Autonomous adaptation to changes in production demands with a reconfigurable robot workcell

Timotej Gašpar
Miha Deniša
Primož Radanovič
Aleš Ude
timotej.gaspar@ijs.si
Jožef Stefan Institute
Ljubljana, Slovenia

ABSTRACT

Current challenges in automation represent automating low-batch production processes where changes in the production parameters happen frequently. These type of production are often happening in Small and Medium-sized Enterprises, which have many time been dismissed as potential end user of automation technologies. This was mainly due to the high costs of setup, both in terms of the costs of the equipment and time required to set it up. In this paper we present a new type of reconfigurable robot workcell for fast set-up of automated assembly processes for SMEs. By developing passive reconfigurable elements and integrating intuitive programming by demonstration methodologies we were able to reduce the costs and set-up times for the automation of few-of-a-kind manufacturing processes without losing the flexibility of the system to cope with changes in market demands.

KEYWORDS

robotics, reconfigurability, ROS, assembly

1 INTRODUCTION

The trend of incorporation robots into manufacturing processes is on the rise. While high cost of process automation does not represent a significant challenge for large enterprises, *Smaller and Medium-sized Enterprises* (SME) might not undertake such an investments. Beside the price of the robots and the necessary accompanying hardware for automation, the cost of the time spend on the integration of robotic systems can also be high. Another hurdle for automatization of processes in SMEs is the need for quick adaptation to ever changing market demands.The paradigm of Reconfigurable Manufacturing Systems (RMS) [6] addresses the efficient and quick adaptation of the production process. Although a RMS can have a more complex design and achieve a lower throughput as classic automation approaches, they proved to be more applicable in processes with the need for often changes [10]. But in order to make RMS affordable for SMEs , a high investments cost of incorporating them in the manufacturing process needs to be avoided [3].

IS2019, October 7–11, 2019, Ljubljana, Slovenia, Europe

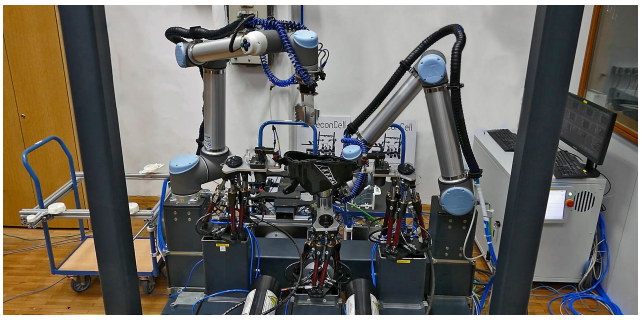

Figure 1: The proposed reconfigurable robot workcell executing an example assembly process.

The goal of the presented system is to offer a reconfigurable robot workcell in line with the RMS paradigm. The workcell must be appropriate for SMEs, where low-volume high-diversity production often takes place. The proposed systems combines a reconfigurable ROS-based software architecture and novel hardware elements that offer cost efficient solutions to reconfigurability. In addition, programming by demonstration methods for teaching of robots assembly skills are included in order to reduce the setup time.

While novel approaches in hardware design for reconfigurable robot workcells are presented in section 2, section 3 describes the software architecture of the cell. Section 4 presents technologies for fast set-up times and intuitive robot programming. Concluding remarks and implementation results are given in the last section.

2 RECONFIGURABLE HARDWARE

While designing the reconfigurable robot workcell in line with the RMS paradigm several aspects need to be taken into consideration: the desired physical properties (size, stiffness, robot workspace, etc.); available factory space; the integration of the workcell into the establish production process without too many significant changes; and the ability of the cell to quickly adapt to changing demands in the process. To ensure the workcell's ability for reconfiguration and adaptation in an affordable way, we introduced several passive reconfigurable hardware component as an alternative to off-the-shelf solutions.

The **reconfigurable frame** of the workcell connects the robot to peripheral modules. Several requirement need to be taken into account while designing the frame. While the cell's stiffness is paramount, as even small frame deformations can result in large positioning errors, the structure must be easily adaptable to ensure the needed reconfiguration. The affordability of such a solution should also be taken into account. To ensure the stiffness of the frame structure can be comparable with welded joints and at the same time enable simple assembly and modifications, rectangular steel beams in combination BoxJoint connectors [7] were selected.

Reconfigurable robot tools were also introduced. A mounted a tool exchange system at the robot's end effector enables a vast array of assembly operations by un/equipping tools needed for various steps in the process. Besides ensuring a stiff coupling between the robot and the tool, the exchange systems provides electrical power, Ethernet connectivity, and pressurised air to the tool.

