Playback Robot Programming Framework for Fiber Spraying Processes

Edgar Schmidt and Dominik Henrich

Abstract

Robot-based automation is still not widespread in small and medium-sized enterprises, since programming industrial robots is usually costly and only feasible by experts. This disadvantages can be resolved by using intuitive robot programming approaches like playback programming. At the same time, there are currently not automatized automatized, like fiber spraying. We present a novel approach in programming a robot system for fiber spraying processes, which extends a playback programming framework inspired by video editing concepts. The resulting framework allows the programming of also the periphery devices needed for the fiber spraying process. We evaluated the resulting programming framework to measure the intuitiveness in the use and show that the framework is not only able to program fiber spraying tasks but is also rather intuitive to use for domain experts.

Keywords

Flexible manufacturing • Small batch production • Intuitive robot programming • Playback programming • Fiber spraying process

E. Schmidt (✉) · D. Henrich

Chair for Applied Computer Science III Robotics and Embedded Systems, Universität Bayreuth, 95440 Bayreuth, Germany

E-mail: Edgar.Schmidt@uni-bayreuth.de

URL: <http://robotics.uni-bayreuth.de>

D. Henrich

E-mail: Dominik.Henrich@uni-bayreuth.de

© The Author(s) 2022

T. Schüppstuhl et al. (eds.), *Annals of Scientific Society for Assembly, Handling and Industrial Robotics 2021*,

https://doi.org/10.1007/978-3-030-74032-0_29

1 Introduction

Robot-based automation is still not distributed in small and medium-sized enterprises (SMEs), even though a low cost and high quality production is possible with the use of robots [1]. This low spread is largely due to the small batch production, which is characterized by different phases of robot use with frequent reconfiguration of the robot system [2]. A fast reconfiguration can be achieved through a speed-up of the robot programming with the use of intuitive robot programming approaches [3, 4]. With the use of such techniques a robot system can quickly be programmed by experts in the domain of the application (*domain experts*), who have little to no programming expertise.

In the production of glass and carbon fiber composites, fiber spraying processes are one of the most commonly used production methods for short fiber-reinforced composites [5], but are only rarely automatized. However, a reduction of the production costs and an increase of the throughput is possible by means of automation of such processes.

We envision a robot programming framework for various kinds of fiber spraying processes, which can easily be programmed by experts in the domain of spraying processes who have little to no programming experience. For this, we need a framework with which we can program fiber spraying tasks and which is intuitive to use.

In this paper we present our programming framework for fiber spraying processes based on the automation approach presented in our work [6], which extends the framework presented in our work [3] for the use case fiber spraying processes. Section 2 gives an overview of the related work regarding the automation of fiber spraying processes and intuitive programming of robots. In Sect. 3, we describe our extensions to the playback programming framework for fiber spraying processes. In Sect. 4, we evaluate the intuitiveness of our framework with a user study. At the end, Sect. 5 summarizes and concludes the paper.

2 Related Work

Most of the commercially available fiber spraying systems are manually operated systems, so that they can be used like a spray paint gun. When used, these manually operated systems expose the domain expert to polluted air during the spraying, the quality of the produced composite depends heavily on the expertise of the operator while at the same time the composite is hardly reproducible and has a high error rate. An automation approach puts away these disadvantages, as seen in an off-line programming approach using graphical programming and virtual reality [7]. This off-line approach is only of limited suitability for small batch production in SMEs since off-line programming requires special training in robot programming.

In the wide variety of robot programming frameworks, only few are intuitive in use. One point is, that textual programming is only suitable for use with robotic experts [8], so that most of the frameworks of robotic manufacturers cannot be used by domain experts.

Commonly used industrial robot programming frameworks are usually based on a certain form of graphical representation for the user interface [9], but many of the frameworks only use one form of graphical representation like icon-based or wizard-based. There also exists frameworks which combine a robot programming paradigm with more than one type of graphical representation, as they utilize a data-flow representation, include a 3D simulation and provide icon based editing [3][10]. As several intuitive representation concepts are combined in these frameworks, they allow domain experts to program complex tasks easily.

