Proceedings of the 2017 18th
International Conference on Advanced Robotics (ICAR)
Hong Kong, China, July 2017

Rapid Hardware and Software Reconfiguration in a Robotic Workcell

Timotej Gašpar, Barry Ridge, Robert Bevec, Martin Bem, Igor Kovač and Aleš Ude
Department of Automatics, Biocybernetics, and Robotics
Jožef Stefan Institute, Ljubljana, Slovenia
{*timotej.gaspar, barry.ridge, robert.bevec, martin.bem, ales.ude*}*@ijs.si*

Žiga Gosar
Elvez d.o.o.
Višnja Gora, Slovenia
ziga.gosar@elvez.si

***Abstract*— In an increasingly competitive manufacturing industry it is becoming ever more important to rapidly react to changes in market demands. In order to satisfy these requirements, it is crucial that automated manufacturing processes are flexible and can be adapted to new production requirements quickly. In this paper we present a novel automatically reconfigurable robot workcell that addresses the issues of flexible manufacturing. The proposed workcell is reconfigurable in terms of hardware and software. The hardware elements of the workcell, both those selected off-the-shelf and those developed specifically for the system, allow for fast cell setup and reconfiguration, while the software aims to provide a modular, robot-independent, ROS-based programming environment. While the proposed workcell is being developed in such a way as to address the needs of production-oriented SMEs where batch sizes are relatively small, it will also be of interest to enterprises with larger production lines since it additionally targets high performance in terms of speed, interoperability of robotic elements, and ease of use.**

***Index Terms*— Reconfigurable, Robotics, ROS**

I. Introduction

In industry, particularly in the realm of *small and medium-sized enterprises (SMEs)*, rapid changes in market demands lead to decreasing product lifetimes and also to more frequent product launches. SMEs must react quickly, efficiently, and in an economically viable way to such market changes. Although robots have been highly successful in many industrial production processes when applied to complex repetitive tasks with long production runs and high unit volume, the frequent shifts in the required product type or in the number of required products, as dictated by the market forces to which SMEs are exposed, often preclude them from exploiting any potential benefits such robots might provide.

These so-called *few-of-a-kind* assembly production scenarios [1] are typical of SMEs and given that SMEs are the "backbone of the manufacturing industry", e. g. providing some $\sim 45\%$ of the value added by manufacturing in the European Union [2], it would be highly beneficial if rapidly reconfigurable robotic workcells could be developed specifically to ease the burden of such use-cases.

The main hindrances to further uptake of SME robot production are the complexities involved in setting up existing solutions, since they usually require expert knowledge as well as significant time for testing and fine-tuning. Since SMEs usually do not have such expertise available, they avoid introducing such solutions, even when they are economically justifiable. Looking at such robotic systems in more detail, we can recognize that these problems are due to the time costs involved in re-configuring and re-programming the robot workcell for new assembly tasks, which are often too prohibitive to make the application of robots profitable.

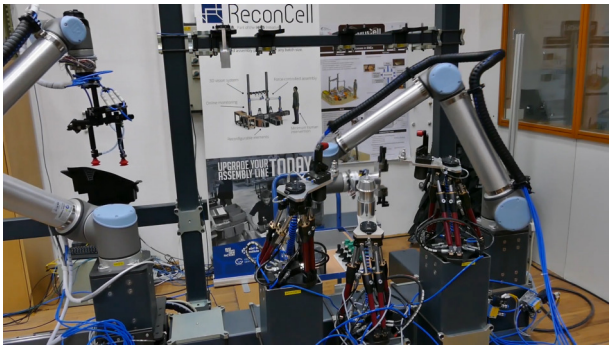


Fig. 1: The proposed workcell prototype executing an automotive housing assembly use case.

A. A Reconfigurable Workcell for Automated Assembly

In this paper, we present the design of a new kind of autonomous robot workcell that is attractive not only for large production lines, but also for few-of-a-kind production [3]. We propose reducing set-up times by exploiting a number of hardware and software technologies, some of which were partially developed in prior work, and some of which are novel contributions in this paper particular to the proposed workcell design. The main novelty of the workcell lies in the automatic reconfiguration of passive fixtures and other passive elements in the cell, which can be performed by the robots installed therein. This reconfiguration process allows the robots to autonomously configure their workspace and prepare the workcell for the execution of new assembly tasks. This way, the set-up times and costs of preparing a new production line can be greatly reduced.

We describe the application of various reconfigurable ele-

978-1-5386-3157-7/17/$31.00 ©2017 IEEE

ments in the workcell, including the use of a reconfigurable steel frame structure with modular beam connectors for both high flexibility and stiffness, robot arms with quick pneumatic tool changers, *plug and produce (P&P)* connectors for simplified coupling of system infrastructure, reconfigurable passive fixtures, and a passive linear rail unit for rapid robot-driven automatic relocation of the robots. The unique combination of these technologies is, to the best of our knowledge, a novel contribution to the field.

