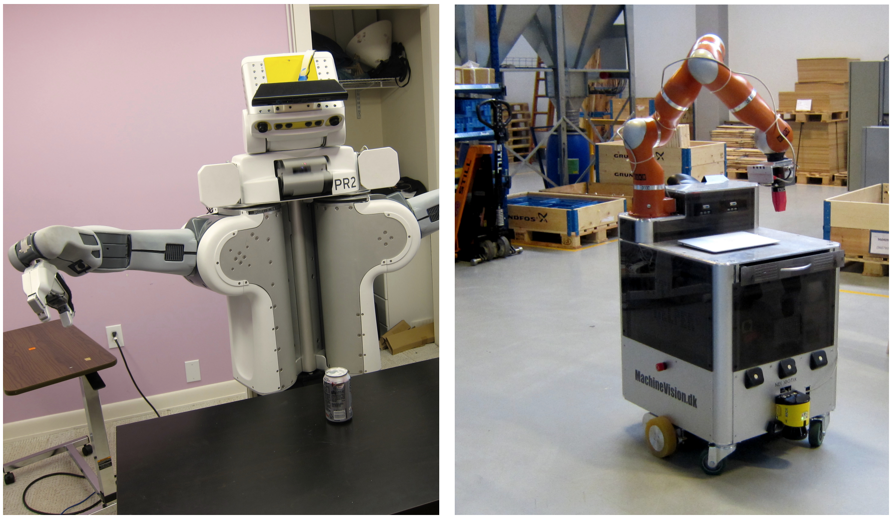

(a) (b)

Fig. B.3.: The two robots used for experiments: (a) the PR2 robot and (b) the Little Helper robot at a production facility

their interfaces are the same.

In the initial experiments with the PR2, we have used the robot in a stationary position. On the Little Helper, however, we have implemented one more skill, which is the simple *drive to station* skill. This skill uses the ROS navigation system to navigate the mobile base in a known map to a workstation that is previously saved in the world model. This skill only has the precondition check of whether or not the station is known in the model. The skill will then call the navigation primitive, in order to bring the mobile robot from the current position to the saved workstation, and check that the robot is at the goal.

Using these three skills, we can easily create small scripts, that is essentially task descriptions. In the following is shown a simple function that calls the skills to get a box of rotor caps, for rotors used in a domestic water pump, and transport them to a rotor press, where the final rotor is assembled:

```
def get_box_of_rotor_caps():
  # Drive to rotor cap press
  drive_to_station('warehouse')
  # Get ID of a box of rotors near the robot
  box_id = get_object_id('full_rotor_box',near='warehouse')
  # Pick up a box of rotors
  pick_up(box_id)
  # Place on robot platform at a specific position
  place('platform',0.30,0.55)
```

```
# Drive to the rotor assembly station
drive_to_station('rotor_press_station')
# Pick up the box from the platform
pick_up(box_id)
# Place the box on the table next to the press
place('rotor_press_table',1.40,0.25)
return
```

This is only the very basic way of using skills to program tasks, and being able to specify robot tasks as simple as the example above opens up the door to a lot of different possibilities in Human-Robot Interaction, that can output scripts like this.

5. Future work

It is our intention to further develop the library of skills, with the 10 (less advanced) skills that are missing for the logistic domain described in [14]. We will do this only on the Little Helper. When we have a sufficient library of skills, we will use them for improved human-robot interaction, so an operator can select the sequence of skills, and their parameters, in a GUI or through direct interaction with the robot. It is imperative that this will be extensively tested in a real production facility. Furthermore, we will experiment with using simple planners to create sequences of skills, and more advanced ones to deal with skill failures, both as pre- and postcondition failures, and during execution.

6. Conclusion

In this paper we have presented a model for programming robust, hardware-abstracted robot skills, for use in task-level programming, following a structure similar to how we humans use basic skills to form more complex tasks. One can argue that the implementation has a number of shortcomings, such as requiring a database of known objects with corresponding grasps. However, we would like to argue that the modular concept of the skills allows to very easily change such components to accommodate the needs of the different scenarios. Our skill model can easily accommodate future extensions, both in skill primitives and world state knowledge. In principle, this concept looks very promising, because the skills are self-sustained, in that they know when they fail and why, and are parametric in their execution. But also because the skills are highly modular, since they are composed of lower level skill primitives. Getting a skill, explicitly designed for one robot, to work on another robot proved to be effortless, given the right skill primitives. However, having the right skill primitives readily available can be the limiting factor in a skill implementation, if the skill is to be implemented in the exact same manner on all robots.

For demonstrations of the described implementations, see the attached video.

ACKNOWLEDGMENTS

The authors would like to thank the Center for Robotics and Intelligent Machines (RIM) at Georgia Institute of Technology for the opportunity of using the PR2 for experiments. Especially, we would like to thank Tucker Hermans from RIM for always being helpful.

This work has partly been supported by the European Commission under grant agreement number FP7-260026-TAPAS.

References

[1] M. Lopes and J. Santos-Victor, “A developmental roadmap for learning by imitation in robots,” *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 37, pp. 308–321, Apr. 2007.

[2] R. Brooks, “A robust layered control system for a mobile robot,” *Robotics and Automation, IEEE Journal of*, vol. 2, no. 1, pp. 14 – 23, Mar. 1986.

[3] C. Breazeal and B. Scassellati, “Robots that imitate humans,” *Trends in Cognitive Sciences*, vol. 6, no. 11, pp. 481–487, 2002.

[4] R. Dillmann, “Teaching and learning of robot tasks via observation of human performance,” *Robotics and Autonomous Systems*, vol. 47, no. 2, pp. 109–116, 2004.

[5] S. Schaal, “Is imitation learning the route to humanoid robots?” *TRENDS IN COGNITIVE SCIENCES*, vol. 3, no. 6, p. 10, 1999.

[6] A. Björkelund, L. Edstrom, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. Robertz, D. Storkle, A. Blomdell, R. Johansson, M. Linderoth, A. Nilsson, A. Robertsson, A. Stolt, and H. Bruyninckx, “On the integration of skilled robot motions for productivity in manufacturing,” in *Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on*, May 2011, pp. 1 –9.

[7] N. Krüger, J. Piater, F. Wörgötter, C. Geib, R. Petrick, M. Steedman, A. Ude, T. Asfour, D. Kraft, D. Omrcen *et al.*, "A formal definition of object-action complexes and examples at different levels of the processing hierarchy," Technical report, 2009. [Online]. Available: http://www.paco-plus.org

[8] T. B. Moeslund, A. Hilton, V. Krüger, and L. Sigal, *Visual Analysis of Humans: Looking at People*, 1st ed. Springer-Verlag New York Inc, 2011.

[9] V. Krüger, D. Kragic, A. Ude, and C. Geib, "The meaning of action: a review on action recognition and mapping," *Advanced Robotics*, vol. 21, no. 13, pp. 1473–1501, 2007.

[10] A. F. Bobick, "Movement, activity and action: the role of knowledge in the perception of motion." *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 352, no. 1358, pp. 1257–1265, Aug. 1997.

[11] D. Newtson, G. A. Engquist, and J. Bois, "The objective basis of behavior units," *Journal of Personality and Social Psychology*, vol. 35, no. 12, pp. 847–862, Dec. 1977.

[12] H.-H. Nagel, "From image sequences towards conceptual descriptions," *Image Vision Comput.*, vol. 6, no. 2, pp. 59–74, May 1988.

[13] G. Rizzolatti, L. Fadiga, V. Gallese, and L. Fogassi, "Premotor cortex and the recognition of motor actions," *Brain Research. Cognitive Brain Research*, vol. 3, no. 2, pp. 131–141, Mar. 1996.

[14] S. Bøgh, O. S. Nielsen, M. R. Pedersen, V. Krüger, and O. Madsen, "Does your robot have skills?" in *Proceedings of the 43rd International Symposium on Robotics (ISR)*, Taipei, Taiwan, Aug. 2012.

[15] M. T. Mason, "Compliance and force control for computer controlled manipulators," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 11, no. 6, pp. 418 –432, Jun. 1981.

[16] H. Bruyninckx and J. De Schutter, "Specification of force-controlled actions in the "task frame formalism"-a synthesis," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 581 –589, Aug. 1996.

[17] J. De Schutter and H. Van Brussel, "Compliant robot motion i. a formalism for specifying compliant motion tasks," *The International Journal of Robotics Research*, vol. 7, no. 4, pp. 3–17, Aug. 1988.

[18] T. Kröger, B. Finkemeyer, and F. Wahl, "Manipulation primitives — a universal interface between sensor-based motion control and robot programming," in *Robotic Systems for Handling and Assembly*, ser. Springer Tracts in Advanced Robotics, D. Schütz and F. Wahl, Eds. Springer Berlin / Heidelberg, 2011, vol. 67, pp. 293–313.

[19] 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, Jun. 2005.

[20] T. Kröger, B. Finkemeyer, S. Winkelbach, L.-O. Eble, S. Molkenstruck, and F. Wahl, "A manipulator plays jenga," *Robotics Automation Magazine, IEEE*, vol. 15, no. 3, pp. 79 –84, Sep. 2008.

[21] S. Bøgh, M. Hvilshøj, M. Kristiansen, and O. Madsen, "Identifying and evaluating suitable tasks for autonomous industrial mobile manipulators (AIMM)," *The International Journal of Advanced Manufacturing Technology*, vol. 61, no. 5-8, pp. 713–726, Jul. 2012.

[22] G. Sánchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Robotics Research*, R. A. Jarvis and A. Zelinsky, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 6, pp. 403–417.

[23] Kavraki, Lydia E., "The open motion planning library (OMPL)," 2010. [Online]. Available: http://ompl.kavrakilab.org

[24] M. R. Pedersen, C. Høilund, and V. Krüger, "Using human gestures and generic skills to instruct a mobile robot arm in a feeder filling scenario," in *Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA)*, Chengdu, Sichuan, China, Aug. 2012.

Paper C

Intuitive Skill-Level Programming of Industrial Handling Tasks on a Mobile Manipulator

Mikkel Rath Pedersen, Dennis Herzog, Volker Krüger

Abstract

In order for manufacturing companies to remain competitive while also offering a high degree of customization for the customers, flexible robots that can be rapidly reprogrammed to new tasks need to be applied in the factories. In this paper we propose a method for the intuitive programming of an industrial mobile robot by combining robot skills, a graphical user interface and human gesture recognition. We give a brief introduction to robot skills as we envision them for intuitive programming, and how they are used in the robot system. We then describe the tracking and gesture recognition, and how the instructor uses the method for programming. We have verified our approach through experiments on several subjects, showing that the system is generally easy to use even for inexperienced users. Furthermore, the programming time required to program a new task is very short, especially keeping traditional industrial robot programming methods in mind.

1. Introduction

Manufacturing companies have experienced the challenges of mass customization for a long period of time. As customers demand more customization options for products, traditional mass production becomes problematic. While mass production focuses on producing large volumes of identical items with high efficiency, mass customization needs to maintain the production volume, but also offer products with a highly varied feature set. In order to do this, flexible robotic solutions appear more promising than stationary robot cells.

