Robotics and Computer Integrated Manufacturing 53 (2018) 72–80

Contents lists available at ScienceDirect

Robotics and Computer Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Skill-based instruction of collaborative robots in industrial settings

Casper Schou[*], Rasmus Skovgaard Andersen, Dimitrios Chrysostomou, Simon Bøgh, Ole Madsen

Department of Materials & Production, Aalborg University, Denmark

ARTICLE INFO

Keywords:
Collaborative robots
Task-level programming
Robot skills
Intuitive robot instruction
Industrial assembly

ABSTRACT

During the past decades, the increasing need for more flexible and agile manufacturing equipment has spawned a growing interest in collaborative robots. Contrary to traditional industrial robots, collaborative robots are intended for operating alongside the production personnel in dynamic or semi-structured human environments. To cope with the environment and workflow of humans, new programming and control methods are needed compared to those of traditional industrial robots. This paper presents a *task-level programming* software tool allowing robotic novices to program industrial tasks on a collaborative robot. The tool called *Skill Based System* (SBS) is founded on the concept of robot skills, which are parameterizable and task-related actions of the robot. Task programming is conducted by first sequencing skills followed by an online parameterization performed using kinesthetic teaching. Through several user studies, SBS is found to enable robotic novices to program industrial tasks. SBS has further been deployed and tested in two manufacturing settings demonstrating its applicability in real industrial scenarios.

1. Introduction

In the last few years, we witnessed the *collaborative robots* era, where robots are not considered a bulky piece of machinery in the production line, but they actively share the workspace with human workers. They can work safely alongside humans without fences, improving the production flow and allowing the automation of new processes.

The scope of this work is to present an intuitive task-level programming tool that is decoupled from specific hardware components and bundles together expert knowledge and experience acquired by successfully solving real industrial scenarios. The definition and conceptual model of the *skills* used in this system have been previously discussed in [1]. There, the skill model is presented as a generic concept which can be used for task-level programming using both automatic and manual methods. In the current paper, we present a framework which is designed and implemented specifically for the intuitive manual instruction of collaborative robots by non-robot experts, and we thereby give an overview of the system's capabilities and its evaluation. The main contributions of this work are:

- A skill-based software system that facilitates easy and fast instruction of *collaborative robots* by inexperienced operators.
- A system that facilitates easy integration of existing off-the-shelf components in terms of hardware and software.
- Multiple successful demonstrations where the adaptability, scalability, and reconfigurability of the approach are investigated by deploying it in various real industrial settings.

Several manufacturers have within last decade released a new generation of robot arms as collaborative robots; e.g., Universal Robots UR3, 5, and 10, KUKA iiwa 7 and 14, and Fanuc CR-35iA. These robots offer safety features allowing them to operate safely in the proximity of humans. They also offer new HRI methods to make task programming easier. A common approach is kinesthetic teaching, where the operator physically interacts with the robot arm. However, the HRI systems supplied with these robots are confined to robot arm configuration only. Consequently, challenges involved in configuring an entire collaborative robot platform persist; e.g., sensor and tooling configuration. A few systems have been released to the market in recent years embedding tooling and sensors and thus presenting a more holistic system and offering more integrated control; e.g., Bosch APAS, KUKA KMRiiwa, ABB YuMi and Rethink Robotics Baxter and Sawyer. An example of an innovative solution is Intera Studio [2] from Rethink Robotics, which offers train-by-demonstration capabilities for non-expert users. However, all of the aforementioned systems are closed ecosystems, and thus bound to the hardware supplied with or supplementary to the robot. In the following, we present related work in Section 2 and we propose a method for manual parameterization of robot skills in Section 3. In Section 4, we discuss in detail the implementation of a robot operating tool utilizing the proposed manual parameterization method. Section 5 focuses on industrial applications of the robot

[*] Corresponding author.
E-mail address: cs@m-tech.aau.dk (C. Schou).

https://doi.org/10.1016/j.rcim.2018.03.008
Received 1 June 2016; Received in revised form 15 March 2018; Accepted 15 March 2018

operating tool while Section 6 summarizes a number of usability assessments made of the tool. We conclude with a discussion about the lessons learned in Section 7, an overview of the future work in Section 8 and the final conclusion in Section 9.

2. Related works

The term *cobot* (short for *collaborative robot*) was introduced in 1996 by Colgate et al. [3] as a term for passive mechanical devices used to aid humans in solving industrial tasks. Today the definition and consensus on the term are not clear, but two main approaches are present in research; truly collaborative robots and robot assistants. The former directly involves the human worker to collaborate in solving an industrial task as a team. Robot assistants, on the other hand, do not necessarily solve a task in direct collaboration with the human worker but serve as an assistant working alongside the human worker.

A key challenge for robot assistants in a dynamic workflow is instructing the robot. A survey on robot programming systems revealed two main approaches; manual programming and automatic programming [4]. In the latter, cognitive and highly autonomous systems are used to make the robot less dependent on instruction details. It relies on planning algorithms based on sensor inputs and a comprehensive world model. A well-known approach is the *sense-plan-act* (SPA) architecture [5], in which the robot switches between sensing, planning, and acting states. Recent research on cognitive robots has a high focus on knowledge gathering, formalization, and sharing [6–8]. Ontologies have become a widely adopted tool, and have resulted in a recent standard [9]. Combined with the idea of the semantic web, knowledge sharing between robots reaches a grand scale; e.g., as pursued in the RoboEarth project [10,11].