The rest of the paper is organized as follows. We first discuss related work and compare our design choices to the state-of-the-art. Next, in Section II, we describe the reconfigurable hardware of the workcell and in Section III, the reconfigurable software system architecture. In Section IV, we describe the application of these new reconfigurable technologies in a focused industrial use-case for automotive light production, demonstrating how the system can be used to reduce production costs by increasing both efficiency and flexibility. Finally, in Section V we conclude and discuss our plans for future work.

B. Related Work

A number of surveys have been released in recent years documenting the development of reconfigurable robotic systems, both in research and in industry [4]–[6]. A prominent example amidst the research on modular reconfigurable robotic systems is the work of Chen [7], [8], who placed a specific emphasis on finding optimal module assembly configurations from a given set of module components for a specific task. His subsequent work on the design of a reconfigurable robotic workcell for rapid response manufacturing [9] is of particular relevance with respect to the workcell proposed in this paper. However, while that work involved the development of a workcell containing hardware elements that can be rapidly reconfigured *manually*, our proposed workcell focuses on introducing hardware elements that can be rapidly reconfigured *automatically* by the system itself with respect to a family of parts within a given product line and its respective assembly task.

In our system this is made possible due to the application of reconfigurable fixtures known as *hexapods* (*c.f*. Section II-C), the use of which in a robot-guided reconfigurable assembly system was first proposed in the work of Gödl *et al*. [10]. A similar reconfigurable fixture concept was later described in the work of Jonnsson & Ossbahr [11] and Salminen *et al*. [12] in the context of the production of bogies in the railway industry. In this paper, we describe a refined version of this concept, involving smaller units, brought to bear on the particular use-case of the production of headlamps for the automotive light industry (*c.f*. Section IV), with a view towards expanding its application to further industrial use-cases in the area of automated robot assembly. We also propose enhancements to the original concepts proposed in [10] via a number of separate hardware augmentations within the workcell as described in Section II.

In the work of Krüger *et al*. [1], [13], a set of methods was developed to facilitate the set-up of complex automated assembly processes, such as they arise in the standard assembly benchmark known as the Cranfield benchmark [14]. The proposed set of methods included pose estimation and tracking of parts using a 3-D vision system, fast and robust robot trajectory adaptation using *dynamic movement primitives (DMP)* [15], and ROS-based software control and state machine programming [16]. In this work we build on these approaches and supplement them with the ability to automatically reconfigure the workcell, as well as with the integration of CAD-based product design for assembly that takes into account the requirements of robotic manipulation. Moreover, the proposed system advances beyond synthetic benchmarks such as the Cranfield benchmark and demonstrates the viability of the system on actual industrial use-cases.

II. Reconfigurable Hardware

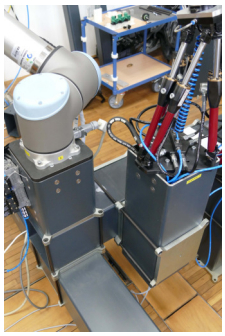
The proposed robotic workcell is in large part constructed of modular hardware that allows for fast and easy reconfiguration; from the structural frame to the fixtures, end-effectors, tool exchange system, P&P connectors, and other peripheral devices. With this approach we make it possible to use the proposed workcell in a wide range of industrial applications and environments. Furthermore, we also make it relatively easy to alter its shape and purpose within those environments. In the following subsections we will give an overview of the technologies and solutions that were used to achieve said hardware reconfigurability.

A. Reconfigurable Frame

The frame structure of the workcell is made of rectangular steel beams that are connected via the *BoxJoint* patented modular frame coupling technology [17], as shown in Fig. 2a. The advantage of this technology is that a workcell frame can be easily configured into a large variety of shapes. Typically in industry, purpose-made frames are either welded into the desired shape or an aluminum modular system is used to construct the desired frame. The issue with specially designed welded frames is that they are not reconfigurable, while the issue with aluminum frames is the fact that aluminum is less thermally stable. By using steel beams as the core element of the modular frame for the cell, we make the cell more stiff, robust, and less susceptible to deformations due to changes in temperature. The latter feature also makes the cell a viable solution for robot welding tasks.

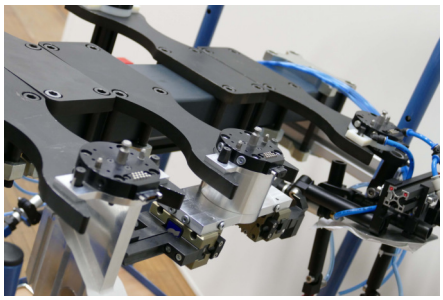
B. Tool Exchange System

Different workpieces demand access to a variety of different robot tools depending on the tasks that are required to be performed on them. In order to ensure that such tasks can



(a) The BoxJoint coupling system holding holding together steel beams.