A key issue of flexible robots is to be able to reprogram them on the fly when a new task is needed in the factory. Currently, industrial robots are programmed by experts to do specific motions and follow specific paths in a highly structured environment. In order for the robots to be truly flexible, they need to be able to handle a variety of different tasks, and they must be reprogrammed fast when a new task in the factory arises.

In this paper, we propose a method for intuitive and fast programming of a mobile industrial robot using robot *skills* in conjunction with a graphical user interface and human gestures. Using the skills as fundamental building blocks, it is easy for a shop floor worker to form complex *tasks*. The skills need to be supplied with parameters, stating on which objects they should be executed, e.g. the `Pick up <object>` skill needs to know which object to pick up. The sequencing of skills is done on a touchpad device, and the teaching of the parameters for the individual skills are done by performing specific gestures, such as pointing at objects to pick up.

The underlying principles of the skills are very general, and can be applied to any robotic system. We have previously shown that a set of skills developed explicitly for one robot can with very little effort be implemented on a completely different robot system [1]. It is our firm belief that this is the direction industrial robotics need to go in order for manufacturing companies to remain competitive in the future.

1.1. Scenario description

For validating our approach, we have chosen real industrial tasks, that are currently performed by human workers in the factory of Grundfos, a Danish manufacturer of domestic water circulation pumps. The use cases we want to be able to program fast and intuitively are handling (or logistic) tasks, such as tending a workstation assembling rotors for water pumps. In this case, the finished rotors are placed in a small kanban box next to the workstation. When the box is full of finished parts, it needs to be replaced with an empty box. It is the transporting of this box to a warehouse and the fetching of an empty one that is the considered use case for this paper.

Programming this task using current programming methods is neither effortless, nor intuitive. In our case, we select the appropriate skills that solve the task on a touchpad, and specific objects to pick up or locations to place on are indicated by pointing at them. Fig. C.1 shows a user with a touchpad, pointing at a location to place a box. Using this method, the programming can be done in as little as two minutes, and the program is working under the usual location uncertainties that are present in a human environment.

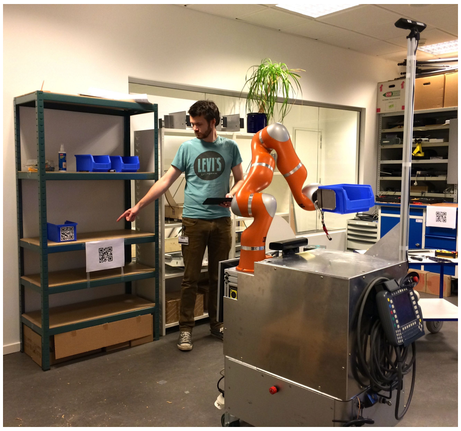


Fig. C.1.: A user with a touchpad pointing at a location to place a box in front of the robot.

The contribution of this paper is the following:

1. Demonstration of a novel skill-level programming approach, using a combination of a graphical user interface and gesture recognition.

2. A thorough evaluation of the intuitiveness and robustness of the approach using several volunteers. Programming of typical industrial logistic use cases can be done in as little as 2 minutes by individuals that have only been provided with a 10 minute introduction to the system.

1.2. Related work

The idea of robot skills is not a new one [2–4], but the use of robot skills for intuitive programming is not gaining much attention from the research community. When it comes to programming of industrial robots in general, focus seems to be on offline programming by using CAD models and augmented reality. Another focus is task planning, often using motion primitives as the planning domain.

Most recent work on motion primitives seem to utilize a variation of the Dynamic Movement Primitives (DMPs) described in [5–7]. The DMPs specify a single movement, and represents the desired kinematic state of a robot limb (or alternatively in the task space) as a mixture of non-linear differential equations. The DMPs have mainly been used for mapping human to robot motion in research in Teaching by Demonstration.

Another detailed work regarding motion primitives and their use is the Manipulation Primitives (MPs) described in [8, 9]. This work expands on the Task Frame Formalism [10], by enabling the use of any sensor and controller in the same task frame (e.g. visual servoing combined with force control), as well as operating in a dynamic task frame. These MPs have also been combined in Manipulation Primitive Nets [11], with simple sensor-level decisions based on the outcome of each single manipulation primitive in the net.

A layered structure of skills and tasks similar to ours is also proposed in [12]. However, the focus is still planning on the task level, but by utilizing somewhat general skills and skill primitives. The implementation of said skills is however at its current state rather simplistic.

The notion of skills described in this paper mostly resemble the Action Recipes of RoboEarth [13, 14]. However, in RoboEarth, the skills (or Action Recipes) keep sensing and action separate. With its focus on knowledge sharing between robots, the Recipes (robot programs) in RoboEarth rather require some capabilities of the robot, such as object recognition and environment models, and the Recipes are only action based.

Common for all the previous work, however, is that they have all, in one way or another, been insufficient to deploy in a real industrial setting, and require expert knowledge of the robot system to use and implement. In this paper we will show that our method is easy to use by the factory worker, and follows a rather straightforward implementation method.

We will give a brief introduction to our notion of skills in 2. We will then go on to describe the human tracking and gestures in 3 and the GUI in 4. We will finally describe the experiments and results in 5 and conclude the paper in 6.

2. Skills and architecture

We see robot skills as fundamental software building blocks, used for robot task programming, that incorporate sensing and action [1, 15]. The core idea of the skills is to unify sensing and action in one unit. This means the shop floor workers programming the task will only have to relate to and use skills such as `Drive to <Location>` or `Pick up <Object>`, and not robot motion or sensing operations. This unification makes the skills themselves more complex internally, as they have to marginalize out the parameters which are not provided by the user. We will explain the aspects of this in the following. Our model of a robot skill is shown in Fig. C.2.

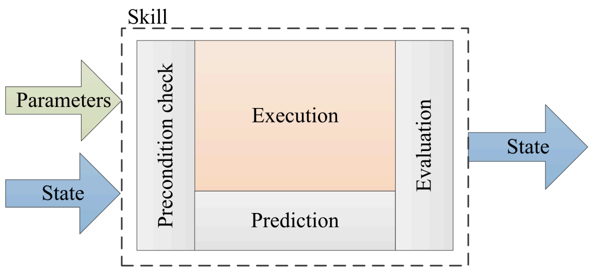


Fig. C.2.: Model of a robot skill

In order to find the right set of skills for solving most industrial tasks, we have analyzed the current tasks of human workers. At Grundfos, the workers are given written instructions on how to perform various tasks, accompanied by images of the relevant workstations or parts. We have previously analyzed more than 500 of these instructions, which revealed only 13 distinct reoccurring general commands such as “insert part A into fixture B", which transfers to skills [15].

As skills are centered around objects, only a certain object as a *parameter* is needed. Everything else is handled within the skill. For instance, the `Pick up <object>` skill is shown which object to pick up by the user pointing at the object. The object is detected using computer vision. Upon skill *execution*, the necessary calculations are made in order to move the arm and gripper so that the object is lifted from the surface and is in the gripper when the skill is finished.

There are three different types of parameters determining how the skill is executed. If we return to the `Pick up <object>` skill, the single *external* parameter, specified by the worker, is which object to pick up. The other two types of parameters are internal to the skill, and the general end user should not be bothered with these, as the robot could compute them on its own. One is related to slight variations in the skill execution, e.g. selecting an overhead grasp instead of a horizontal grasp. The second type of internal parameter is hardware-specific and specified by the skill programmer, and includes robot speeds, contact forces, etc.

In order to facilitate programming and robustness of tasks, the skills contain a notion of their effect, as well as an estimation of whether or not the skill can be executed. When a skill is started, it is first verified based on a set of *preconditions* that the skill actually *can* be executed. The skills include a *prediction* layer, that estimates the outcome of the skill based on the supplied parameters and assumed correct execution. After skill execution, it is verified based on a set of *postconditions* that the skill executed as predicted (this is labeled *Evaluation* in Fig. C.2).

Since the skills need to access prior knowledge of objects and locations in the factory, we provide a simple world model containing only previously detected objects and locations, in a database-like structure.

2.1. System architecture

For this experiment we have implemented methods for *a)* specifying a sequence of skills to form a task, *b)* specifying parameters for each skill through gesture recognition, and *c)* executing the saved task. Since our experimental platform is running ROS, in most cases the communication between programs is done through either topics, services or actions provided in ROS.

The skills are implemented as classes, and each skill inherits from a parent skill class, in order to conform to the same layout. The parent class contains placeholder methods for parameter-based execution, establishing communication with the robot, as well as evaluating all pre and postconditions in the skill. The methods for execution and communication are overridden when a skill is implemented.

One central program handles communication with the graphical interface and task specification. Essentially it receives a list of unparameterized skills from the GUI, which it can then supply to a separate teaching program in order to add parameters to the sequence. The teaching program in turn communicates with the gesture recognition software, in which the user input is determined. A more detailed description of the GUI will be provided in 4, and more details on the teaching program will follow in 5.2.

The world model can be accessed from all of the above components, but it is only the skills that can change the state of the world. This conforms to the notion that it is the skills themselves that afford a change in the world state.

3. Tracker and gesture recognition

This section will introduce the reader to the tracking of human users, the implementation of the gesture recognition and the considered gestures for programming the robot.

3.1. Implementation of tracker

As the fundamental engine for 3D body tracking we have used NiTE 2.2 coupled with the OpenNI 2.2 SDK for accessing 3D sensor hardware. In our experiments we use an Asus Xtion PRO Live RGB-D sensor. We have added a visualization, see Fig. C.3, which shows the tracked person and the status of the gesture recognition process.

The NiTE tracker is capable of tracking multiple persons. However, we are only interested in the gestures of one person, namely the instructor currently teaching the task to the robot. Therefore, we use and recognize a certain *attention* gesture which identifies the instructor, so any tracked person becomes the instructor by performing this gesture. The green color, see Fig. C.3, indicates the instructor; ignored users are marked in blue. Only the location of the instructor and his/her gestures are published to other programs for further processing on a ROS topic.

3.2. Gesture recognition

We consider the recognition of the following gestures: *attention*, *stop*, *abort*, *follow*, and *pointing*. The meaning of each gesture when teaching is described in 5.2. The recognition is based on thresholding different features of the instructor's body pose. The pointing gesture differs slightly, since the instructor

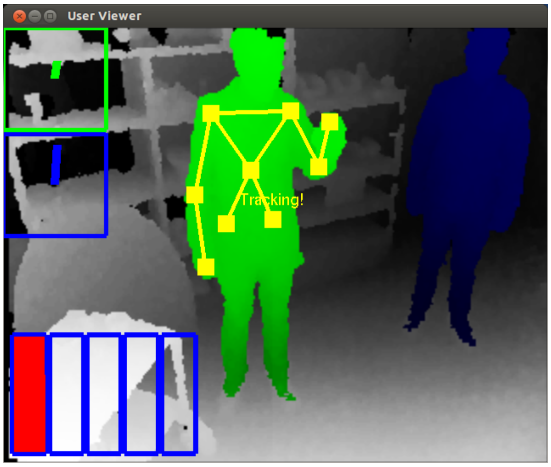


Fig. C.3.: Tracker GUI, with the *attention* gesture being performed by the instructor (in green). The untracked user is marked in blue. The bars in the bottom left of the image show gesture recognition progress.

is supposed to keep the pointing steady. In addition, it is not a simple event and includes a pointing direction encoded as a ROS 3D pose. The considered gestures are shown in Fig. C.4.