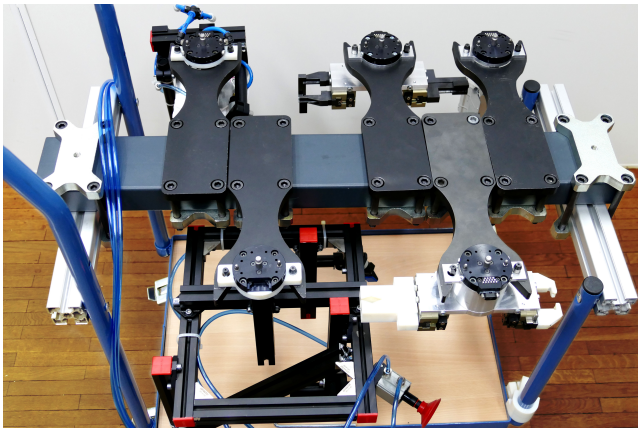

Figure 2: Various robot tools mounted on a rack. The robot can attach the one needed for the current task.

Special "**Plug & Produce**" (PnP) connectors were developed to ensure the connectivity to peripheral modules. These peripheral modules are crucial in a reconfigurable environment, as they enhance the cell with various functionalities. Used modules can include various fixtures, material flow management, tool storage, etc. These modules need to be introduced or removed from the workcell as quick as possible with as little disturbance to the process as possible. The design of the PnP connectors provides a highly repeatable, stiff, and quick mechanical coupling of the modules to the cell. Beside a mechanical coupling, connectors enable the transfer of data, pressurised air, and electrical power. This enables the peripheral modules to be self sufficient and connect to the overall structure of the cell as quickly as possible. While PnP connectors allow us to introduce new modules into the workcell such modules often need to be introduced manually, and can not be regarded as fully autonomous.

A concept of **passive reconfigurable elements** introduces needed reconfiguration into the cell, while reducing the cost of the elements. In contrast to standard off-the-shelf solutions, which often include active components, these passive reconfigurable elements do not contain any actuators or sensing equipment. By removing these components the price tag is lowered. To compensate for the missing sensors and actuators in these passive elements, robot is used in the reconfiguration step. A number of passive hardware components were used in various assembly operations. One example of a passive reconfigurable element is a passive rotary table (depicted in Fig. 3). By rotating the table, the workpiece on the table is oriented in the desired way. This is achieved by releasing the table's brakes, re-orienting it by the robot arm, and engaging the brakes as the desired orientation is reached. The last orientation of the table is stored by storing the robot's position.

3 RECONFIGURABLE SOFTWARE

Providing connectivity with respect to the data flow is another paramount issue for a proper workcell. Peripheral modules should be connected to the workcell and between each other, in order to receive and broadcast data and instructions. The data should be parsable by all software components within the system. To ensure the software modularity and connectivity, the proposed software architecture is ROS-based. The software system architecture of the workcell is depicted in Figure 5.

A **robot workcell ROS backbone** was implemented to ensure the needed connectivity. Just the data flow between all the modules is not enough to achieve the desired modularity of the system. The data should be structured in a way that is parsable by all the modules in the workcell. The suitable framework is offered by the Robot Operating System (ROS) [8], which enables the development of software components that need to share data over the common network. In addition it allows monitoring and controlling the complete workcell.

ROS-based modules ensure that they are all connected within the workcell through the ROS network. All modules are equipped with the computational hardware that enables running ROS *nodes*. This means each module's data and functionalities are available through the workcell ROS network. They are denoted as Micro computer in Fig. 5. A top-level task scheduling software can controll all modules as soon as they are plugged into the cell. They are connected to the cell using the described hardware components (PnP connectors or tool exchange systems).

Low level real-time robot control is another crucial part of the proposed reconfigurable workcell. To follow the previously described paradigms of seamless integration of all the hardware components in the workcell, robots should be treated as a ROS enabled module. While industrial robots are equipped with a control box that provides real-time control, most of them do not offer support for running ROS nodes and in turn are not able to communicate over the ROS network. A special communication layer that connects the robot module to the rest of the ROS network was implemented. In order to not make the workcell robot-specific an abstraction layer that supports different types of robots was introduced. It enable programming of new strategies through a suitable control interface and various trajectory and feedback control strategies. Again, independently of the selected robot. This abstarct layer enchances the overall modularity of the cell.