(b) The passive reconfigurable fixture being reconfigured by a UR10 robot.



(c) Tool exchange stands holding various end-effectors for the robot.

Fig. 2: Reconfigurable hardware in the ReconCell robot cell.

be efficiently and precisely executed in the workcell given the limited number of robots therein, we developed a stand where different end-effector tools can be placed (*c.f.* Fig. 2c). The robots can then pick the appropriate tool for the different stages of the task. So if the task to be performed is the assembly of different pieces, the robots can equip different grippers for each piece that needs to be assembled into the given workpiece. If reconfiguration of the cell is needed to assemble a different workpiece, new end-effectors can be placed on the pre-prepared robot tool stands. The stands that hold the end-effectors were custom designed and developed and are mounted directly on the steel beam frame with the same BoxJoint technology that was used to assemble the frame.

C. Reconfigurable Fixtures

In small production lines where shifts in demand occur frequently, it is difficult to maintain a robotic workcell due to the time it takes to re-adjust all of the fixtures within it. In order to help mitigate against this specific problem, we added passive reconfigurable flexible fixtures to the workcell. The fixtures are designed in a Stewart platform-esque configuration with six legs, hence the name "hexapod". These fixtures can be dynamically reconfigured by the robot arms on demand by connecting a robot to a fixture via the tool

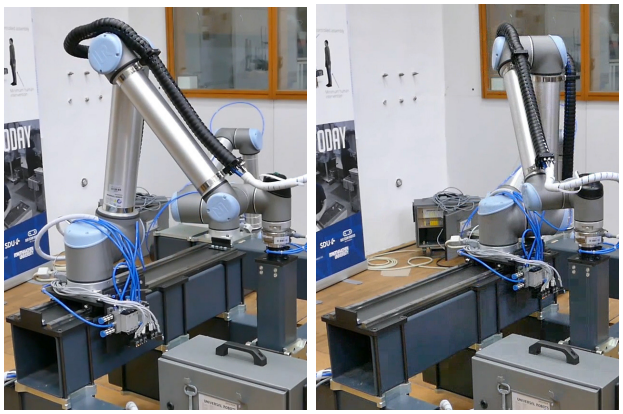


Fig. 3: Robot base reconfiguration using the passive linear unit by connecting the tip of the robot to the frame.

exchange system, releasing the fixture brakes, manipulating the fixture into the desired pose, re-applying the brakes, and disconnecting the robot from the fixture. This process is illustrated in Fig. 2b. By designing the fixtures to be passive and sensor-less it is possible to manufacture them for relatively low cost. With this technology, not only can the workspace be easily adapted to cope with the change in the workpiece, but the workspace can be adapted by the robotic workcell itself.

D. Passive Linear Unit

To make it possible to enlarge the work area of the cell the robot was mounted on a custom passive linear rail unit. This way the robot base can be moved along a linear axis. The purpose of the passive linear unit is to expand the work area of the robot within the work cell with minimal additional cost. Conventional actuated solutions are extremely expensive and thus inappropriate for SMEs. A way of reducing the cost without significantly reducing the functionality, is to omit actuation and position sensing from the linear unit. The robot itself is used to propel itself along the linear rail. This is achieved by connecting the tip of the robot to the frame, with the use of the previously discussed tool exchange system, before using the robot actuators and position control to move the base of the robot. This approach is appropriate for applications where the need to move the robot is relatively infrequent.

III. Reconfigurable Software System

The introduction of a robotic system into a production line represents a big investment and change for small or medium sized companies. The high costs usually stem from the price of the necessary hardware and the time spent for the integration of the robotic system into the production line. One of the time-consuming aspects of the integration involves the programming of task sequences for the robots involved in the production process. The programming process

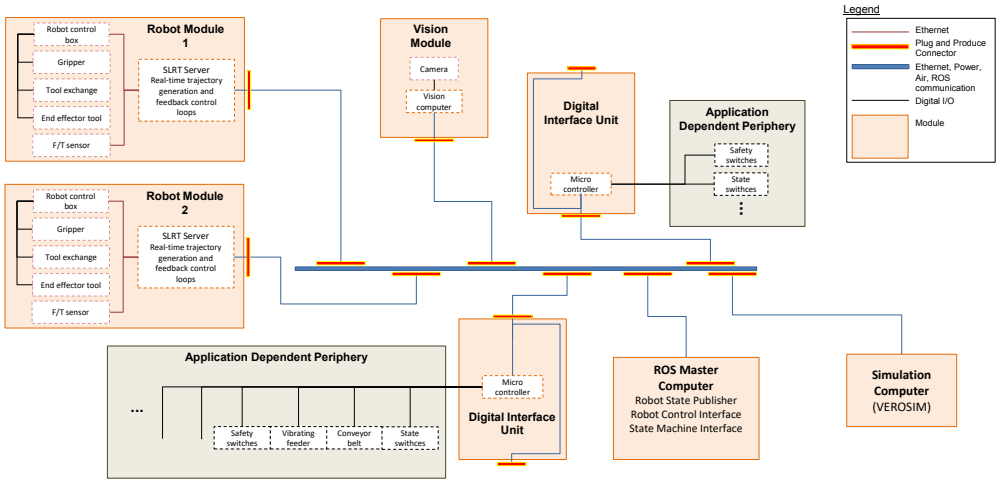


Fig. 4: Schematics of the workcell software and hardware architecture.