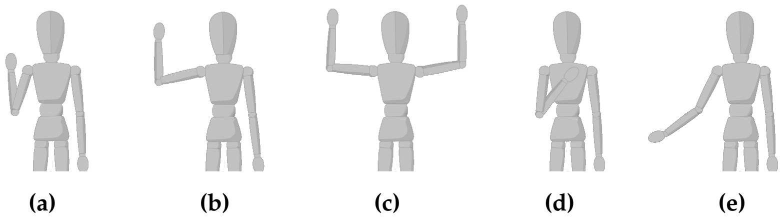


Fig. C.4.: The gestures available during teaching: (a) *attention*, (b) *stop*, (c) *abort*, (d) *follow* and (e) *pointing*.

Let us first consider the thresholding. From the tracker we receive from each joint of the instructor an orientation, its location, and a confidence value between 0.0 and 1.0. In general, we assume that the joints which take part in the thresholding process for one gesture must have a confidence of at least 0.5, otherwise we consider a gesture as invalid. We name the locations provided by the tracker of the right shoulder p_1^r, elbow p_2^r, and hand p_3^r. p_1^l, p_2^l, and p_3^l are the corresponding locations of the left arm. In most cases, the gestures are only valid for the right arm. For the instructor we calculate a normalized vertical upwards vector u and a forwards vector v from the orientation of the torso, i.e. directions in the frame of the instructor. In addition we calculate the location between the shoulders, $s = \frac{1}{2}(p_1^r + p_1^l)$, and a location $s' = s + 0.12 \cdot v$, which is an estimation of the position of the sternum.

Now, we consider the thresholds of the available gestures. In all of the thresholds we have determined the limits empirically (stated in meters in the following), based on experiments with a small number of people.

- **Attention:** Right hand is held up, slightly above the shoulders:

$$0.00 \leq (p_3^r - s) \cdot u < 0.15$$

- **Stop:** Right hand is held up high above the shoulders:

$$(p_3^r - s) \cdot u \geq 0.15$$

- **Abort:** Both hands are held up above the shoulders (this gesture cancels both gestures above):

$$(p_3^r - s) \cdot u \geq 0.00 \ \wedge \ (p_3^l - s) \cdot u \geq 0.00$$

- **Follow:** Right hand close to the sternum s':

$$|p_3^r - s'| \leq 0.10$$

- **Pointing:** Right arm is not simply pointing straight down, i.e. there is some angle between the downwards vector $-u$ and the direction $d = {}^{(p_3^r - p_2^r)}/_{|p_3^r - p_2^r|}$ of the right forearm. The pointing should also be steady (gesture only valid when right elbow and hand do not move more than $10 {}^{cm}/_{s}$):

$$\begin{aligned} & \cos^{-1}(-u \cdot d) \geq 35^\circ \\ \wedge \quad & (p_3^r - p_{3,t-1}^r) \leq 0.10\Delta_t \\ \wedge \quad & (p_2^r - p_{2,t-1}^r) \leq 0.10\Delta_t \end{aligned}$$

  where $p_{3,t-1}^r$ and $p_{2,t-1}^r$ are the corresponding hand and elbow positions in the previous frame, and Δ_t is the duration passed between two frames. Any of the previous gestures cancels the pointing gesture.

We do not consider a gesture recognized until it is held for at least one second. Therefore, we use a counter $c_i \in [0, 1]$ for each gesture which is incremented by Δ_t when the gesture i is valid. If a gesture is invalid, the counter is decremented by $2\Delta_t$. Once a counter c_i becomes 1, the gesture is recognized, all counters are reset to 0, and we ignore further gestures for some time (1.5 s). The counters c_i for each gesture are represented by the bars in Fig. C.3, bottom left.

For the pointing gesture, we publish not only its occurrence but also the right hand location p_3^r and the pointing direction d (the orientation of the forearm) as a 3D pose: The hand location defines the origin and the coordinates for the pointing vector are $(d, v', d \times v')$, where $v' = v - (d \cdot v)d$ is a vector orthogonal to d. The location of the instructor's torso and the coordinate frame $(u, v, u \times v)$ are also published every frame, when the instructor can be tracked. This location is used for following the instructor.

4. User interface

The main interaction with the robot is done through a GUI on an iPad, see Fig. C.5. The GUI is implemented in Python using PyQt, and features a tab-based layout where the two most important tabs are for *sequencing* and *teaching*.

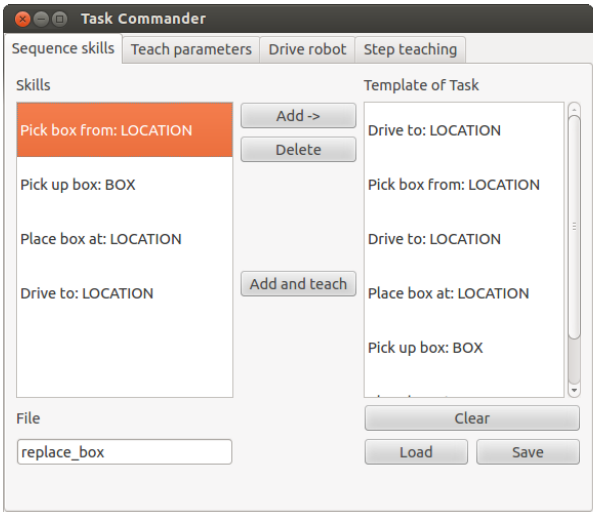


Fig. C.5.: The user interface for task teaching. In the image the sequencing tab is shown.

In the sequencing tab, the user constructs a task sequence by adding any of the available skills on the robot to the current task: The user can choose to drag and drop the skills (left side of the GUI in Fig. C.5) to the current sequence (right side in the GUI), and also rearrange the skills in the task if necessary. When the user is done with the *sequencing* step, the sequence is given a unique filename and saved.

After the sequencing step, the user needs to specify the parameters for each skill in the sequence. This is done in the teaching tab. When the teaching is started, the user is provided feedback on the current state of the teaching operation on the display, e.g. what skill is currently being parameterized, and what gestures the system expects the user to perform. Upon teaching of all parameters for the task, the task is saved with parameters, and can be executed at any time.

Beside the GUI on the touchpad, the user can get visual feedback on the tracking and the world state on a monitor fixed on the robot. For visualization of the world state we use Rviz, with custom markers that show the locations and objects presently known in the world model. For visualization of the tracker status we use the visualization shown in Fig. C.3. This visual feedback is not meant to be necessary for the end user when using the system, but proves useful in getting acquainted with the various gestures and checking for errors in the world state.

5. Experimental results

We begin this section by introducing the reader to the robot system used for experiments, the Little Helper robot shown in Fig. C.1. We then go on to a detailed description of the scenario, and how a user programs the example task of replacing kanban boxes. We finally show experimental results of various people programming the robot in different environments.

5.1. System introduction

Unlike service robots for domestic applications, which are developed at many universities, the focus of the Little Helper is industrial applications, and for addressing the needs for flexible automation. The Little Helper is built from standard industrial components, and consists mainly of three hardware components. For manipulation we use a 7 DOF KUKA Light-Weight Robot (LWR) arm. The robot arm is mounted on a chassis which also contains the robot controller and a computer controlling the entire system. The chassis is mounted on a Neobotix MP-655 differential drive mobile platform. The entire system is running ROS.

For object detection we have used a Microsoft Kinect RGB-D camera. Using QR codes for marking e.g. locations, objects and tools is common in industrial environments. To abstract away object recognition, which is not the focus in this paper, we adopted this methodology in our experiments as well. For this we have implemented a QR code detector, that reads the data contained in the QR code, which specifies which object or location it represents. Furthermore, it calculates the 3D pose of its QR code based on 3D coordinates of the four corners (joined from the RGB image and the depth point cloud). This allows us to easily work with new objects and locations, as it is a matter of printing a novel QR code, and fixing it to the object or location.

For these experiments we have implemented the world model as a database-like program, only containing locations of detected QR codes. This enables us to uniquely identify a specific box or location, as well as an easy method for adding new detections, removing old entries or updating information of a specific entry, e.g. when moving a box. Presently, the world model is limited to objects and locations relevant for the task at hand. Referring to 1.1, this means that the world state is populated with locations of the workstation and the warehouse, as well as locations of all previously detected boxes. Objects and locations are saved as 3D poses in global coordinates.

5.2. Experiment introduction

In order to program this task, the user can construct the sequence from the following skills:

`Pick up <object>`	Aims to pick up a specific object which is visible to the robot, so that the object is located in the gripper and lifted from the supporting surface.
`Pick from <location>`	Picks up an object located at a specific location. The skill first determines which object, if any, is at the location, and then picks up the object like the previous skill.
`Place at <location>`	Puts down an object, currently in the gripper, at a specific location, and moves the gripper away from the object.
`Drive to <object/location>`	Uses the navigation software to drive the robot to a location in front of a previously detected object or location.

Since this paper deals with the usefulness of skills for HRI purposes, we will not provide a detailed description of the implementation of the skills in this paper, but instead point the reader towards the previous work on skill integration [1]. One important point, though, is that this method not only works for these four skills, but enables task programming of any sequence of general skills, that require simple parameters as input. As previously mentioned, this means that although the skill integration is relatively complex, and requires expert knowledge, the skills themselves are highly useful for task programming.

As previously described, there are two steps in programming the robot; *sequencing* and *teaching*. As the sequencing is already previously described, we will now focus on the teaching. Upon teaching the parameters, the user is given feedback on what to do at any given step on the touchpad. For instance, when teaching the `Pick up <object>` skill, the user is told to either instruct the robot to follow him, or to point at an object the robot should pick up. This information is provided from a finite state machine, which also handles the teaching of parameters. Based on the current state of the teaching phase and the supplied gesture, the system determines what to do. The finite state machine for teaching is shown in Fig. C.6.