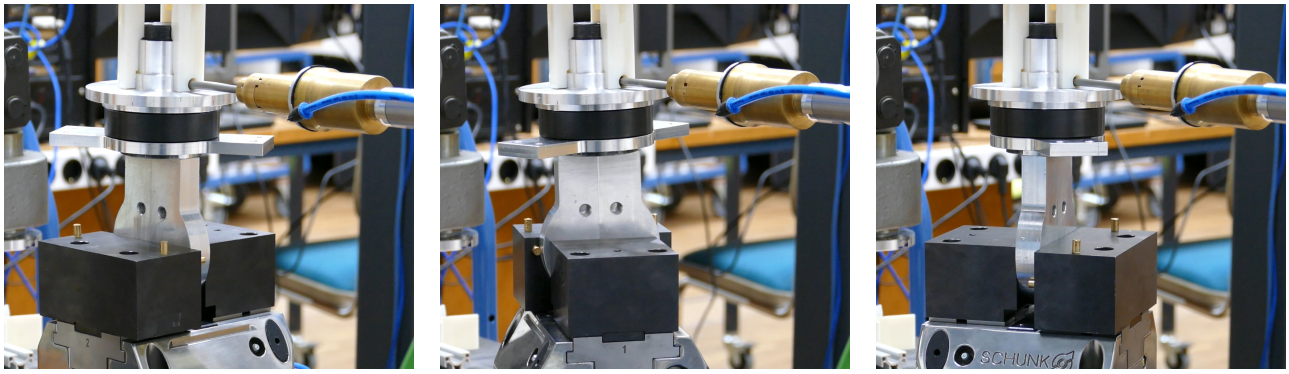

Figure 3: An example use of the passive rotary table. As can be seen in the figure, the table is being used to fasten screws on 3 different sides of a workpiece. As it would be impossible to reach the object on all three sides with the screwdriver, due to the kinematic restrictions, the table needs to be used.

4 ASSEMBLY SEQUENCES

During a set-up of a classic assembly workcell, a significant portion of the time is dedicated to determining, writing and compiling of the assembly sequence. In order to reduce the set-up time and enable short reconfiguration times between different assembly processes, this should be done as fast as possible. In this section we present a set of technologies that facilitate and accelerate the programming of robot workcell assembly operations.

Learning of **assembly skills through programming by demonstration** (PbD) enables defining the robot motions for a complete assembly process in an intuitive and faster way for non-expert users. PbD provides an approach to define these motions in a natural way and avoids coding complex programs in a robot-oriented programming language [1, 2, 4]. The two present PbD approaches are kinesthetic teaching and remote guidance.

With kinesthetic guidance the user moves the robot by physically guiding through its workspace and thus showing the desired movement. This approach is commonly used in collaborative robotics as it is effective to use with robots which have torque-controlled actuators [5]. The quality of the dynamic model and torque sensors greatly impact the ease of guidance and the needed physical effort. This in terms effect the quality of the shown movement and the smoothness of the demonstrated path.

While useful, the drawbacks of non-perfect kinesthetic guidance represent an inconvenience when working towards methods to shorten times of robot programming. As a result, a large amount of time can be spend to achieve the desired movement and/or configuration of the robot. To mitigate these drawbacks a remote control interface was developed and integrated in the workcell. A displacement of the analogue sticks of a consumer grade joystick was mapped to the Cartesian space velocities. This allows the user to control the robot in a smooth way and can mitigate the drawback of kinesthetic guidance when needed.

A **database of assembly skills** acquired during PbD should be be accessible throughout the entire software framework of the workcell. In order to handle storing and loading of the learned skills, MongoDB database was integrated into our system. Whenever a new skill is learned, a new named entry is created in the database. An assembly sequence can then read the desired database entry from the database and move the robots accordingly. If we wish to update a certain skill, we can simply overwrite the entry with a newly modified skill. In this way, we avoid changing the top-level assembly sequence program.

Figure 4: A consumer grade joystick interface that we used to perform precise motions of the robot in Cartesian space.

State machine assembler is a crucial part of any workcell. An engine for state machine code generation was developed to further accelerate the programming process of the workcell assembly sequence. While there are numerous ROS-based packages aimed at facilitating the high-level task programming by using state machines, defining complex robot behaviours with these tools can be complex. It requires a programmer to dedicate his attention to the structure of the state machine, the basic code, and the programming language syntax. To expedite and enhance this process, a method for code generation, A meta-scripting and templating method was was developed to speed up the process. The details on this are omitted in this paper and the reader is referred to our previous work on this topic [9].