Contrary to reducing human intervention, recent research on manual programming instead focuses on bringing traditional online robot programming from engineers to the shop floor operators. This decreases the need for high-wage engineering hours and allows the operator to channel valuable process knowledge and experience into the instruction. A programming method that provides easy and quick instruction of robots is *task-level programming* [12]. It constitutes a higher abstraction level than traditional online robot programming, by focusing on task-related actions before low-level device control. Tasks are constructed by concatenating these actions, which are often referred to as *skills* of the robot [1]. Numerous representations of robot skills that would be suitable for task-level programming have been pursued during the last decades [13–17].

Stenmark [17] presents a skill definition which makes the interconnection of skills a controlled process using pre- and postcondition checks. In [18], Stenmark and Malec exploit their definition of skills to simplify robot programming using semantic knowledge stored in their Knowledge Integration Framework. Based on the knowledge, the system aids the user in configuring and parameterizing skill sequences in an offline programming environment. Despite the same goal, the framework we propose in this paper uses manual sequencing and parameterization of skills to ease online robot programming. Thus, extensive digital modeling of the scenario is not required.

Intuitive robot programming requires novel approaches of skill parameterization as opposed to traditional online programming using teach pendants [19]. For the spatial task information, such as target points and trajectories, a suitable approach for collaborative robots is kinesthetic teaching. In kinesthetic teaching, the human directly interacts with the robot and pilots it through the intended trajectory [20–22]. Kormushev et al. [20] present an approach where both position and force profiles are taught to the robot. Wrede et al. [22] present a method for kinesthetic teaching of redundant manipulators in spatially constraint workspaces where the user first teaches the allowable kinematic configuration. Subsequently, the user teaches the task-related trajectory and meanwhile the robot adheres to the previously taught kinematic configuration. Contrary to Kormushev et al. and

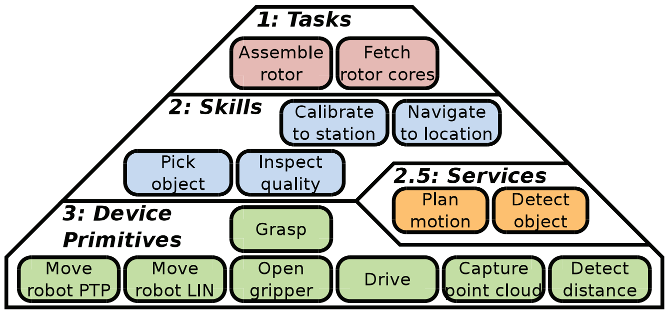

Fig. 1. The three abstraction layers, 1: Tasks, 2: Skills, and 3: Device Primitives. Services use multiple devices and provide higher-level functionality to the skill layer.

Wrede et al., manual teaching is an integral part of skill parameterization in our work. Moreover, kinesthetic teaching is, in our approach, combined with an intuitive graphical user interface to create a holistic method for sequencing and parameterizing skills.

3. Task-level programming using skills

Robotic skills combine low-level functionality from multiple devices into coherent higher-level programming blocks. This section briefly introduces the concept of robotic skills and how they are used for task-level programming. The skill model is based on the concept presented in [1] and is expanded here to include online manual parametrization.

3.1. Conceptual architecture for skill based programming

The skill based architecture has three layers of abstraction: Tasks, skills, and device primitives, as illustrated in Fig. 1. Device primitives are functions provided by a single device. e.g., a robot arm, a camera, or a force/torque sensor. In traditional robot programming, these functions are used directly by the user. In skill-based programming, they are used by the *skill developer* to design skills providing functionality on a higher level. The purpose of skills is to enable shop floor factory workers to program robots quickly and efficiently. Tasks can be programmed by sequencing and parameterizing a number of skills. The three layers can be defined as:

Device primitives: A collaborative robot is a composition of several devices; typically including a robot arm, a gripper, cameras, and sensors. Each device provides a certain set of functionalities denoted *device primitives*. Devices of the same type, e.g., gripper, derive the same primitives, e.g., grasp. Thus, device primitives abstract on specific implementation details on each device. Therefore, programming based on device primitives is independent of specific hardware components. Consequently, devices can be exchanged without affecting the above control layers, as long as any new devices provide the same device primitives.

Skills: A skill is a composition of sensing and manipulation primitives which together generate a change to the physical world. A skill is characterized by a number of parameters which must be specified before execution. The parametrization makes the generic skill specific in the sense that it now performs one specific change to the world when executed.

Tasks: Tasks are programmed by concatenating and parameterizing skills, and they are directly related to solving specific goals in the factory. The task layer is decoupled from the hardware layer of the robot.

Certain functionalities; e.g., motion planning, object detection, and pose estimation; do not come from a single device, and they do not change the state of any object. Thus, they do not qualify as skills or as device primitives. Appropriately, the *service layer* is introduced as an