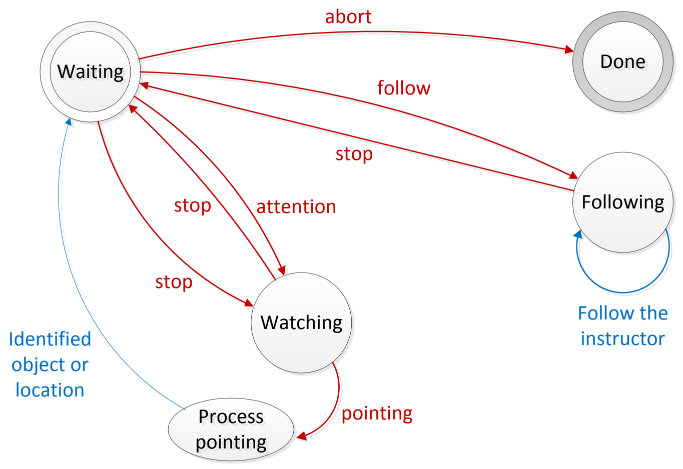


Fig. C.6.: Finite state machine for teaching. Gesture signals are marked in red, and separate processes in blue. Note that the *stop* gesture only transitions from the Waiting to Watching state immediately after a pointing gesture is recognized.

If the user performs the *follow* gesture, he can instruct the robot to follow him to a new location. This functionality has been implemented as a PD controller, which outputs translational and rotational velocity commands to the mobile platform, based on the location of the instructor in the frame of the tracking camera.

The parameter for a skill is computed as soon as the robot recognizes a pointing gesture. When a pointing gesture is recognized, the current scene is scanned for QR codes, and the distance to the pointing vector is calculated for all detections. The distance D between a QR code's 3D position q and the normalized pointing vector d is given by

$$D = |(p_3^r - q) - ((p_3^r - q) \cdot d) \cdot d| \tag{C.1}$$

where p_3^r is the location of the hand (see Sec. 3.2). The exact chosen parameter is given by the QR code closest to the pointing vector. Should the user wish to cancel a chosen parameter, and redo the pointing for one skill, he/she can use the *stop* gesture to cancel the input and redo the pointing.

Upon teaching the last skill in the sequence, the teaching phase is terminated by the *abort* gesture, and the interface returns to the teaching tab, where the programmed parameters for each skill are shown in the task specification. The user can now perform a test run of the skill, either by executing the whole task or one skill at a time.

For the experiments described in the next section, we have asked participants to program tasks such as the replacing of kanban boxes previously described. Here, the user needs to instruct the robot to pick up a box of full parts from a table next to the assembly station. He/she will then tell the robot to place it at the warehouse, pick up an empty box, and place the empty box back at the table next to the workstation. After the task is done, the robot should return automatically to a waiting area. An example of the full skill sequence with parameters is shown below.

1. `Drive to <WorkstationTable2>`
2. `Pick from <WorkstationTable2>`
3. `Drive to <Warehouse3>`
4. `Place on <Shelf1>`
5. `Pick up <EmptyBox4>`
6. `Drive to <WorkstationTable2>`
7. `Place on <WorkstationTable2>`

5.3. Experimental results

We have conducted the experiments with several people in different scenarios. We have initially performed experiments in the running production facility at Grundfos, with only minor modifications to the environment [1, 16], mainly related to the navigation (i.e. adding low barriers around production equipment with overhang, so the robot does not collide with these). For a systematic evaluation, and in order to not disturb the production further, we have moved to other environments, that can be controlled. One is a small, cluttered environment, and the other is a large open environment, with external disturbances in the form of daylight and people passing by. Upon moving to a new scenario, the only requirement has been to attach QR codes to static locations, such as shelves and tables, and to build a map for navigation. After this, the programming can be carried out immediately, as the world model is populated with detected objects and locations during teaching.

We have conducted the tests with a total of 17 different participants. The participants were roughly evenly distributed with respect to age, height, gender, and whether they had any previous experience with robots. The tests were conducted by having the participants program the use case of replacing kanban boxes. Experiments were carried out in three steps: All participant were *1)* given the same 10 minute introduction to the system, *2)* programmed the sequence of skills in the GUI once, and *3)* used the gestures to teach the parameters for the entire task 3 times. The participants are shown in Table C.1. The rightmost column shows the mean teaching time for the 3 attempts. Minimum average time is 134s, maximum is 377s. These experiments show a general picture of how intuitive this approach is.

Participants generally found the sequencing step quite intuitive, and programmed the sequence fast. The times used for sequencing are shown in Fig. C.7a, and also reveal that sequencing the skills was quite intuitive. However, the exact choice of skills to use was quite different. Even though the outcome of the two distinct skills that pick up an object is quite different (`Pick up <object>` vs `Pick from <Location>`), there was no general tendency to use one skill instead of the other. When asked directly to program a different scenario of picking specific boxes from a shelf and placing them at various locations, there was still an even distribution in the choice of skills. From this it is clear that there is some confusion regarding the choice of which skill to use when.

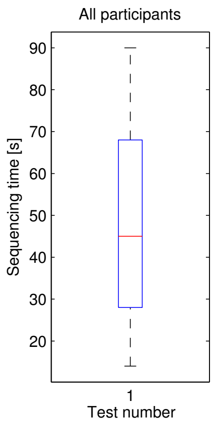
For the teaching experiment, we first investigate the correlation between the mean teaching time for each participant (the time to specify a full set of

ID	Gender	Age	Height $[m]$	Loc.	Exp.	μ_t $[s]$
1	M	28	1.80	A	Yes	143
2	M	26	1.91	A	Yes	377
3	M	33	1.77	A	Yes	252
4	M	30	1.85	A	Yes	170
5	M	24	1.85	B	No	142
6	M	25	1.90	B	No	172
7	M	24	1.80	B	No	194
8	F	41	1.75	B	No	248
9	F	48	1.56	B	No	229
10	F	44	1.60	B	No	145
11	M	26	1.90	B	Yes	228
12	M	25	1.82	B	Yes	134
13	M	31	1.78	B	No	146
14	F	28	1.65	A	No	194
15	M	30	1.90	A	No	337
16	F	27	1.73	A	No	159
17	F	26	1.77	A	No	146

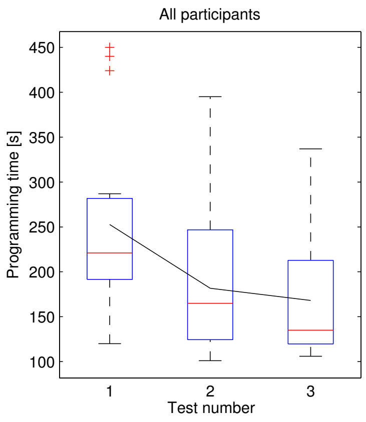
Table C.1.: Participants in the experiments. Loc. specifies the location and Exp. specifies previous experience with robots. μ_t is the mean teaching time of 3 attempts.

parameters for the sequence) and the continuous variables age and height. Mean teaching times with respect to age and height are shown in Fig. C.8. A low correlation of 0.11 between age and teaching time suggests that age is not a factor that needs to be investigated further. However, the correlation between height and teaching time was slightly higher at 0.29. This was also apparent during the experiments; because the tracking camera was mounted pointing slightly downward, the head was not fully visible for some of the taller participants. This resulted in bad or lost tracking of the user, which in turn leads to longer teaching time. This is however merely a hardware issue, as the camera could be adjusted and its position with respect to the robot recalibrated, but this was not done for the experiments. The adjustment would only need to be performed once, as it was apparent from the experiments that an orientation that would be able to track the taller users would still be able to track shorter ones.

In order to investigate the learning curve for using the gesture approach, the teaching times for all participants are shown in Fig. C.7b. We observe a general improvement in teaching time, both with respect to the mean time and the variance. This suggests that the approach is in fact quite intuitive, as



(a)



(b)

Fig. C.7.: Box charts showing sequencing time (a) and teaching times (b) for all participants. The black curve in (b) shows the mean teaching time.

participants generally learned how to use the gestures effectively within the 3 attempts, and even on the first attempt taught the task parameters within minutes.

We have also investigated the learning curve based on gender, location, and previous experience with robots. Gender and experience are closely related to the intuitiveness of the system, where location is related to the robustness of the system in the case of outside disturbances.

Generally, female participants performed better on all attempts, see Fig. C.9a. The female group was generally shorter than the male group, which could have an influence, but otherwise almost evenly distributed. Even though the mean time for the first attempt is similar to the one for male participants, the variance is lower, and for attempts 2 and 3 the mean time is significantly reduced. The female group is in fact the group with the lowest mean and 2^{nd} lowest variance on the 3^{rd} and final attempt.

There is some difference among participants experienced and inexperienced in robotics, shown in Fig. C.9b. The experienced group was all male, but other variables were distributed almost evenly. What is surprising is the high variance of the teaching time for the first attempt for the experienced group, which could be due to these participants having a notion on how the system works. The experienced group improved rapidly, however, resulting in very low mean and variance of teaching time on the third attempt. The final mean time of the inexperienced group was only slightly higher. However, the

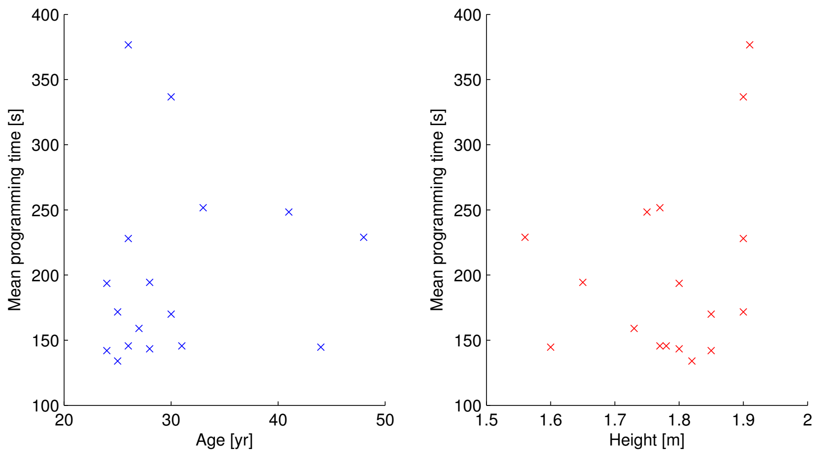


Fig. C.8.: Mean teaching time for participants with respect to participant age (left) and height (right)

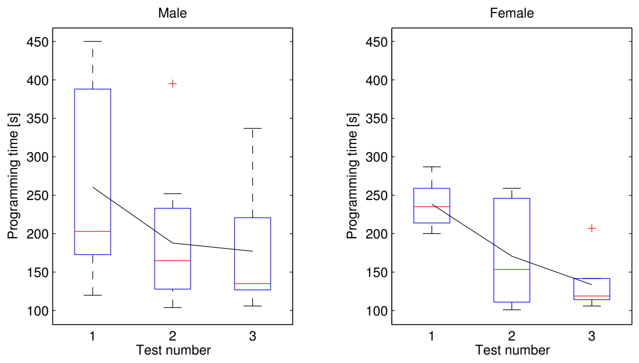
variance was much greater, suggesting the experienced group would perform better and more consistent on subsequent attempts.

When we consider the two different locations used for experiments, it is clear from Fig. C.9c that results were faster and more consistent at the large, open environment (B). We believe that this is due to the tracker not performing well in the small and cluttered environment (A), increasing the teaching time. In some cases background objects were considered part of the instructor, especially when standing close to these objects. This difference is hardware-related, and will most likely not be an issue in a real production facility, which mostly resembles the open environment.