is usually done either via on-line programming using a robot teach pendant directly connected to the robot controller, or via off-line programming in a simulation environment, both of which require knowledge of the specific robot system. With this in mind, we developed a software system that would facilitate the programming of robot tasks regardless of the robot system. The software system is designed to be distributed, modular and offers seamless adaptation of the robot cell. The package also provides the necessary tools to enable simple, intuitive programming of robot tasks.

Our system was build within the *Robot Operating System (ROS)* framework [16], where the *Matlab Simulink Real-Time (SLRT)* platform [18] was used to develop the hard real-time components of the cell. We chose ROS because it provides a reliable open source framework, the capacity for cross-platform and multi-language flexibility, and a vast number of useful libraries and tools. However, ROS in its current form does not provide any form of hard real-time implementation, which is a crucial requirement for reliable and accurate robot control. The second iteration of ROS, named ROS 2 [19], has been proposed to address this issue, but it is currently still in the alpha stage under heavy development. That is why SLRT was used to build a robot control server (SLRT server) in the current version of the workcell, which is responsible for high frequency real-time trajectory generation and force control.

A. System Architecture Overview

The SLRT server connects directly to the robot controller via Ethernet and is responsible for communicating with the robot. It operates as a proxy, offering advanced trajectory generation methods and feedback control strategies. Standard robot controllers usually only offer basic control methods, but our approach, on the other hand, gives the system great flexibility, since any control method can be easily and quickly implemented on the SLRT server without having to modify the robot controller, which may be difficult or impossible to modify in any case. It also makes our system independent of the robot. For a new robot to be integrated within the cell, its controller must offer the ability to receive joint configurations over Ethernet and then only the kinematic model must be adapted on the SLRT server.

The SLRT server also connects to some other measurement units (e.g. force/torque sensors) that can be used for closed loop control policies (e.g. force control). We developed a ROS package that acts as a driver for the SLRT server and provides tools and functionalities for running, reconfiguring and calibrating the robot cell. The ROS architecture provides the versatility needed for connecting different modules to the workcell and our tools facilitate programming of the desired workcell task using data from the connected modules. The nodes from our package run on a ROS Master Computer, where ROS core is also run in order to synchronize communication throughout the system. Modules can also have their own individual nodes running on their respective computers. The main functionalities of the package will be briefly explained in the following subsections.

Figure 4 shows elements of the software architecture of a typical workcell design. In the schematics there are two "Robot Modules", each representing a robot with its robot control server, one additional measurement unit, and all of its tools and grippers. The "ROS Master Computer" refers to the computer in the system that runs the ROS core and our

ROS package. In order to access various periphery from the workcell a "Digital Interface Unit" is used for bridging the connection from PLC to ROS. Other typical modules include a "Vision Module" and a "Simulation Computer". The Vision Module represents a user programmable processing unit for typical vision tasks like quality control and part detection for manipulation. The Simulation Computer offers a dynamic simulation environment provided by the VEROSIM software package [20], where the production can be planned and evaluated. Depending on requirements, the workcell design can be adapted by adding or removing various modules.

B. Simulink Real-Time Server

A core part of the robot module is the previously mentioned SLRT server, which has been developed to ensure that the robot can be reliably controlled with the maximum frequency of the provided robot controller (125 Hz in the case of UR-10 robots). The SLRT server sends the robot controller the desired joint configuration. The inner loop of the SLRT server can run on a higher frequency than the robot controller if we have a measurement unit connected to it with a higher frequency readout. In our case, where a 1000Hz force/torque sensor has been connected to the SLRT for force control, the SLRT server runs at 1000 Hz as well. In the case of a UR-10 robot, the SLRT server can process 8 samples per robot controller loop for a better estimation of the force/torque derivative. This provides a better filtered force/torque signal for high quality force control, improving the stability, accuracy and speed of robot trajectories in contact with the environment.

As mentioned before, a real-time robot control server was developed as a proxy between the ROS system and the robot controller. The main motivation behind this approach is that trajectory generation and control is handled by a hard real-time system that is robot independent. This way, the trajectory generation and control algorithms can be developed independently of the used robot. Various trajectory generation algorithms have been implemented so far to cover the most common robot motion needs in the context of automated assembly. These are:

- trapezoidal speed profile in joint space,
- minimum jerk for position and minimum jerk SLERP for orientation trajectories in Cartesian space.
- admittance force control [21],
- joint space dynamic movement primitives for free-form movements [22],
- Cartesian space dynamic movement primitives free-form movements in Cartesian space [15].