The playback robot programming paradigm is of good suitability for an intuitive robot programming framework, as it can be used easily by domain experts who have no knowledge about programming itself. In playback robot programming, the robot is manually guided through the domain expert, and the trajectory is been recorded and stored by the programming system. After that, the trajectory can be played back. This approach is already used for different applications, like deburring [2], spray painting [11], welding [12], contour following [13], and also workpiece assembling with dual arms [14]. Since this approach exploits the process specific knowledge of the operator directly, it is suitable as robot programming paradigm for fiber spraying processes.

Summarized, there does not exist a robot programming framework for the programming of fiber spraying processes, which on the one side utilizes an intuitive robot programming paradigm like playback programming, and on the other side uses an easy to understand and intuitive to use graphical user interface.

3 Programming Framework for Fiber Spraying Processes

In the following subsections we will describe our approach. First, we explain our use case fiber spraying process, then we define important terms. After that, we explain the general structure of the programming framework and describe the user interface in general. At last, we give a closer insight into the programming of the periphery for the fiber spraying processes. Since this framework is based on previous work [3], we put a focus into the extension of the programming framework.

3.1 Fiber Spraying Process

In a *fiber spraying process* for the manufacturing of fiber composite ceramics, continuous oxidic fiber bundles (rovings) are chopped to uniform length immediately before being ejected out of the cutting unit and into a slurry spray, by which they are entrained. Due to the angle and the distance between the tool head (cutting unit and spray gun) and the mold surface the bundles are infiltrated during flight with the slurry. The infiltrated fiber bundles, with slurry being inside the fiber bundles as well as between the fiber bundles reach the mold surface and are orientated randomly (Fig. 1).

Fig. 1 Schematic illustration of a manual fiber spraying process

The most important requirement for this process is that the spray gun mounted on the robot toolhead should have a constant distance and orientation to the mold surface. This can be supported by the use of an external axis like a turntable, which allows the spray gun to reach every surface by rotating the mold. Since the spray pattern of the spray gun has also direct influence on the quality of the produced composite and the used fiber bundles are expensive, the material flow and the spray pattern has also to be programmable with high accuracy through the use of pressure regulators. In case of [6], five pressure values have to be programmed.

3.2 Definitions

Playback Programming, also called Walk-Through programming [12] or Programming by Guiding [15], describes a programming approach which consists of two phases: the programming phase and the playback phase. In the programming phase, the robot can be guided either kinesthetically or by a teach-pendant. While guiding, the playback programming framework records the robot joints motions and stores it. In the playback phase, the stored motions can be played back like it was recorded. We call the concatenation of motions in a playback program a *sequence*. A sequence of n -dimensional joint configurations with timestamps for each joint configuration, where n is the amount of joints of the robot, is further called a *trajectory*.

Besides the term *actor* in [3], we additional define the term *periphery device*: A periphery device is a special case of an actor, characterized in the programming framework by his own *periphery sequence*, which consists of device-specific commands. Examples for periphery devices used in our framework are a turntable which acts as a external axis and pressure regulators which define the material flow and the spray pattern.

A command in the periphery sequence of a turntable is called a *turntable motion*. These motions are specified by a duration, given in seconds [s] and a rotatory distance in degree [°] in the rotatory axis of the turntable. A command in the periphery sequence of the pressure

regulators is called a *pressure configuration*. A pressure configuration specifies a duration, given in seconds [s], and five pressure values in bar [bar]. The flat and round jet are defining the spray pattern of the spray gun, while the slurry value defines the amount of slurry given into the process. The pilot air defines the de-/activation of the actual material flow through the spray gun and the cutter value defines the cutting unit speed.

3.3 General Structure

The programming framework is designed as a client-server application using TCP/IP communication, whereby the graphical user interface acts as the server and the robot act as a client. When a client tries to connect to the server, a registration dialog will pop up. If the periphery devices are controlled centralized through the controller of the robot, the periphery is registered together with the robot. The framework allows to register more than one robot with periphery devices at the same time and the server application can be used via a tablet or a personal computer.

After registration of the robot, the programming of a task can take place immediately. The user first records a robot sequence or loads an already existing sequence from the storage. This sequence can be reviewed or edited at any time before its execution. The periphery can also be programmed right after the registration, but is usually programmed after a trajectory is available. When the programming of the fiber spraying task is finished and the playback phase is started, the sequences are processed into machine code by an interpreter. This machine code is sent to the robot controller and is executed for playback.