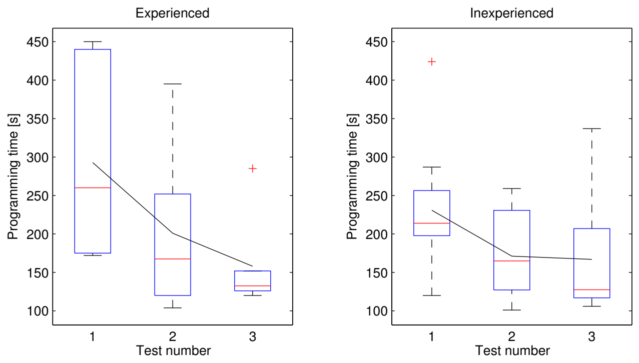
6. Conclusion

In this paper we have demonstrated how a combination of robot skills, a graphical interface and gesture recognition can help to make programming of flexible industrial robots intuitive. We have implemented a set of skills, which are general and parametrizable, so that the factory workers can concatenate these skills to form a task.

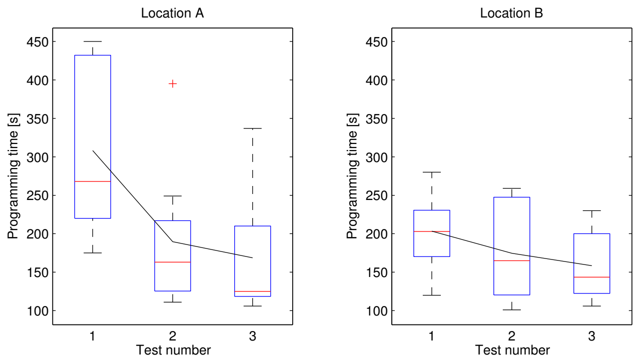
We have tested our approach on several people, systematically evaluated in two different scenarios. The experiments show that the approach described in this paper is in fact intuitive, as all participants programmed the task within minutes, even on the first attempt, and even without any previous experience with robots. Furthermore, both the variance and mean of the teaching time is



(a) Male vs. female participants



(b) Experienced vs. inexperienced participants



(c) Cluttered (A) vs. open (B) environment

Fig. C.9.: Box charts comparing teaching time for each attempt, when considering discrete variables in the experiments. The black curve indicates mean teaching time.

significantly reduced after 3 attempts of programming the robot. There were some issues related to the tracking of the instructor, due to the orientation of the camera and noisy tracking in a small environment.

Compared to traditional robot programming, this approach seems both faster and more general. Even if all objects and locations were known with absolute certainty, programming the task using traditional programming methods would still be significantly slower, and require expert knowledge.

Overall, the approach of skills was intuitive for all users, as the outcome of the skill is easily understandable. General feedback from the participants was also positive, especially with regard to the approach. However, feedback on the performance of the tracker, and when to use certain gestures was not so positive. The gestures were generally easy to use, but some attention will have to be paid to making the use of gestures more intuitive, and especially the tracking more robust.

It should obviously be noted that if the system is deployed in a real industrial setting, the training of factory workers to use the system would be much more elaborate than just a 10 minute introduction. As experiments already show significant improvement for just 3 attempts at programming the robot, given this short introduction, the workers would in a short time be very proficient at programming the robot.

The reader may argue that the programming is so intuitive because the tasks considered here are simple in general and that, for instance, programming constructs like loops are missing (e.g. the task of taking *all* objects from a certain location and putting them on a tray). However, there are many handling tasks in the industry, similar to the tasks discussed in this paper, which are not automized. The reason is exactly that state of the art programming techniques do not allow to program these tasks on the fly. Regarding loops, which is a next step in our work, it should be noted that previously programmed tasks could be executed by the Manufacturing Execution System (MES). In this case, loops, if-else clauses, etc. are handled by the MES.

7. Future work

For these authors, the immediate continuation of this work is to expand the scenario and skills. This includes, but is not limited to, picking and placing multiple objects in a cluttered environment. This will also require a full implementation of intrinsic parameters, as well as determining the exact parameter to use based on later skills in the task. A complete, general skill implementation so to say.

ACKNOWLEDGMENTS

The authors would like to thank Grundfos for the possibility of doing experiments in the factory. We would also like to thank the volunteers for testing the proposed method. This work has partly been supported by the European Commission under grant agreement number FP7-260026-TAPAS.

References

[1] M. R. Pedersen, L. Nalpantidis, A. Bobick, and V. Krüger, "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. [Online]. Available: http://renaud-detry.net/events/crs2013/papers/Pedersen.pdf

[2] V. Krüger, D. Kragic, A. Ude, and C. Geib, "The meaning of action: a review on action recognition and mapping," *Advanced Robotics*, vol. 21, no. 13, pp. 1473–1501, 2007.

[3] N. Krüger, J. Piater, F. Wörgötter, C. Geib, R. Petrick, M. Steedman, A. Ude, T. Asfour, D. Kraft, D. Omrcen *et al.*, "A formal definition of object-action complexes and examples at different levels of the processing hierarchy," Technical report, 2009. [Online]. Available: http://www.paco-plus.org

[4] C. C. Archibald, "A computational model for skills-oriented robot programming," PhD Thesis, University of Ottawa, Ottawa, Canada, 1995.

[5] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Trajectory formation for imitation with nonlinear dynamical systems," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 2, 2001, pp. 752–757.

[6] S. Schaal, "Dynamic movement primitives -a framework for motor control in humans and humanoid robotics," in *Adaptive Motion of Animals and Machines*, H. Kimura, K. Tsuchiya, A. Ishiguro, and H. Witte, Eds. Springer Tokyo, Jan. 2006, pp. 261–280.

[7] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Robotics Research*, ser. Springer Tracts in Advanced

Robotics, P. Dario and R. Chatila, Eds. Springer Berlin Heidelberg, Jan. 2005, no. 15, pp. 561–572.

[8] T. Kröger, B. Finkemeyer, and F. Wahl, “Manipulation primitives — a universal interface between sensor-based motion control and robot programming,” in *Robotic Systems for Handling and Assembly*, ser. Springer Tracts in Advanced Robotics, D. Schütz and F. Wahl, Eds. Springer Berlin / Heidelberg, 2011, vol. 67, pp. 293–313.

[9] T. Kröger, B. Finkemeyer, S. Winkelbach, L.-O. Eble, S. Molkenstruck, and F. Wahl, “A manipulator plays jenga,” *Robotics Automation Magazine, IEEE*, vol. 15, no. 3, pp. 79 –84, Sep. 2008.

[10] H. Bruyninckx and J. De Schutter, “Specification of force-controlled actions in the "task frame formalism"-a synthesis,” *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 581 –589, Aug. 1996.

[11] 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, Jun. 2005.

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

[13] M. Waibel, M. Beetz, J. Civera, R. D’Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. M. M. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, “RoboEarth,” *IEEE Robotics Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.

[14] M. Tenorth, A. Perzylo, R. Lafrenz, and M. Beetz, “Representation and exchange of knowledge about actions, objects, and environments in the RoboEarth framework,” *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 643–651, 2013.

[15] S. Bøgh, O. S. Nielsen, M. R. Pedersen, V. Krüger, and O. Madsen, “Does your robot have skills?” in *Proceedings of the 43rd International Symposium on Robotics (ISR)*, Taipei, Taiwan, Aug. 2012.

[16] O. Madsen, S. Bøgh, C. Schou, R. Andersen, J. Damgaard, M. R. Pedersen, and V. Krüger, “Integration of two autonomous mobile manipulators in a real-world industrial setting,” in *IROS Workshop on Robotic Assistance Technologies in Industrial Settings (RATIS)*, Tokyo, Japan, Nov. 2013.

Paper D

Planning of Industrial Logistic Tasks in the Robot Skill Domain

Mikkel Rath Pedersen, Volker Krüger

The paper has been submitted October 1st 2014 to
2015 IEEE International Conference on Robotics and Automation (ICRA).

The layout has been revised.

Abstract

Automated task planning for robots is usually implemented on a motion primitive domain, where the focus is on constructing meaningful, general motion primitives. In this work we propose planning on the higher abstraction level of robot skills. Skills are general, functional blocks that contain both sensing and action, have a well-defined expected outcome and set of preconditions, and are thus immediately useful for planning. By adding a simple world model, we show that this is in fact the case, by planning and executing the sequence of skills and their parameters based on the desired goal state and the current state from the world model. Experiments show that the approach is immediately applicable, given a skill-equipped robot, and that inconsistencies between the world model and actual world state are overcome simply by replanning.

1. Introduction

Despite a high degree of automation in the industry, a lot of tasks are not feasible to automate using traditional methods. This is mainly because programming industrial robots is expensive, both in terms of manpower and downtime, and the fact that industrial robots traditionally are considered mass-production equipment, that repeatedly performs specific motions in a highly structured environment. With the introduction of mobile manipulators, this is changing. One area that we have found especially suitable for mobile manipulators is logistic tasks, e.g. transportation of parts between workstations or part feeding [1].

Even though mobile manipulators are able to solve these tasks, the problem remains how to program them on the fly. In our previous work, we have used a set of robot *skills*, that are easily understandable by the factory worker, to explicitly program tasks on the robot [2]. Users programmed a task by selecting the desired sequence of skills, and specified the parameters by pointing at objects to manipulate. Generally, even robotics novices felt that programming the robot in this way was intuitive.

However, this approach has severe limitations in terms of the flexibility of the programmed tasks, since the sequence of skills in the task is fixed, and thus e.g. the number of objects to manipulate is likewise fixed. In this work we instead implement planning based on the skills and the world state, that is concurrently updated during skill execution, so the programming a task simply becomes a matter of specifying the desired goal state, e.g. that all objects should be at a certain location.

We consider a STRIPS-like approach [3], where the skills are the software building blocks that make up a task. Our skills already conform to the STRIPS paradigm, in that they have the sensing built in to verify the relevant parameters and conditions, and contain the corresponding code for actions. It is then possible to limit the planning domain for logistic tasks to skills such as *pick up "object"*, *drive to "location"*, etc., as well as monitor the execution of tasks, and perform simple replanning in case of errors.

The contribution of this paper is the following:

- Ad-hoc task planning of logistic tasks on a robot equipped with skills, using the
- current world state extracted from a continuous world model as the initial (discrete) world state for the planning problem.
- Execution and monitoring of the planned task on a real robot, and replanning in the case of errors, ensuring the goal is satisfied.

1.1. Related work

Establishing fitting representations of skills to use for task-level programming is the focus of much research around the world [4–7]. However, skills are usually primitive formal descriptions of compliant robot motions, called primitives, which are defined as simple, atomic movements, that can be combined to form a task [8–10]. The so-called "manipulation primitive nets," described best in [11], are comparable to our notion of skills, in that they are sequences of manipulation primitives, with primitive logics based on the outcome of each single primitive in the net. They are however still quite complex structures, with complex parameters, not easily understandable by robotics novices. In this work we instead propose skills on a higher level, so the skills become a middle layer between the classic motion primitives and robot tasks.

