Modeling, Identification and Control, Vol. 44, No. 4, 2023, pp. 141–154, ISSN 1890–1328

Programming Fine Manufacturing Tasks on Collaborative Robots: A Case Study on Industrial Gluing

D. Schäle M. F. Stoelen E. Kyrkjebø

HVL Robotics, Western Norway University of Applied Sciences, N-6812 Førde, Norway. E-mail: dasc@hvl.no

Abstract

This paper explores programming of fine manufacturing tasks using collaborative robots. We conduct a case study based on an industrial gluing task, comparing two programming approaches: Learning from Demonstration (LfD) and Computer-Aided Manufacturing (CAM). We investigate the suitability of these approaches for ad-hoc automation of fine manufacturing tasks by expert operators, and discuss the strengths and weaknesses associated with their usage. The case study reveals that there are benefits and limitations to both approaches. The CAM-based approach provides a precise path for execution without the need for robot programming expertise, but is strongly dependent on the quality of the gauging process. The LfD approach is intuitive and quick to set up, but is strongly dependent on the quality of the demonstrations. Our findings suggest that there is a potential for a hybrid solution combining the best of both approaches in a unified interface, and provide a foundation for future research on hybrid programming interfaces for fine manufacturing tasks using collaborative robots.

Keywords: Cobots, Manufacturing, Robot programming, CAM, Learning from Demonstration

1 Introduction

Collaborative robots (Cobots) are a promising alternative to conventional industrial robots for small- and medium-sized enterprises (SME's). Their built-in safety features allow them to work together or in physical proximity of human operators. The ability to make physical contact with human operators opens up new programming interfaces for these robots. At the same time, the programming interfaces for cobots require higher levels of intuitiveness and efficiency compared to conventional industrial robots. In cobot settings, it is the operator on the factory floor who is responsible for programming the robots, unlike conventional industrial robots where this task falls to robot experts.

A common way of programming cobots today is to move the robot manually into key-poses (waypoints), store them and connect them with different movement commands on the robot's teach pendant - a process known as walk-through programming (Villani et al., 2018). A second approach used predominantly for programming traditional industrial robots, but also applicable to cobots, is offline programming (OLP) (Heimann and Guhl, 2020). OLP allows for precise motion planning, although it may disrupt the collaboration between robots and operators and requires formal education in programming and robotics. A more recent approach in the rise of robot learning methods is learning from demonstration (LfD). In LfD, the operator provides multiple examples of the full trajectory through kinesthetic teaching or teleoperation (Siciliano and Khatib, 2016, Ch. 74). The information demonstrated by the operator to the robot encompasses the complete implementation of the movement, including its kinematic and dynamic properties, as well as auxil-

doi:10.4173/mic.2023.4.1

iary sensor signals. The aim is to capture the nuances of the expert's task performance and transfer them to the robot.

The walk-through programming and especially LFD enable intuitive and fast programming of robots by workers without a strong robotics background, which enables workers to manifest their expertise in manufacturing in a robot program, without the need for a system integrator in between. However, these two approaches have limitations when it comes to programming fine manufacturing tasks that involve complex geometries and which approach the upper limit of the accuracy cobots typically can deliver; tasks, that conventionally would be described with fine-grained toolpaths or machine code. Approaching such tasks with lead-through programming would require a very high number of waypoints, resulting in cluttered robot programs. Changes in the task often result in tedious reprogramming, limiting the flexibility of the production. The LfD approach circumvents the problem of defining waypoints, but it can be difficult to achieve the desired accuracy by demonstration. And if all dimensions of the task are supposed to be programmed by demonstration, not just the Cartesian poses of e.g. a sanding machine, but also its rate of rotation, then the tools/hardware need to be prepared with sensors to record these parameter values during the demonstrations.

In this paper, we present and discuss two contrasting programming approaches for automating fine manufacturing tasks for a case study on industrial gluing with a Franka Emika Panda collaborative robot. We want to showcase their potential strengths and shortcomings in the light of programming precise manufacturing tasks on cobots by factory workers. Our goal is to identify a potential gap in programming precise manufacturing tasks on cobots, and to point to future directions of research towards interfaces that enable operators to program such tasks on the factory floor.