C. ROS Software Package

To allow the robot workcell to be accessed, controlled and calibrated within the ROS environment, various ROS nodes have been developed to offer an interface to the previously discussed SLRT server and other modules in the workcell. In this section we will describe their core capabilities and inter-functionality.

1) SLRT State Publisher: The purpose of this ROS node is to read the data stream from the SLRT server and publish it within the ROS network via ROS topics using conventional ROS messages. The published data covers all of the relevant information about the robot, such as joint positions, velocities, payload, tool information, forces from the force/torque sensor, and several control flags for different control strategies. One of the vital packages in ROS is *tf2*, which is used for keeping track of multiple coordinate frames in the system. We use the ROS *robot_state_publisher* package, which latches onto the joint position topic and, using the robot kinematic description from the URDF file, tracks coordinate frames in all joints of the robot system.

2) Action Servers: These are nodes built with the ROS *actionlib* stack that handle communication to the SLRT server and are used to trigger robot motion and monitor the progress of the trajectory. The *actionlib* provides a reliable client-server interface where the lower-level communication and logic is handled by the action server node, while the client simply triggers the motion by sending a goal request. After the action server node detects that the robot motion is finished, it sends a result message to the client. The unique advantage of using the actionlib stack instead of ROS services to trigger robot motion is that it offers preemption requests and feedback messages during execution. This means that a client can preempt motion that is already being executed and also receives information from the action server during execution such that the motion can be monitored.

Each trajectory generation algorithm that is implemented on the SLRT server is offered as a separate action server with its own goal, feedback and result messages. The low level logic of all of the action servers ensures that only one motion can be executed at a time.

3) ROS Services: Services provide an interface for handling short duration tasks such as changing the state of a digital output. Our ROS package so far includes services that:

- change the robot mode from position control to gravity compensation mode,
- trigger direct joint control on the SLRT server,
- set digital outputs on the robot controller.

As with Action servers, ROS services are very practical for programming the top-level robot task program.

4) Database: A robot workcell needs to keep track of its state at all times, be it during operation or downtime. Some sort of persistent storage is required, however none of the basic ROS functionalities, such as the ROS parameter server, offer that. We decided to follow a common approach with wide support in the community and implemented a MongoDB database. There are different ROS packages offering simple interfaces in C++ and Python for all clients in the

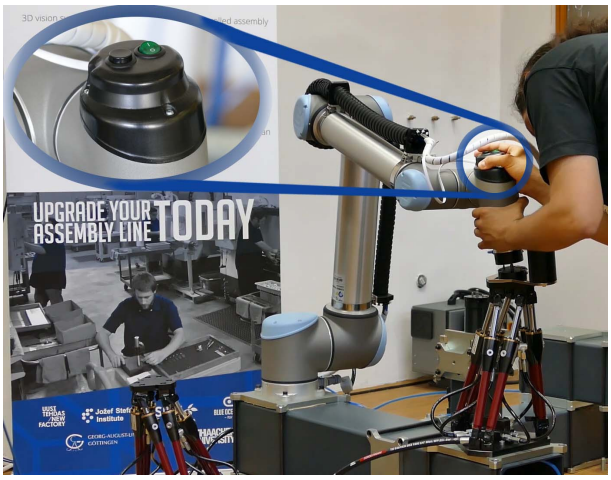


Fig. 5: Robot capture demonstration with close-up of custom-designed capture button and switch system.

ROS network to read from and write to the database using ROS type messages.

The information stored in the database consists of poses of different elements in the workcell, saved robot configurations in Cartesian or joint space, calibration parameters and other parameters. We have developed a node that reads transforms from the database and publishes them using *tf2*. These transforms can then be used for robot assembly tasks.

5) Robot Capture: This is a versatile program for capturing and storing various robot related configurations in the database. It is commonly used in conjunction with the kinesthetic guiding of the robot, where the programmer of the robot workcell can freely move the robot in its workspace and then save the points of interest. The process of kinesthetic guiding and data capturing is eased by the addition of a hardware capture button and switch system that was custom-designed for the workcell and that is attached to robot arms in the workcell as illustrated in Fig. 5. When the robot capture program is active, the capture button can be pressed during kinesthetic guiding to trigger data recording. The program allows for the robot end-effector tool center point to be saved in Cartesian coordinates, as well as the robot joint state (position and velocity), while also providing a robot-to-robot calibration mode. The first functionality is commonly used for calibrating the workcell state (reconfigurable fixture positions, tool pick-up slots) and for saving pick-and-place poses of the robot assembly task. The saved joint configurations are generally used for path planning and the robot-to-robot calibration mode is used to define the relative transformation from one robot base to another, when more robot modules are present in the workcell.