3.4 User Interface

The main part of the interaction takes place through the graphical user interface. The design is based on guidelines for the design of operating concepts from the ISO-9241 standard, and the work of [16, 17]. The user interface is further designed to be operated via a touch screen.

The user interface is divided into three parts: The control bar at the top, the timeline area at the bottom left, which shows the data-flow of our current program, and the simulation window at the bottom right. Furthermore, it has an edit mode and a playback mode, which separate both phases of the playback programming paradigm: The edit mode is used for the record phase and the playback mode is used to playback. The different modes with their interfaces are shown in Fig. 2.

The control bar acts as a control area, which contains all functions to control the different actors. These functions are divided into three groups in edit mode. The first group are file operations, the second and third group are extensions to the playback programming paradigm like editing of sequences and synchronizing between actors. In play mode, functions for start, pause and stop are provided.

Fig. 2 The graphical user interface of our programming framework, showing the topbar, timelines and 3D-simulation in the edit mode where the actual programming takes place (left) and the playback mode, where the execution of the program is supervised (right)

The timeline area consists of label areas and graphical representations of sequences. In a label area the name of the actor is shown and additional control buttons related to the actor are positioned, like the de-/activation of the hand guidance and the recording of a robot or the programming menu in case of periphery devices. The graphical representation of a sequence is shown as a velocity graph for a robot or turntable. The turntable timeline includes an additional indicator for the direction of the rotation. For the pressure regulators, the pressure values in [bar] are shown. The start and the end of the pressure configurations is indicated, as it hints the material flow of the program. All timelines can be folded resulting in a space-saving but less informative representation, so that a better overview over the registered actors is given but only binary information is displayed.

In the simulation window, a model of the robot is shown with the exhaust unit and the turntable. Further, a 3D representation of a trajectory is also shown if one is available. The timelines and the simulation window are connected so that the simulation window is showing the configuration of the robot when selecting a part of the timeline. During playback mode, the simulation window shows the current position of the physical actor. Below the simulation window, statistical values are displayed, which show the duration of the whole program and the material consumption of slurry and fiber of the program.

3.5 Programming of Periphery Devices

By clicking on the programming button of a periphery device, a window replacing the simulation window is shown. This programming window is designed based on the definition of single commands, which are selected with a combo box. The periphery sequence is shown as a preview and the current command is highlighted in color. The different input parameter can be set through spin boxes. Additionally, timestamps can be defined by selecting from the timeline. Already programmed commands can be edited or deleted. When a command is added to the periphery sequence, an empty run until the next configuration is added automatically, so that the sequence is well defined over the whole program time.

Fig. 3 Two periphery devices have to be programmed in a fiber spraying process. While the turntable acts as external axis (left), the pressure regulators control the actual material flow in the program

For the programming of a turntable motion a start and an end timestamp has to be defined, from which the duration of the motion is calculated. With the input of the moved distance for this time period the motion is fully defined. The corresponding velocity will be calculated from the input and is shown in the timeline. For the use of large scale motions like constant coverage of a cylindrical mold, the adjustment of a velocity for a motion and the setting of a endless running mode of the turntable is achieved with the help of an expert menu. The programming window with the timeline is shown in Fig. 3 left.

Besides the definition of timestamps, the pressure configuration programming takes place through the input of the round jet, flat jet and the slurry. The resulting spray pattern is visualized below the input fields and provides additional information about the expected spray pattern when using the defined configuration. The pilot air and the cutter value can be set in a separate expert menu, as their values only differ for special use cases in the given process. Also, the delay time between the activation of the pressures can be set in the expert menu, which is mostly only needed when the spray gun is changed. The base timeline shows a green bar when every component is running, indicating that the pressure regulator system is spraying at this time interval. In contrast to the aforementioned indicators in the timeline, the green bar indicates that all components are ready to spray. The input is converted into single configurations for the activation of every pressure value, so that actually five pressure configurations are inserted into the sequence when the input is processed. The programming window with timeline is shown in the right subfigure of Fig. 3.

4 Experiments

In this section we will first explain our experiment setup in general. After this, we describe the outcomes of our user study in which we try to measure the intuitiveness of our system when programming fiber spraying tasks.

4.1 Experiment Setup

The participants of this study were 10 persons from the research field of robotics and material engineering, divided in two groups of same size. The average age of the participants was 26.9 years, with 24 years the youngest and 33 years the oldest participant. One half of the participants has expertise in programming and one half has little to no experience in programming. Three of the participants have experience with fiber spraying processes and can be called domain experts. All of the domain experts have also little to no programming knowledge.