The first programming approach is based on computer aided manufacturing (CAM), and is in line with the common practice in robot machining of generating tool paths from CAD models. The second approach is based on LfD, common to physical human-robot interaction, and enables the operator to generate a tool path based on their demonstration of the task. The rationale behind selecting these approaches, which we view as two opposing ends of a spectrum of programming interfaces, is to explore whether they are principally suited for programming fine manufacturing tasks on cobots, and to facilitate the identification of potential intermediate solutions that can leverage the advantages offered by both ends of the spectrum.

The case study we consider in this paper is based

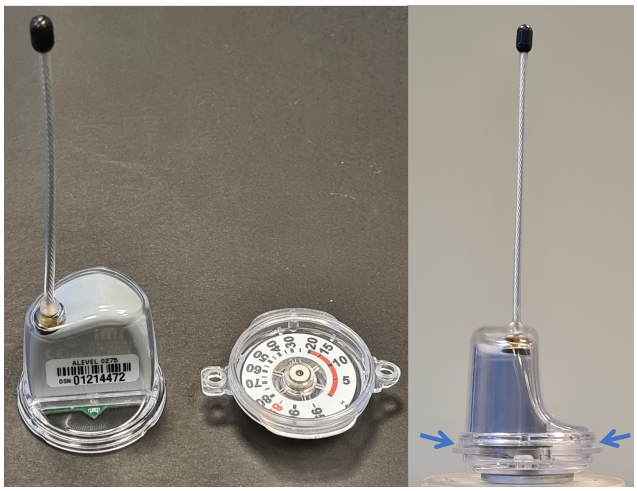


Figure 1: Case study task: Gluing operation in the assembly of an industrial sensor housing. The **blue arrows** indicate the groove where glue is applied.

on a real-world manufacturing task from an industrial research partner. The partner company has a long-term vision to automate production lines by augmenting the workforce with flexible robot fleets made up of cobots mounted to autonomous guided vehicles (AGVs). These Cobot-AGVs are designed to move around the factory floor, docking to different workstations, dynamically support human coworkers, or take over and complete certain tasks autonomously. One of these workstations is dedicated to a precise gluing operation in the assembly of an industrial sensor housing. The sensor housing consists of a top and a bottom part, which when put together, leave a 1×1 mm groove along the sensor's circumference (Fig. 1). The sensor has a diameter of 40mm.

The groove must be filled with an UV curable adhesive to hold top and bottom part together and to seal the sensor. At present, the adhesive is applied manually with a glue gun equipped with a dosing needle. In this process, the workers put the sensor on a rotating fixture and spin the sensor around slowly with one hand while applying the adhesive with the glue gun in the other hand (Fig. 2). The sensor is then removed from the fixture and cured in an UV chamber.

The company's goal is to automate the gluing operation with cobots in a flexible manufacturing process where the factory workers can quickly create gluing programs for new sensor shapes. Thus, this task makes a good example to investigate the challenges of programming manufacturing operations that require conventional robot precision, but also an accessible programming interface in a cobot-centric setting.

The control parameters that must be programmed

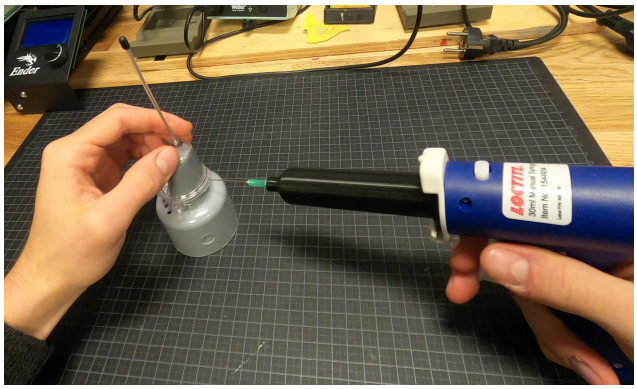


Figure 2: Case study task: Manual gluing operation similar to the process used in the partner company today.

in this gluing task are the relative motion between the glue gun mounted to the robot flange and the sensor, and the rate of glue flow. The relative motion between sensor and glue gun is comprised of the 1 degree of freedom (DOF) rotation of the sensor and the 6DOF movement of the glue gun in Cartesian space. These movements are comprised of a geometric path, defining the spatial properties, and time, which together define the dynamic properties of the movements. The dynamic properties of the movement are correlated with the glue flow rate and need to be synchronized. Within limits, depending on the viscoelastic properties of the glue, the same amount of glue per unit length along the path can be dispensed by moving the glue gun faster with a higher glue flow rate, or by moving slower with a lower flow rate. The desired amount of glue per unit length depends on the design of the sensor, e.g. the volume of the cavity at an instance along the gluing path. In the case study in this paper, we focus on programming the geometric path, and only briefly discuss the dynamic properties and glue flow; which we leave for future work.