D. Additional Functionalities

1) Programming by Demonstration: Robot programming by kinesthetic guiding has been in increasing demand in recent years. More and more robot manufacturers are starting to implement this functionality on their robots. It provides an intuitive method for teaching the robot how to move to either points in Cartesian or joint space or over whole trajectories. The robot used for our work allows kinesthetic guidance via the so called *Gravity compensation mode*. In this mode, the robot controller estimates the input torques to the robot motors to compensate for the effect of gravity on the robot links. The gravity compensation mode should always take into consideration the payload on the robot end-effector, otherwise the calculated torques would not be correct, which would potentially cause the robot to drift from its position.

2) Adaptation of Learned Trajectories: When programming a robot task via kinesthetic guidance in physical contact with the workpiece, it is very often the case that the learned trajectory is not ideal or optimal. By learning the executed forces on the end-effector of the robot and by using admittance control, we can then use the displacement due to the force error as a correcting offset to our DMP encoded trajectory. By adapting the trajectory to follow a desired force profile we can achieve better and faster executions of our first demonstrated trajectory [21].

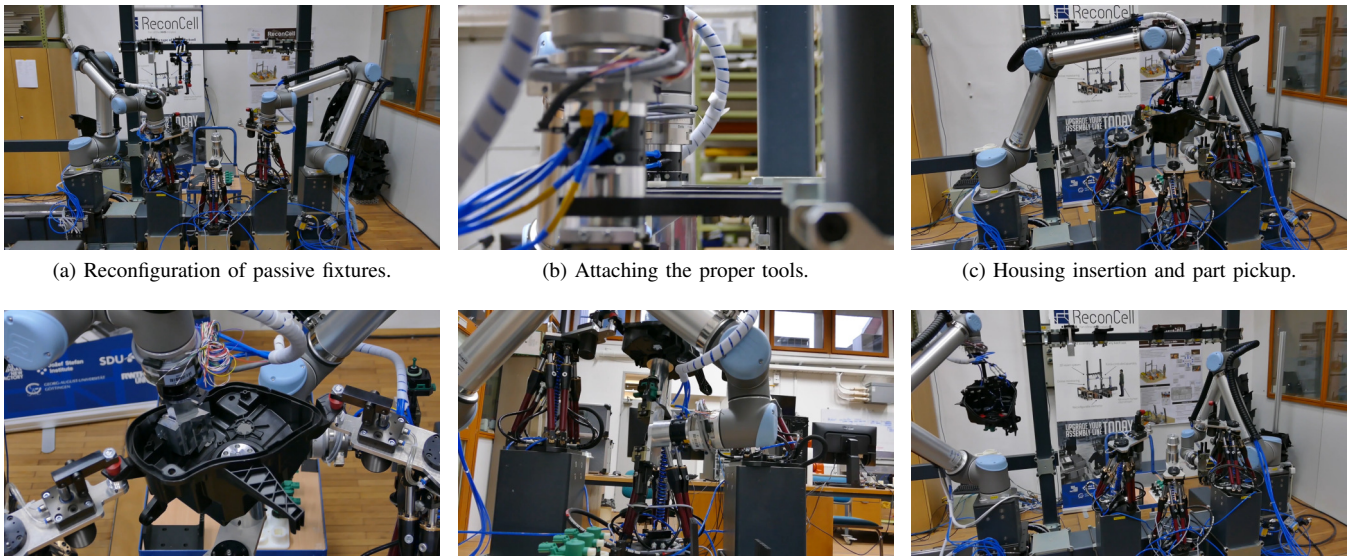
IV. Use Case Evaluation

As noted above, the manufacturing industry aims towards lower production costs and high efficiency assembly lines with high repetition, flexibility, and easy-to-use interfaces. Manual work and quality is highly dependent on workers' qualifications, skills and their knowledge of the assembly process. Customers expect that the supplier company is very flexible in coping with changes in demand. This is why SMEs seek to time every task carefully and look for optimizations.

The number of parts produced in a single company can vary substantially per project in a single year. In the industry of automotive light production, this variation is typically between 100,000-300,000 units per item. However, these lights are not assembled in one batch. With new orders it is often necessary to switch from one automotive light type to another. For the assembly of each light type it is necessary to reconfigure the workcell. This is where the technologies described in this paper become useful.

A. Description of the challenge

In the automotive light industry, each light requires its own unique assembly device, which is typically very large and cumbersome. When the subcontractor company stops producing the parts to match the regular demand, assembly devices must not be discarded because they are required such that the supplier company is able to produce spare parts for at least the next five years. This means that the assembly devices are stored at the company for those five years after the production has ceased. The suppliers therefore require



(a) Reconfiguration of passive fixtures. (b) Attaching the proper tools. (c) Housing insertion and part pickup.