As hardware setup for our experiments we use the cell proposed in [6], which utilizes a six degree of freedom robot. Further we use a tablet that provides touch input as input device. The turntable is mounted in the center of the cell providing a rotation around the z-Axis of the robot. The pressure regulators are connected via EtherCAT with the robot control while the turntable is commanded through serial port. A smooth hand guidance is possible through the use of a hybrid force controller, which combines the advantage of a P-controller in fine movement with the advantage of a N-controller for large-scale motions [18].

4.2 Evaluation of Intuitiveness

We rated the intuitiveness in the use of our programming framework with the MINERIC toolkit [19]. For this, the participants had to program a fiber spraying task with the goal to produce a plate. We separated the task into three sub tasks. Task 1 was to program an initial run, which was a meander trajectory there and back over the mold. Task 2 was to edit the trajectory from task 1 so that the robot executes two runs over the mold. Task 3 was to program the periphery devices so that the mold rotated between the runs and for every run other pressures configurations had to be set. Each participant was introduced in the interface before a task was explained. Example solutions are shown in Fig. 4.

Fig. 4 Possible solutions for the sub tasks of the fiber spraying task which was used to measure the intuitiveness of our framework

Fig. 5 Outcomes of the user study. The left chart shows the results of the effectivity. Higher values are better. The middle chart shows the results of the efficiency. Lower values are better. The right chart shows the QUESI scores and subscores representing the satisfaction. Higher values are better

The effectivity score is shown in the left chart of Fig. 5. It is measured in percent where a higher score indicates an effective use. The overall rating is 80%, which indicates a rather effective framework to use. Furthermore, the subscores show that task 2 was very effective in use, but the programming of the periphery could be more effective. The efficiency score is shown in the middle chart of Fig. 5. Its value ranges from 0 to 220, where lower scores indicate a better efficiency. The overall rating of 55.83 is translatable to little to some effort when using the framework. The increase of the efficiency for task 3 may be explained with the fact that the periphery programming is harder to understand, while the editing is easy to understand. The QUESI score and its subscores are shown in the right chart of Fig. 5. The QUESI score is measured with five subscores on a scale from 1 to 5, where 5 means that the participants are fully satisfied with the framework and 1 means that they are not satisfied [20]. An overall QUESI score of 3.42 shows that the participants were rather satisfied, but there can be further improvements. The subscores are *subjective mental workload* (W), *perceived achievement of goals* (G), *perceived effort of learning* (L), *familiarity* (F), and *perceived error rate* (E). The score of 2.7 for perceived error rate indicates, that in the process of using the framework some errors were made. Compared to our work in [17], the use of the framework received good ratings although the framework provides now more functionalities and is used to program a more difficult application.

As of the remarkable difference in effectivity and efficiency in task 3, we discuss our observations while the participants were completing this task. Although the current periphery configuration was highlighted, a previous programmed configuration was often mistakenly edited instead of a new one and so the programming was corrupted. Due to the lack of an Undo button, this error could not be corrected by the user and he/she had to start this task again. Also, the option to transfer timeline ranges from selection was used delayed or not at all, resulting in a strenuous text input. These facts caused that task 3 was overall more

strenuous for the user than the other two, resulting in a not so effective use. We assume further, that these problems frustrated the user and lead to a also not so well rating in the efficiency.

5 Conclusion and Future Work

In this paper, we presented a framework for the programming of fiber spraying processes, which utilizes a framework inspired by video concepts. We described the new extensions to the framework, with which a fiber spraying task can be programmed, especially periphery devices in form of a turntable and pressure regulator. As of the outcomes of the user study, the framework is rather intuitive to use but the periphery programming has to be further improved with the the observations made in the user study.

Since our framework is only capable of editing parts of a trajectory in their temporal course, a new trajectory has to be programmed by hand guidance every time when only minor details in the spatial course of the sequence have to be changed. This disadvantage could be compensated with the opportunity to edit the spatial course of a trajectory manually or automatically. Also, a CAD model of the mold could be integrated in the simulation window and this model could be used for further optimization in the programming process, like giving feedback when the mold is actual been splashed or overspray is detected.