Much research has been done on task planning, the best known being the STRIPS planner [3]. Many other algorithms have followed, based on the same ideas, in that they act on a set of actions that alters the current world state [12, 13]. This includes breaking down assembly tasks into some form of motion primitives the robot can interpret [14]. The problem with modeling the world state, and maintaining this model, persists in these implementations. Furthermore, using primitive motion or sensing actions as the planning domain, it naturally becomes more difficult to create a plan that solves even the relatively simple examples of logistic tasks mentioned above. In this work we limit the world model to model the specific scenario and available set of skills, and plan on the higher-level skill domain.

One approach has been to use logic programming to describe available robot actions and their results, using the situation calculus [15, 16]. In the situation calculus, the world state is only dependent on an initial state and the actions performed by the robot. This approach did however lack the interfacing to the actual robot, something GOLEX [17] overcame, although for a very simple scenario. Recent work suggests that the overall idea of using logic programming could be an option for implementing our notion of skills [18].

2. Skills for planning

In a nutshell, we see robot skills as the high-level building blocks that can be used to construct full robot programs, or tasks. A *task* is a fully instantiated sequence of skills, with specified parameters, and is characterized as solving a specific goal in the environment where the robot is deployed, e.g. *fill the feeder with parts* or *empty the workstation*. The skill sequence and parameters can either be specified manually by an operator, or – like in this work – by planning into a desired goal state.

Our model of a robot *skill* is shown in Fig. D.1. Each skill has one or more intuitive parameters as input, and contains all functions that are necessary, based on the input, to check *before* executing that the skill can be applied; to execute the high-level action; and to check *after* execution, if the skill indeed had the predicted effect on the world. For instance, consider the *place at* skill. This requires the surface to place the object on as a parameter. Before executing the necessary sensing and action operations to place the object, it is verified that an object is in the gripper, there is free space at the surface, the surface is reachable, etc. Likewise, after execution it is verified that the object is present at the predicted location, the gripper is removed from the object, etc.

Each skill has associated a set of preconditions and a prediction of the outcome of its execution in the real world, which are used for the checking procedures before and after execution. These can either be verified from the world model or through sensing operations, depending on the condition. For instance, to determine if a surface to place an object on is within reach might be quickly assessed from the world model, but whether or not there is actually free space on the surface needs to be verified through sensing.

During the execution of the skill, the robot performs the pre-programmed sensing and action operations to accomplish the skill, based on the input parameter. These are specified by the skill programmer, and should ideally be implemented in such a way that the skill can handle all parameters that are relevant in the factory. The operator is never exposed to the individual op-

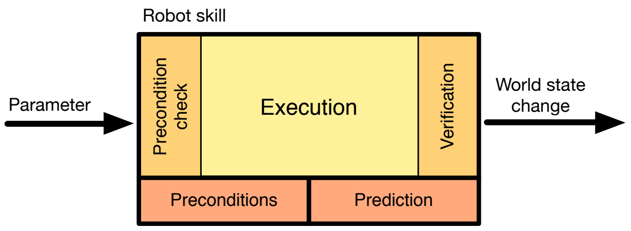


Fig. D.1.: Conceptual model of a robot skill. The preconditions and prediction (red blocks) are informative aspects of the skill. The functional blocks (in orange and yellow) asserts that the skill can be executed, executes the sensing and action operations, and verifies correct execution, all based on the informative blocks and the supplied parameter. The world state is updated throughout the functional blocks, depending on the outcome of sensing or action operations.

erations within the skill, but only has to relate to the complete skill and its intuitive parameters.

It should have become apparent by now that these skills are almost immediately useful for planning, since they only require high-level parameters, and contain a description of their preconditions and expected outcome. Since this information is already present, the skills can immediately be formulated in a planning language like PDDL (Planning Domain Definition Language) [19]. This formulation could in principle also be automated. What is lacking, both for planning and pre- and postconditions checks, is a world model.

Defining a general world model is an open problem, and we will not attempt to solve it in this paper. The world model should contain only the skill-relevant information about the environment where the robot is deployed. By this we not only mean ad-hoc modeled information about e.g. previously detected objects or surfaces the skills can act on, although this is a major part of the necessary world model, but also implicit information, e.g. the factory map used for navigation or the current internal state of the robot. Since the robot is operating in a human-populated environment, the world model is prone to disturbances. When this happens, a manually specified task fails, and the only fallback would be to call an operator to fix the problem. With the planning approach, however, the system can create a new plan based on the updated world state.

Given a skill-centric world model, maintained by the skills themselves, it is possible to create a plan based on the current state of the world. This implies that it should be possible to decompose the world model into a language like PDDL, that requires the initial state of the world specified as a set of binary state variables, or *predicates*. Since the *current* state is extracted from the world model, and always used as the initial state for the planner, it is only the specification of a desired goal state that is needed to plan a complete robot task.

In a deployed robot system, it should be intuitive for an operator to specify the desired goal state, e.g. through a GUI, voice commands or gesture recognition. This interaction is however not part of this work, as we are investigating how skills can be used for planning.

3. Implementation and considered scenarios

The example logistic task that forms the basis for this work deals with the transportation of small kanban boxes in a factory, and multiple part feeding. The scenario is illustrated in Fig. D.2. In this scenario, parts for an assembly line are manufactured and placed in kanban boxes at the workstation. The number of full boxes at the workstation is dynamic, and the robot has no knowledge of when boxes appear at the workstation. The boxes should periodically be emptied into a feeder, that provides parts directly to an assembly line. Furthermore, there are two shelves where boxes can be placed – one for storage of parts in full boxes and one for empty boxes, that can be removed at any point in time. The robot can grasp one box at a time, and carry 2 additional boxes in a rack mounted on the robot platform.

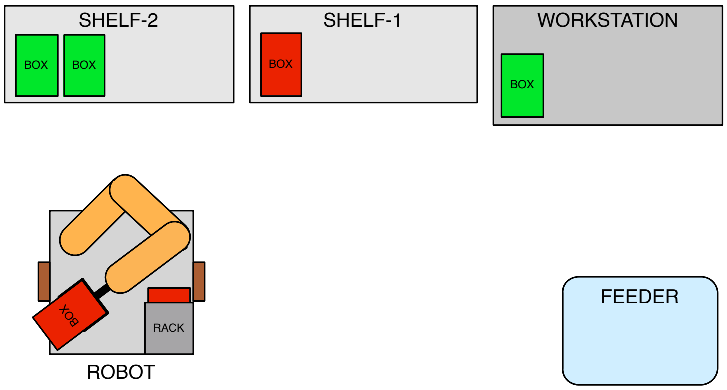


Fig. D.2.: Example logistic task of sorting boxes and tending to a feeder for a production line. Parts are produced at the workstation and placed in boxes. Full boxes are green and empty boxes are red. The robot can carry 1 box in the gripper, and 2 in a rack mounted on the robot.

We restrict the skills to manipulate boxes in order to abstract away object recognition and grasp detection, which is a key requirement for truly general skills, but not the issue in this work. All boxes and locations are marked by

QR codes, from which we can easily detect the pose and the box or location information, encoded in the QR code. Furthermore, the robot is equipped with a passive gripper, that only gives one available grasp for the boxes. Since the QR codes marks the skill-relevant objects and locations, the world model for this scenario consists of previously detected QR codes. This will be elaborated in 3.2.

To manipulate the boxes, we have implemented 4 skills on the robot, with the intuitive effects; `pick up <box>`, `place on <location>`, `drive to <location>` and `unload in <feeder>`. We will not describe the implementation of the skills in this paper, but instead refer to our previous work, as the skill used for this work are similar [2]. It should be noted that the pick and place skills are able to utilize the rack on the robot as well, depending on the supplied parameter.

Although the skills are only able to manipulate boxes, an important aspect of the skills is that they only require abstract parameters, and not e.g. contact forces or specific 3D coordinates. As such, the approach presented in this work is also valid for much more complicated skills, even skills that require several parameters, and still using only an ad-hoc world model. We summarize the preconditions and prediction of each skill in the following, since these are relevant for planning.

3.1. PDDL domain definition

Now we will describe the PDDL encoding of the planning *domain*. A PDDL planning domain is a description of the *predicates*, or state variables, that are valid for the specific domain, and the available *actions* that change these predicates. Predicates always evaluate to true or false, but in this work we will also make use of the extension to *numeric fluents*, which can assume numeric values. We also make use of *typing*, which enables parameters of certain types, so we can distinguish between boxes, locations and the feeder. Actions are formulated in the STRIPS notation by the set of input parameters, and their preconditions and effects expressed by predicates. We will later describe how we construct the PDDL *problem* from the current world state. A PDDL problem is the specific scenario to plan a solution for, using the actions available in the domain, and is expressed by *a)* the set of objects in the scenario and their types, *b)* the initial state, and *c)* a desired goal setting, the latter two expressed by predicates or numeric fluents.

Since the skills have high-level parameters as input, and contain preconditions and prediction of the outcome, they map easily to PDDL actions. We use a total of 9 predicates for describing the states and actions, most of which have the intuitive meaning. For reference, the complete PDDL domain description is shown in the appendix.

Box and robot locations are specified by the predicates `box-at` and `rob-at`. For boxes, this is the location where the box is currently placed on, and for the robot it marks the location in front of which it is currently parked, so that all boxes at the location are reachable. In the initial state of the PDDL problem, we always set the `rob-at` predicate for the robot rack, since the boxes in the rack are always reachable, and we specify in the `DRIVE_TO` action that the robot can not drive *from* the robot rack. This ensures that, from a planning point of view, the robot can always reach the boxes in the rack.

For the feeder and boxes we use the predicate `empty`. The gripper state is specified by the two predicates `holding` and `empty-handed`, where the first specifies which box is currently in the gripper, and the second states if the gripper is empty. The use of the `empty-handed` predicate is to avoid negated predicates, specifically as a precondition for the `PICK_UP` action, since the robot should not be carrying a box when executing this skill.

For specifying the desired locations of full or empty boxes, we introduce the predicates `full-loc` and `empty-loc`, which are set when specifying the planning problem. These could be set as constants in the domain file, since the scenario is limited, but by setting them as predicates we can change these locations at programming time, while keeping the domain file static. For determining if the boxes are at the right locations, depending on the previous predicates, we introduce the predicates `at-empty-loc` and `at-full-loc`. Initially, `at-empty-loc` gets set for all non-empty boxes, since we are only interested in the *empty* boxes. Similarly, the `at-full-loc` gets set for all empty boxes. When placing a box, we set either of these two predicates, depending on the `full-loc` and `empty-loc` predicates for the place location. Consider the example of placing `box-1` at location `shelf-2`. If e.g. `(full-loc shelf-2)` is satisfied, then `(at-full-loc box-1)` is set as the effect. Furthermore, when a box is unloaded in the feeder, the `empty` predicate is set, and additionally the `at-empty-loc` predicate for that box is negated, since the box was previously non-empty (a precondition of the `unload` action).