(d) Robot 1 part assembly. (e) Robot 2 part assembly. (f) Finished product removal

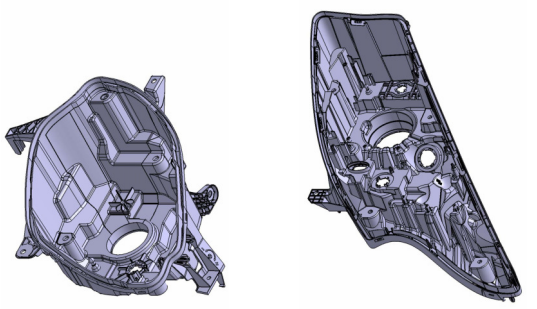
Fig. 6: Key parts of the assembly process for the Elvez company automotive light production use-case.

a significant amount of storage space just to house these assembly devices since, given that they produce many different types of parts, the assembly devices begin to accumulate.

Production of spare parts is a low quantity piece production scenario and usually occurs only a few times per year. This poses a problem because the large, cumbersome assembly devices must be swapped into the production process in order to cover the demand for the spare parts, an exercise that is highly inefficient given such low-quantity, low-frequency situations. It would therefore be extremely useful for suppliers to have a single robot cell available which is capable of assembling many different types of lights, while also being rapidly reconfigurable for alternating production scenarios.

Automotive lights (headlamps) are made up of typical structural elements such as housing, actuators, bulb holders, adjustable screws, heat shields, wires, etc. In our experiments we demonstrated that the developed reconfigurable robot workcell provides the much needed flexibility and fast set-up characteristics for automated assembly processes in the context of automotive lights. The key parts of the tested assembly process are seen in Fig. 6. By working together with the Elvez company, we were able to show that the proposed workcell can be automatically reconfigured for the successful assembly of different car headlamps, two of them shown in Fig. 7.

Before the start of the production of a new light housing model, the reconfigurable fixtures must be placed in the appropriate configuration such that they can accommodate the initial workpiece, which is the housing (main body of the headlamp) that comes directly from an injection molding machine. This step happens only once per production



(a) Light housing model X07. (b) Light housing model X82.

Fig. 7: Two different automotive headlamp housings that Elvez produces.

scenario and need not be repeated during production of a single type of headlamp (*c.f.* Fig. 6a). In the next step, one of the robots equips itself with a gripper (*c.f.* background of Fig. 6b) with which it will pick up the housing and insert it into the fixtures (*c.f.* Fig. 6c). During this time, the other robot equips itself with a double-headed gripper designed for picking up multiple parts that need to be inserted into the housing later on (*c.f.* foreground of Fig. 6b) and proceeds to pick up the relevant bulb holder and small motor parts one after another (*c.f.* right robot of Fig. 6c). Following this, pneumatic clamps mounted on the hexapods close to ensure proper part affixment. The robot holding the bulb holder and small motor parts proceeds to insert them into the housing one by one (*c.f.* Fig. 6e). At the same time, the other robot

swaps its tool with a magnetic tool necessary to pick up a metal heat shield and insert it into the housing, and proceeds to do just that (*c.f.* Fig. 6d). Finally, this same robot swaps its tool again with the housing gripper tool and after the pneumatic clamps have been opened, proceeds to remove the housing from the fixtures, before moving it on to the next step of the production process. After this, the cycle is complete and can be repeated as necessary.[1]

V. Conclusion and Future Work

In this paper we presented a new reconfigurable robot workcell that aims to assist the manufacturing industry with small production batches where shifts in demand occur frequently. The developed workcell is built of both reconfigurable hardware elements and modular software components. With respect to the hardware reconfigurability, we presented a unique combination of various technologies that allow for fast setup and reconfiguration. Affordable passive flexible fixtures are automatically reconfigured via robot manipulation. The software system architecture was built to be robot-independent. We developed a real-time robot control server that is responsible for low-level real-time trajectory generation. On top of the real-time server, we developed a ROS-based architecture for high-level control and communication. To demonstrate the benefit of using such a workcell in a real industrial scenario, a case study was developed in collaboration with a partner from the automotive industry. By applying the developed workcell to a real industry use-case, we demonstrated the applicability of the developed technologies.

In the future, we will focus on methods for easing the programming tasks so that it will become possible for non-experts to program the workcell by themselves. A special visual programming interface is being developed that will remove the need for a robotic expert for assembly task programming. This will allow companies to use their existing production experts to program assembly tasks.

Acknowledgment

This work has been funded by the Horizon 2020 ICT-FoF Innovation Action no 680431, ReconCell (A Reconfigurable robot workCell for fast set-up of automated assembly processes in SMEs) and partially funded by the GOSTOP programme, contract no C3330-16-529000, co-financed by Slovenia and the EU under the ERDF.