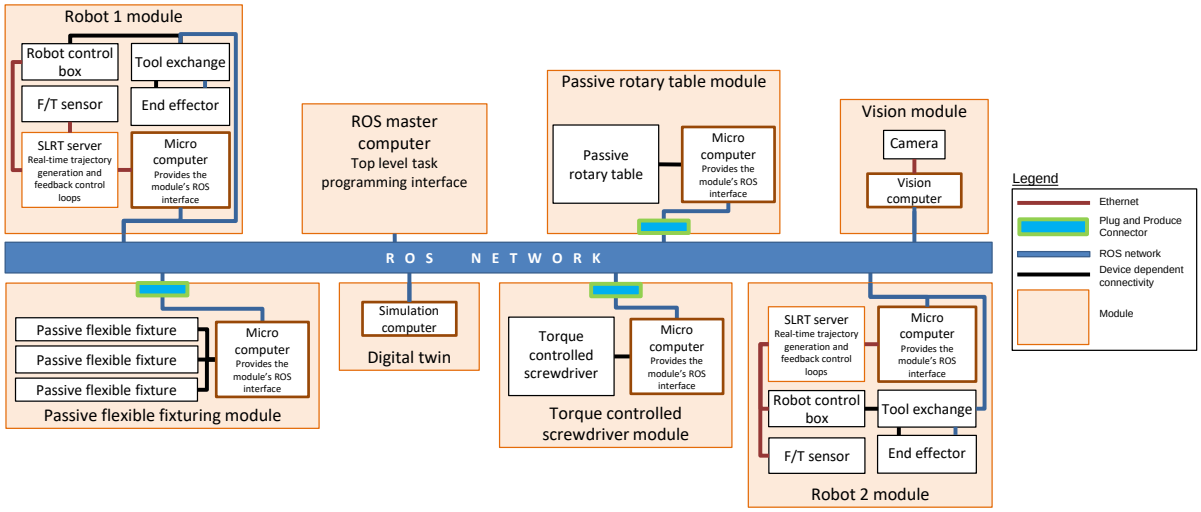

Figure 5: Software architecture for the reconfigurable robot workcell with various software and hardware modules.

5 CONCLUSION

In this paper we present a new type of robot workcell which is highly reconfigurable with innovative hardware concepts and components with a ROS-based software backbone. Throughout the work that lead to the presented results we focused not only towards providing methods for autonomous reconfiguration of the cell in order to adapt to production changes, but also to shorten set-up times by implementing various programming by demonstration technologies. In order to show the industrial relevance of our work we evaluated the proposed paradigms, the underlying technology and the overall quality of the cell through the implementation of various use-cases. The use-cases were provided by SMEs from different fields of industry and our task was to automate part of the production line that is currently done either manually or does not posses the desired flexibility. These use-cases range from the (1) assembly of automotive headlights, (2) the assembly of linear drives, (3) the assembly of a robotic gripper, (4) assembly of airport runway lights and finally the (5) assembly of printed circuit boards (PCBs). The successful implementations provided us with the overall proof that the developed solution are of interest in the industry. We were also able to acquire the first reference key performance indicators, e. g. cycle time, reconfiguration time, setup-time, etc. Throughout the implementation of the various use-cases some of the key equipment stayed the same (i.e. robots, tool rack, etc.), however other parts of the cell were reconfigured according to the requirements of each experiment. Some application-specific periphery modules were either added or removed.

ACKNOWLEDGEMENTS

This work has received funding from the EU's Horizon 2020IA ReconCell (GA no. 680431) and the the EU's Horizon 2020 RIA AUTOWARE (GA no. 723909).

REFERENCES

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57, 5 (2009), 469–483.

[2] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. 2008. Robot programming by demonstration. In *Springer handbook of robotics*. Springer, 1371–1394.

[3] Thomas Dietz, Ulrich Schneider, Marc Barho, Susanne Oberer-Treitz, Manuel Drust, Rebecca Hollmann, and Martin HÃđgele. 2012. Programming System for Efficient Use of Industrial Robots for Deburring in SME Environments. In *ROBOTIK 2012; 7th German Conference on Robotics* (2012). VDE, 1–6.

[4] Rüdiger Dillmann. 2004. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems* 47, 2-3 (2004), 109–116.

[5] M. Hersch, F. Guenter, S. Calinon, and A. Billard. 2008. Dynamical System Modulation for Robot Learning via Kinesthetic Demonstrations. *IEEE Transactions on Robotics* 24, 6 (2008), 1463–1467.

[6] Yoram Koren, Uwe Heisel, Francesco Jovane, Tosliiniichi Moriwaki, Guenter Pritschow, A. Galip Ulsoy, and Hendrik M.J. Van BrÃijssel. 1999. Reconfigurable Manufacturing Systems. *CIRP Annals - Manuf. Technol.* 48, 2 (1999), 527–540.

[7] Alison Millar and Henrik Kihlman. 2009. *Reconfigurable flexible tooling for aerospace wing assembly*. Technical Report. SAE Technical Paper.

[8] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. 2009. ROS: An open-source Robot Operating System. In *ICRA workshop on open source software*. Kobe, Japan, 5.

[9] Barry Ridge, Timotej Gaspar, and Ales Ude. 2017. Rapid state machine assembly for modular robot control using meta-scripting, templating and code generation. In *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. Birmingham, UK, 661–668.

[10] G. Zhang, R. Liu, L. Gong, and Q. Huang. 2006. An Analytical Comparison on Cost and Performance among DMS, AMS, FMS and RMS. In *Reconfigurable Manufacturing Systems and Transformable Factories*, Anatoli I. Dashchenko (Ed.). Springer Berlin Heidelberg, 659–673.