2 Related Work

Robot programming can be separated into online methods such as lead-through programming, and offline methods which utilize software tools to generate programs without the need for direct involvement of the physical robot (Villani et al., 2018). Offline methods minimize downtime of the robot, but usually require advanced knowledge in 3D modelling and programming. For online methods, researchers have been studying various input modalities such as kinesthetic teaching, hand-gestures and vocal commands; often with the goal to make the programming as intuitive as possible for non-experts.

The factors that constitute efficient and intuitive programming are diverse and dynamic, and will change from operator to operator and from task to task. Ideally, the operator should be presented with a modular interface, where it is possible to switch between modalities and programming tools on the go to utilize their specific advantages for the work at hand.

In the following, we present different approaches to programming manufacturing tasks to give an overview of the wide range of interfaces used in the literature, and to provide a backdrop for our case study. We start with looking at online approaches to programming with human-like interfaces, continue along the spectrum with some interesting hybrid approaches, and end with CAD-based offline programming.

In Halim et al. (2022), a multimodal interface for no-code programming of cobots is presented. The operator can teach poses with hand and finger gestures captured with a camera system, as well as give certain voice commands to the robot which trigger actions in a finite state machine. They implemented a graphical user interface which provides feedback to the user and serves as a backup if the speech input fails e.g. due to background noise.

In Nemec et al. (2018), a LfD framework is presented that is focused on kinesthetic teaching and relies on the physical interaction between operator and robot as the main programming interface. The authors combine Dynamic Movement Primitives (DMPs) with a controller that allows for variable stiffness in the tangential and normal directions of the demonstrated path. The stiffness is coupled to spatial variability in the demonstrations and the speed of motion during the execution of the trajectory by the robot. This enables the operator to tackle difficult demonstrations by teaching the spatial and temporal components of the trajectory separately, and refining the previously learned trajectory through physically intervention during the execution by the robot.

In Iturrate et al. (2021), a LfD-based framework is proposed for programming and executing industrial gluing tasks, e.g. applying glue to PCBs. In contrast to Nemec et al. (2018), this approach uses only a single demonstration of a given task and instead of incremental refinement of the trajectory during execution, Iturrate et al. (2021) support the operator during kinesthetic teaching with an admittance controller with time-varying damping. To assist the operator in demonstrating a task, the authors estimate the surface normal of the workpiece to adjust the damping during kinesthetic teaching such as to maintain a higher

damping in the direction constrained by the workpiece and lower damping in the non-constrained directions. The surface normal is recorded and encoded with a mixture of radial basis functions. During reproduction of the task the robot runs a hybrid force-position controller. The force in direction of the surface normal is treated as process parameter and is manually set by the operator in a software interface. The authors argue that setting the contact force manually is more practical and accurate than trying to teach all aspects of the task through demonstration.

Hein et al. (2008) proposed a hybrid interface, utilizing ideas from offline programming to support the operator during online programming. The operator can control the robot with a camera-tracked, hand-held input device. In contrast to conventional tele-operation, the operator is supported through assisting algorithms informed by a CAD model of the workcell while driving the robot. During tele-operation, the CAD model is used e.g. for collision avoidance, adapting jogging speed, and dividing the workspace into different subspaces which affect the movement characteristics of the robot. The CAD model can also be used for automatic path planning which only require a start and end pose. Using this approach, the operator can create robot programs which consist of both automatically and manually taught motions.

Fossdal et al. (2021) present an approach for machine control from a CAD environment. The idea behind their approach is to shorten the feedback loop between user input in software and machine output/performance, such that the operator can quickly gain understanding on how the machine/robot actions their intent, and thus make for a steep learning curve of how to specify their intent in software to achieve the desired result on the machine/robot. For this, the system must enable swift experimentation with machine/robot parameters and communicate clearly the state, scope and limitations of the machine/robot. The authors argue that such a system has potential to benefit operators who are proficient in CAD but not in machine/robot programming.