References

[1] N. Krüger, A. Ude, H. G. Petersen, B. Nemec, L.-P. Ellekilde, T. R. Savarimuthu, J. A. Rytz, K. Fischer, A. G. Buch, D. Kraft, W. Mustafa, E. E. Aksoy, J. Papon, A. Kramberger, and F. Wörgötter, "Technologies for the Fast Set-Up of Automated Assembly Processes," *KI - Künstliche Intelligenz*, vol. 28, pp. 305–313, Nov. 2014.

[1]This assembly process can be seen in its entirety in the video available here: `https://www.ijs.si/~aude/housing_assembly.mov`

[2] European Factories of the Future Research Association, "Factories of the Future: Multi-Annual Roadmap for the Contractual PPP under Horizon 2020," tech. rep., Publications office of the European Union: Brussels, Belgium, 2013.

[3] "ReconCell: A Reconfigurable robot workCell for fast set-up of automated assembly processes in SMEs." `http://www.reconcell.eu/`. Accessed: 2017-04-13.

[4] R. M. Setchi and N. Lagos, "Reconfigurability and reconfigurable manufacturing systems: state-of-the-art review," in *IEEE International Conference on Industrial Informatics (INDIN)*, pp. 529–535, 2004.

[5] Z. M. Bi, S. Y. T. Lang, M. Verner, and P. Orban, "Development of reconfigurable machines," *The International Journal of Advanced Manufacturing Technology*, vol. 39, pp. 1227–1251, Dec. 2008.

[6] M. Fulea, S. Popescu, E. Brad, B. Mocan, and M. Murar, "A literature survey on reconfigurable industrial robotic work cells," *Applied Mechanics and Materials*, vol. 762, p. 233, 2015.

[7] I.-M. Chen, *Theory and applications of modular reconfigurable robotic systems*. PhD thesis, California Institute of Technology, 1994.

[8] I.-M. Chen, "Rapid response manufacturing through a rapidly reconfigurable robotic workcell," *Robotics and Computer-Integrated Manufacturing*, vol. 17, no. 3, pp. 199–213, 2001.

[9] I.-M. Chen, "A Rapidly Reconfigurable Robotics Workcell and Its Applications for Tissue Engineering." `https://dspace.mit.edu/handle/1721.1/3753`, 2003.

[10] M. Gödl, I. Kovač, and A. Frank, "A robot-guided reconfigurable assembly system," in *Proceedings of the 3rd International CIRP Conference on Reconfigurable Manufacturing*, 2005.

[11] M. Jonsson and G. Ossbahr, "Aspects of reconfigurable and flexible fixtures," *Production Engineering Research and Development*, vol. 4, no. 4, pp. 333–339, 2010.

[12] K. Salminen and I. Kovač, "Role Based Self-Adaptation of Reconfigurable Robotized Systems for Sustainable Manufacturing," in *Proceedings of the 22nd International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2012)*, (Helsinki, Finland), June 2012.

[13] T. R. Savarimuthu, A. G. Buch, C. Schlette, N. Wantia, J. Rossmann, D. Martínez, G. Alenyá, C. Torras, A. Ude, B. Nemec, A. Kramberger, F. Wörgötter, E. E. Aksoy, J. Papon, S. Haller, J. Piater, and N. Krüger, "Teaching a Robot the Semantics of Assembly Tasks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017. doi: 10.1109/TSMC.2016.2635479.

[14] K. Collins, A. J. Palmer, and K. Rathmill, "The Development of a European Benchmark for the Comparison of Assembly Robot Programming Systems," in *Robot Technology and Applications*, pp. 187–199, Springer, Berlin, Heidelberg, 1985. DOI: 10.1007/978-3-662-02440-9_18.

[15] A. Ude, B. Nemec, T. Petrič, and J. Morimoto, "Orientation in cartesian space dynamic movement primitives," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Hong Kong), pp. 2997–3004, 2014.

[16] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, (Kobe, Japan), 2009.

[17] "BoxJoint Plates." `http://www.boxjoint.se/BoxJoint/BoxJoint_Plates.html`. Accessed: 2017-03-23.

[18] "Simulink Real-Time - Simulink - MATLAB & Simulink." `https://www.mathworks.com/products/simulink-real-time.html`. Accessed: 2017-03-16.

[19] "ROS 2." `https://github.com/ros2/ros2/wiki`. Accessed: 2017-04-12.

[20] "VEROSIM." `http://www.verosim-solutions.com`. Accessed: 2017-04-14.

[21] F. J. Abu-Dakka, B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krüger, and A. Ude, "Adaptation of manipulation skills in physical contact with the environment to reference force profiles," *Autonomous Robots*, vol. 39, no. 2, pp. 199–217, 2015.

[22] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," *Advances in Neural Information Processing Systems*, pp. 1523–1530, 2002.