Besides the predicates, we introduce three numeric fluents, where two are related to the capacity of placing locations, and one is related to cost minimization. Since we know the width of both the boxes and the valid place locations (which is encoded in the QR codes), we can estimate the total number of boxes that can be placed at a certain location. This is the `loadlimit` fluent, which is set in the initial state in the problem definition. Furthermore, we set the number of boxes currently at a certain location using the `currentload` fluent. The `pick` action decreases this value by 1 for the location where the box was picked from, and the `place` action increments it. It is a precondition of the `place` action that there is a free spot on the place location, i.e. that `currentload` < `loadlimit`.

The third fluent, related to cost minimization, is the relatively standard `total-cost` fluent. In the effect of each action (see the appendix) the cost is increased. We initially related the cost to the actual execution time of each skill. This did however result in similar costs for the manipulation skills and the driving skill, though with a much higher variance on the driving skill. Instead, we now specify the costs manually. Since the robot is working in a human-populated environment, it is desirable to limit the use of the driving skill, and we thus introduce a cost of 10 on the `DRIVE_TO` action. Furthermore, the robot should utilize the rack when relevant, by carrying several boxes when driving. For this reason we introduce a cost of 1 on the `PICK_UP` and `PLACE_ON` actions, when the box is picked from or placed on the rack. For other locations, as well as for the `UNLOAD` action, the cost is 2. This results in a combined lower cost of picking a box from e.g. the workstation and placing it in the rack, than picking the box and placing it at e.g. a shelf.

3.2. World model

As previously mentioned, we have implemented an ad-hoc world model for this scenario. Since locations and boxes are marked by QR codes, the world model is a database of previously detected QR codes. Each entry contains the type of object and a unique value for that object, where the type is e.g. BOX or SHELF. Both the type and value are contained in the string encoded in the QR code. Additionally, we encode the width of locations, again to abstract away surface detection, which is not the focus of this work. Besides these parameters, which are extracted directly from the QR code, we calculate the 3D pose of each detection from an RGB-D image, and save it in a fixed frame. Finally, the two boolean parameters *empty* and *in-gripper* are set for each detection, both with the intuitive meaning, where both apply to boxes, and the *empty* parameter also applies to feeders.

For extracting the current state from the world model as a PDDL problem, we have implemented a parser for the world model. This outputs the contents of the PDDL problem containers `:objects`, which is a list of all objects, and their types, in the planning scene, and `:init`, which is the initial state expressed by predicates. The final part of a PDDL problem, the `:goal` container, is specified manually. The `:objects` definition is trivial to extract, since we only have to map each entry in the world model to the PDDL types *box*, *feeder* or *location*, specified in the domain file. Most of the `:init` container, which will contain the current state from the world model, is similarly trivial to encode into predicates, as the information is directly accessible from the world model – i.e. the predicates `empty-handed`, `holding` and `empty`. The `rob-at` predicate is determined based on the transformation between the current robot position and each location that can hold boxes. If this transformation, for any one location, is equal to the destination coordinate (with some tolerance) for the `DRIVE_TO` skill (that is a specific pose relative to the location), the predicate is set for this location. If no location qualifies, the predicate is set for the constant `unknown-loc` (which is not a valid location to place boxes). As previously mentioned, the function `loadlimit` can be determined from the width of the locations, and the (fixed) width of the boxes. Finally, `box-at` and `currentload` are based on the box locations. We determine at which discrete location each box is located, based on the location poses and widths, and the box poses.

3.3. Planner and plan monitoring

Since this work does not deal with planning algorithms, but instead focuses on creating and executing plans expressed as a sequence of skills, we will use a readily available planner. As our PDDL domain specification uses numeric fluents, this restricts us to use a planner that can deal with PDDL version 2.1 level 2 [19], and can do cost minimization. For this reason we use the `metric-ff` planner [20]. After we construct the PDDL problem file from the world model and the manual goal specification, we call the planner as a subprocess, and parse the output, which contains the plan as a sequence of skills with their parameters - including the single skill-relevant parameter.

The plan monitoring is depicted in Fig. D.3. In the case of skill errors, be it unsatisfied pre- or postconditions or execution errors, replanning is performed. Similarly, when the plan is completed, the planner is called again, to verify that the goal state has been satisfied.

4. Experimental evaluation

We evaluate our approach by two different goal specifications – filling the feeder and sorting full and empty boxes on the correct shelves. Furthermore,

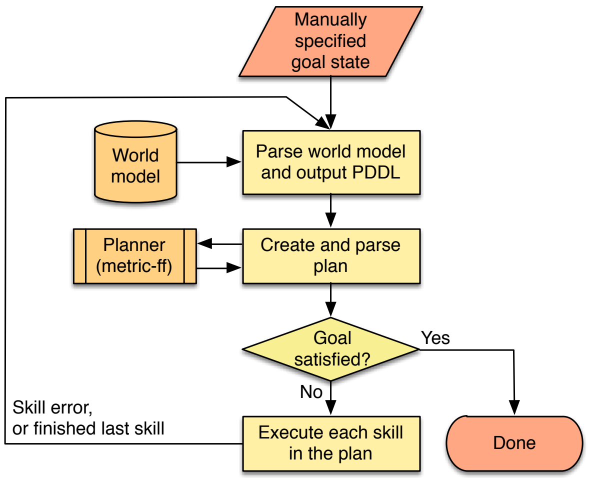

Manually specified goal state
World model
Parse world model and output PDDL
Planner (metric-ff)
Create and parse plan
Goal satisfied?
Yes
No
Skill error, or finished last skill
Execute each skill in the plan
Done

Fig. D.3.: Approach to planning and execution of tasks. The desired goal state is the only input, and replanning is performed in the case of skill errors.

we introduce disturbances to both scenarios during execution. The plans are executed on a real robot, in an environment like previously described, with 4 boxes present. The environment is shown in Fig. D.4.

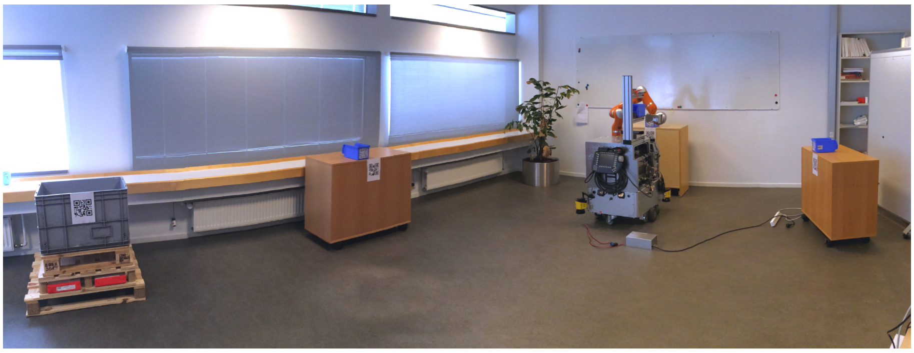


Fig. D.4.: Environment used for experiments, with the feeder on the left, the robot in front of the workstation, the two shelves and four boxes.

Unless otherwise specified, we use the following as the initial state, where some predicates are intentionally left out for readability:

```
(:init ...
    (box-at BOX-1 SHELF-2)
    (box-at BOX-2 WORKSTATION)
    (box-at BOX-3 WORKSTATION)
    (box-at BOX-4 SHELF-1)
    (empty BOX-1)
    (empty FEEDER)
    (full-loc SHELF-2)
    (empty-loc SHELF-1)
... )
```

The reader should notice that the planner is operating under the closed world assumption, where all non-specified predicates are by default negated, i.e. evaluate to false – in the above, this implies that e.g. boxes 2, 3 and 4 are non-empty.

A common goal for all scenarios, is the condition that the robot should not be carrying any boxes, neither in the gripper (using the `empty-handed` predicate) nor the rack (setting the `currentload` function to 0 for the rack).

4.1. Undisturbed experiments

As a first experiment, we consider the scenario where the robot has to add parts to the feeder. We simply specify the goal as the negated predicate `(not (empty FEEDER))`, as well as the two previously mentioned conditions that the robot should not carry any boxes. Since no condition is set for where empty or non-empty boxes should be placed, the robot drives to and picks up any non-empty box, drives to the feeder, unloads the box, and finally drives to and places the box at any free location. Parsing the world model and planning takes a little less than $0.1s$, and the plan consists of 6 skills. The total plan execution time is $3.5min$.

The second experiment concerns sorting the boxes, so the empty boxes arrive at SHELF-1 and non-empty boxes arrive at SHELF-2, as specified in the initial state. In order to do this, we use the previously mentioned `at-full-loc` and `at-empty-loc` predicates. The parser for the world model sets the following in the initial state, based on the box `empty` state, and their locations, since none of the boxes are at the right locations (also see 3.1 for clarification on the use of these predicates):

```
(:init ...
    (at-full-loc BOX-1)
    (at-empty-loc BOX-2)
    (at-empty-loc BOX-3)
    (at-empty-loc BOX-4)
... )
```

Since we set the desired location of empty and non-empty boxes in the initial state, and set e.g. the `at-full-loc` predicate for all empty boxes, the goal specification simply becomes:

```
(:goal ...
    (forall (?b - box) (at-empty-loc ?b))
    (forall (?b - box) (at-full-loc ?b))
... )
```

We observe that the found plan makes use of the rack as expected, since the robot drives to the workstation and picks up both boxes (2 and 3). It then continues to SHELF-2 (where full boxes should be placed), places both boxes, and finally swap boxes 1 and 4 between the two shelves. Again, the parsing and planning takes less than $0.1s$, although slightly shorter than for the feeder filling scenario, most likely since there are no negated predicates in the goal specification for this scenario. Since there are now 14 steps involved in the found plan, the execution time is $7.5min$.

4.2. Experiments with introduced disturbances

We now introduce disturbances to both scenarios. For the feeder-filling task, the goal and initial state is the same as before. When the robot has picked up a box, and is driving towards the feeder, we physically remove the box from the gripper. Since a precondition for the `unload` action is that a box is in the gripper, this skill fails, as expected. The robot then replans based on the current state, and finds and executes a similar plan as before, but with a different box. The execution time is then $4.3min$, after executing a total of 9 skills.

For the sorting of boxes, we introduce the three different disturbances that can occur in a human-populated environment: *a)* a box in the world model is not present in the real world, *b)* a box is removed from the rack during task execution, and *c)* a box is at a different location than expected. For all these experiments, we have removed BOX-4 entirely, since we can test the disturbances with the remaining 3 boxes. For all 3 cases, the robot initially plans the same task, which is driving to the workstation, picking up boxes 2 and 3, and driving to and placing the boxes at SHELF-1.

In the case of disturbance a, we physically remove BOX-2 from the workstation. When the `PICK_UP BOX-2` skill is called, the precondition that the box is present fails, and the world model is updated accordingly. The robot then replans, and either picks up BOX-3 or drives directly to SHELF-1, if BOX-3 was already picked up. This single box is then placed on the shelf.