Acknowledgements This work has partly been supported by the European Regional Development Fund (ERDF) under the project “Roadmap zur flexiblen Produktion individueller Produkte” (Roadmap flexPro). The authors would like to thank Pascal Ruppert and David Harrer for their valuable help in the implementation of the prototype system.

References

1. Perzylo A., et al.: SMERobotics Smart Robots for Flexible Manufacturing. In: IEEE Robotics and Automation Magazine. pp. 78–90 (2019)
2. Dietz T., Schneider, U., Barho, M., Oberer-Treitz, S., Drust, M., Hollmann, R., Hägele, M.: Programming System for Efficient Use of Industrial Robots for Deburring in SME Environments. In: ROBOTIK 2012, VDE, 1–6 (2012)
3. Riedl M., Henrich D.: A Fast Robot Playback Programming System Using Video Editing Concepts. In: Tagungsband des 4. Kongresses Montage Handhabung Industrieroboter, Springer Vieweg, 259–268 (2019)
4. Schraft R. D., Meyer C.: The need for an intuitive teaching method for small and medium enterprises. In: VDI BERICHTE (2006)
5. Witte, E., Mathes, W., Sauer, M., Kühnel, M.: Composites-Marktbericht 2018: Marktentwicklungen. Ausblicke und Herausforderungen. Carbon Composites e.V. und Industrievereinigung Verstärkte Kunststoffe e.V, Trends (2018)

6. Schmidt E., Winkelbauer J., Puchas G., Henrich D., Krenkel W.: Robot-Based Fiber Spray Process for Small Batch Production. In: Annals of Scientific Society for Assembly, Handling and Industrial Robotics (2020)
7. Abhoff C.: Herstellung von Tailored Composites. Fraunhofer WKI, (2016). https://www.wki.fraunhofer.de/content/dam/wki/de/documents/Mediathek/themen/hofzet/HOFZET_Faserspritzen_2016-10.pdf
8. Orendt E. M., Riedl M., Henrich D.: Programmierung von Robotern. In: Handbuch Mensch-Roboter-Kollaboration, 243-270 (2019)
9. Rossano G. F., Martinez. C., Hedelind M., Murphy S., Fuhlbrigge T. A.: Easy robot programming concepts: An industrial perspective. In: IEEE International Conference on Automation Science and Engineering (CASE), 1119-1126 (2013)
10. TracePen DE [online]: <https://wandelbots.com/de/>, last access: February 1st 2021
11. Ferraguti F., Landi C. T., Secchi C., Fantuzzi C., Nolli M., Pesamosca M.: Walkthrough programming for industrial applications. In: 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 31-38 (2017)
12. Ang Jr. M. H., Lin W., Lim S. Y.: A walk-through programmed robot for welding in shipyards. In: Industrial Robot: An International Journal, 377-388 (1999)
13. Bascetta L., Ferretti G., Magnani G., Rocco P.: Walk-through programming for robotic manipulators based on admittance control. In: Robotica, 31(7) (2013)
14. Park, C., Park, K., Park D., Kyung J.-H.: Dual Arm Manipulator and Its Easy Teaching System. In: Proceedings of 2009 IEEE International Symposium on Assembly and Manufacturing, 242-247 (2009)
15. Lozano-Pérez, T.: Robot Programming. Proceedings of the IEEE **71**(7), 821-841 (1983)
16. Kraft M., Rickert M.: How to teach your robot in 5 minutes: Applying UX paradigms to human-robot-interaction. 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 942-949 (2017)
17. Colceriu C., Riedl M., Henrich D., Brell-Cokcan S., Nitsch V.: User-Centered Design of an Intuitive Robot Playback Programming System. Annals of Scientific Society for Assembly, Handling and Industrial Robotics, 193-203 (2020)
18. Stolka, P., Henrich, D.: A hybrid force following controller for multi-scale motions. IFAC Proceedings Volumes **247-252**, (2003)
19. Orendt E. M., Fichtner M., Henrich D.: MINERIC toolkit: Measuring instruments to evaluate robustness and intuitiveness of robot programming concepts. 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 1379-1386 (2017)
20. Naumann A., Hurtienne J.: Benchmarks for intuitive interaction with mobile devices. Proceedings of the 12th international conference on Human computer interaction with mobile devices and services (MobileHCI '10), 401-402 (2010)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.