Since the found plan includes placing either BOX-2 or BOX-3 in the rack, for testing disturbance *b* we physically remove the box that the robot has placed in the rack. After driving to SHELF-1, the robot places the box in the gripper, and calls the `PICK_UP` skill for the box in the rack. Since the precondition for this skill fails, as the box is not present, the box is cleared from the world model, and the robot replans. However, in this case the goal is already satisfied, and no further action is required.

For the final disturbance, *c*, we completely remove BOX-2 from the scene, so the initial plan is simply to move BOX-3 to SHELF-1. The disturbance is that BOX-4 is moved to the workstation, although it is still at SHELF-1 according to the world model. In this case, the robot only executes the initial plan. However, since the poses of entries in the world model are updated whenever the corresponding sensing is performed, the final check determines that the goal is not satisfied after the initial plan is executed (since BOX-4 is non-empty, and not at SHELF-1). Therefore, the robot executes the plan found in the final check, which moves BOX-4 to the shelf.

As a final experiment, we demonstrate that our approach scales well. Without executing in a real environment, we create a PDDL problem, containing 100 boxes, 15 shelves, 3 workstations and 3 feeders, where all predicates are randomized. The goal is the combination of filling the 3 feeders and sorting the boxes. Planning is still successful, but now takes between 35 and 40s on a standard laptop. Considering that this is about the same time it takes for the robot to navigate between two locations, the planning is still relatively cheap from a temporal point of view – and having the operator manually specify tasks of this size would be next to impossible.

5. Discussion

Concerning replanning in the case of skill failures, this is not always the best approach. Specifically, simple replanning in the case of postcondition failures would not always yield good results, as this suggests the skill does not work as intended. With replanning, like we do in this work, the new plan will include the skill that previously failed. This would most likely be a good solution in most cases, but with the possibility that the robot will continue to try executing this skill. Another possibility is to temporarily render the skill inapplicable, in which case it is possible that a plan can not be found, if no other skills can produce a similar effect.

Since we are using STRIPS-like actions in this work, only one skill can be executed at a time. However, it would be possible for the robot to pick or place boxes in the rack while driving. This would require implementing durative actions, with the constraint that they can only be executed when the rack is involved. An alternative would be to specify separate, durative actions for picking and placing in the rack. This would also, however, also require multithreaded skill execution, which has not been implemented for this work.

6. Conclusion and future work

In this work we have shown that by implementing a set of robot skills, the task planning problem can be simplified to these skills, which is especially useful for logistic tasks, such as transportation or part feeding. We show that by utilizing an ad-hoc world model in conjunction with the skills, we can create robot programs that solve logistic tasks simply by specifying the desired goal setting – and inconsistencies between the modeled and real world, can be overcome simply by replanning.

The truly flexible robot in a future factory needs to solve a variety of dynamic tasks, with minimal intervention, and to be adapted to new tasks on the fly. We show that if the robot is equipped with skills, and supplied with a simple model of the factory, and the objects that are relevant for the robot, this is indeed possible. And even without human intervention, as the goals could be sent to the robot directly from the Manufacturing Execution System (MES).

Still, the implementation of *general* robot skills is an open problem. A skill such as *pick up <object>*, that can deal with any relevant object in the factory, would include advanced methods for object detection and pose estimation, as well as grasp detection. Our future work includes exactly the implementation of general skills, as well as a more generalized world model. We will investigate the use of these skills with task planning and other methods for task programming in the future.

APPENDIX

6.1. Logistic tasks PDDL domain

```
(define (domain box_logistics)
(:requirements :adl :typing)
(:types box robot location - object
        loc feeder - location)
(:constants unknown_loc robot_rack - loc)
(:predicates
  (rob-at ?r - robot ?l - location)
  (box-at ?b - box ?l - loc)
  (holding ?b - box ?r - robot)
  (empty-handed ?r)
  (empty ?o - object)
  (empty-loc ?l - loc)
  (full-loc ?l - loc))
  (at-empty-loc ?b - box)
  (at-full-loc ?b - box)
(:functions (loadlimit ?l - loc) (currentload ?l - loc) (total-cost))

(:action DRIVE_TO
:parameters (?to - location ?from - location ?r - robot)
:precondition (and (rob-at ?r ?from) (not (= ?from robot_rack)))
:effect (and (not (rob-at ?r ?from)) (rob-at ?r ?to)
        (increase (total-cost) 10)))

(:action PICK_UP
:parameters (?b - box ?l - loc ?r - robot)
:precondition (and (rob-at ?r ?l) (box-at ?b ?l) (empty-handed ?r))
:effect (and (not (box-at ?b ?l)) (holding ?b ?r) (not (empty-handed ?r))
        (decrease (currentload ?l) 1)
        (when (= ?l robot_rack) (increase (total-cost) 1))
        (when (not (= ?l robot_rack)) (increase (total-cost) 2)) ) )

(:action PLACE_AT
:parameters (?l - loc ?b - box ?r - robot)
:precondition (and (rob-at ?r ?l) (holding ?b ?r)
              (<= (+ (currentload ?l) 1) (loadlimit ?l)) )
:effect (and (box-at ?b ?l) (not (holding ?b ?r)) (empty-handed ?r)
        (increase (currentload ?l) 1)
        (when (empty-loc ?l) (at-empty-loc ?b))
        (when (full-loc ?l) (at-full-loc ?b))
        (when (= ?l robot_rack) (increase (total-cost) 1))
        (when (not (= ?l robot_rack)) (increase (total-cost) 2)) ) )

(:action UNLOAD
:parameters (?f - feeder ?b - box ?r - robot)
:precondition (and (rob-at ?r ?f) (holding ?b ?r) (not (empty ?b)) )
:effect (and (empty ?b) (not (empty ?f)) (not (at-empty-loc ?b))
        (increase (total-cost) 2)) )
)
```

ACKNOWLEDGMENTS

The authors would like to thank Matthew Crosby for his valuable insight in planning in general, and specifically PDDL encoding. This work has partly been supported by the European Commission under grant agreement number FP7-610917-STAMINA.

References

[1] S. Bøgh, M. Hvilshøj, M. Kristiansen, and O. Madsen, "Identifying and evaluating suitable tasks for autonomous industrial mobile manipulators (AIMM)," *The International Journal of Advanced Manufacturing Technology*, vol. 61, no. 5-8, pp. 713–726, Jul. 2012.

[2] M. R. Pedersen, D. Herzog, and V. Krüger, "Intuitive skill-level programming of industrial handling tasks on a mobile manipulator," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[3] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1972.

[4] C. Breazeal and B. Scassellati, "Robots that imitate humans," *Trends in Cognitive Sciences*, vol. 6, no. 11, pp. 481–487, 2002.

[5] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2, pp. 109–116, 2004.

[6] M. Lopes and J. Santos-Victor, "A developmental roadmap for learning by imitation in robots," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 37, pp. 308–321, Apr. 2007.

[7] A. Björkelund, L. Edstrom, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. Robertz, D. Storkle, A. Blomdell, R. Johansson, M. Linderoth, A. Nilsson, A. Robertsson, A. Stolt, and H. Bruyninckx, "On the integration of skilled robot motions for productivity in manufacturing," in *Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on*, May 2011, pp. 1 –9.

[8] V. Krüger, D. Kragic, A. Ude, and C. Geib, “The meaning of action: a review on action recognition and mapping,” *Advanced Robotics*, vol. 21, no. 13, pp. 1473–1501, 2007.

[9] H. Bruyninckx and J. De Schutter, “Specification of force-controlled actions in the "task frame formalism"-a synthesis,” *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 581 –589, Aug. 1996.

[10] T. Kröger, B. Finkemeyer, and F. Wahl, “Manipulation primitives — a universal interface between sensor-based motion control and robot programming,” in *Robotic Systems for Handling and Assembly*, ser. Springer Tracts in Advanced Robotics, D. Schütz and F. Wahl, Eds. Springer Berlin / Heidelberg, 2011, vol. 67, pp. 293–313.

[11] 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, Jun. 2005.

[12] S. Gottschlich, C. Ramos, and D. Lyons, “Assembly and task planning: A taxonomy,” *Robotics & Automation Magazine, IEEE*, vol. 1, no. 3, pp. 4–12, 1994.

[13] G. Biggs and B. MacDonald, “A survey of robot programming systems,” in *Proceedings of the Australasian conference on robotics and automation*, 2003, pp. 1–3.

[14] H. Mosemann and F. Wahl, “Automatic decomposition of planned assembly sequences into skill primitives,” *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 5, pp. 709 –718, Oct. 2001.

[15] Y. Lespérance, H. J. Levesque, F. Lin, D. Marcu, R. Reiter, and R. B. Scherl, “A logical approach to high-level robot programming–a progress report,” in *Control of the Physical World by Intelligent Systems: Papers from the 1994 AAAI Fall Symposium*, 1994, pp. 79–85.

[16] H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. B. Scherl, “GOLOG - a logic programming language for dynamic domains,” *The Journal of Logic Programming*, vol. 31, no. 1-3, pp. 59–83, 1997.

[17] D. Hähnel, W. Burgard, and G. Lakemeyer, “GOLEX—bridging the gap between logic (GOLOG) and a real robot,” *KI-98: Advances in Artificial Intelligence*, pp. 165–176, 1998.

[18] A. Ferrein and G. Lakemeyer, “Logic-based robot control in highly dynamic domains,” *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 980–991, Nov. 2008.

[19] M. Fox and D. Long, “PDDL2. 1: An extension to PDDL for expressing temporal planning domains.” *J. Artif. Intell. Res.(JAIR)*, vol. 20, pp. 61–124, 2003.

[20] J. Hoffmann, “The metric-FF planning system: Translating "ignoring delete lists" to numeric state variables,” *J. Artif. Int. Res.*, vol. 20, no. 1, pp. 291–341, Dec. 2003.

Fin.

SUMMARY

Efficient, transformable production systems need robots that are flexible and effortlessly repurposed or reconfigured. The present dissertation argues that this can be achieved through the implementation and use of general, object-centered robot skills.

In this dissertation, we focus on the design, implementation and uses of a robot programming paradigm, focused on robot skills, that facilitates intuitive and explicit task-level programming by laymen, such as factory workers, as well as ad-hoc task planning in the skill domain.

We show how these robot skills can be modeled and implemented, even on different robot systems. Furthermore, we show how laymen can intuitively program tasks on an advanced mobile manipulator, using the skills as the fundamental building blocks. Finally, we demonstrate how the same skills can be used for ad-hoc task planning, where the robot system instead constructs the task autonomously, exactly when it is needed.

It is the firm belief of this researcher that industrial robotics need to go in a direction towards what is outlined in this dissertation, both in academia and in the industry. In order for manufacturing companies to remain competitive, robotics is the definite way to go – and current industrial robot systems are plainly too difficult to program for new tasks on a regular basis.

ISSN: 2246-1248
ISBN: 978-87-7112-193-3

AALBORG **UNIVERSITY PRESS**