While some of the authors in the papers above identified kinesthetic teaching and LfD as a fast and intuitive way for lay users to program manufacturing tasks on cobots, Neto and Mendes (2013) argue in equal measure for their CAD-based workflow for offline programming of industrial robots. The authors argue that CAD drawings are a useful tool for robot programming since humans are used to use drawings for explaining complex tasks to fellows, making CAD drawings an intuitive abstraction of robot programs. The OLP approach in Neto and Mendes (2013) generates robot programs directly from CAD drawings in a common CAD software (Autodesk Inventor) and does not require any additional commercial OLP (e.g. RoboDK) or CAM software. The starting point of the workflow is a CAD assembly model of the workpiece and the robot cell. To define a robot tool path, the operator either places end effector/tool models in the assembly which represent the desired tool pose in each segment of the path, or draws lines in the assembly which represent the positional data of the toolpath and then again places end effector/tool models to define the orientation in each segment of the path. Potential collisions must be anticipated by the user while creating the toolpath. After defining the toolpath in CAD, the actual robot tool path is generated through a custom software interface. As usual for OLP, the accuracy of the operation depends on the accuracy of the robot. Absolute positioning accuracy with a robot is not always easy to achieve and typically requires effortful calibration procedures. Furthermore, if objects in the modelled work cell have erroneous transformations to the common reference frame (e.g. robot base frame) that do not correspond precisely with the real world set-up, the operation is not performed accurately. Determining the transformations of objects to the reference frame in the real set-up can be costly. Depending on the required accuracy, an advanced external measurement system or probing with the robot is reported to be necessary.

The digest of literature above gives an impression of the many approaches to programming manufacturing tasks. The user interfaces and the modalities the authors choose to transfer instructions from operator to robot are substantially different. While some of the design choices can be explained in that the approaches are catering to different user groups or applications, we find it interesting that all of the authors consider their approach as intuitive and/or convenient. The question of what constitutes intuitive programming in the context of automating fine manufacturing tasks is therefore still very much an open question.

3 Case Study on Industrial Gluing

In this section, we first describe the experimental setup of the case study on programming fine manufacturing tasks based on the cooperative gluing scenario of our industrial partner described in Sec. 1. We then go into detail on how we implemented this task with a CAM-based approach, and how a gauging procedure is necessary when using a CAM-based approach to programming. Subsequently, we detail how we implemented the gluing task with a LfD-based approach. Our description of the two approaches has an emphasis on the different steps and software tools needed to complete

the programming of the gluing task. Lastly, we summarize the hardware and software implementation of the experimental setup for a comprehensive overview.

3.1 Experimental Setup

The experimental setup in our case study is inspired by the setup the workers use during the manual gluing operation (Fig. 3). The workpiece in the case study is a slightly modified mock-up of the industrial sensor; featuring an elliptic footprint ($d_1 = 40mm, d_2 = 30mm$) and a groove that runs along the circumference in variable heights. The cross section of the groove on the mock-up is similar to the one on the sensor (1×1mm). The design of this workpiece enforces movements of the glue gun tip along at least two axes, and thus makes the comparison between hand-guided toolpaths and computer generated toolpaths more general. The workpiece is placed on a rotary axis which is actuated by a Dynamixel MX-106T servo motor. In order to rotate the servo motor during manual demonstration of the task, the servo can be driven with a rotary encoder. Using the rotary encoder to rotate the workpiece has the advantage that the human demonstrator does not have to overcome the high gearbox friction and inertia of the servo motor that would be present when turning it manually. With this external axis and the glue gun mounted to the robot flange, the gluing paths can be reproduced in a similar way as done by the factory workers. However, rotating the workpiece with the servo motor is not merely emulating the human workflow, but is necessary to circle around the workpiece with a desired orientation of the glue gun. The Franka Emika Panda robot alone cannot orbit the workpiece with arbitrary end effector orientations (when mounted to a table top) since its 7th joint cannot rotate a full 360 degrees. A cobot without this restriction of its last joint, the Universal Robots UR5e, was considered as an alternative for the case study. However, the UR5e has 6 joints compared to the 7 joints of the Panda. The lack of kinematic redundancy, which is clearly noticeable during fine kinesthetic teaching on the UR5e, was deemed as a greater limitation and hence the Panda was chosen for the case study. Further details on the hardware integration of the setup can be found in Sec. 3.5.

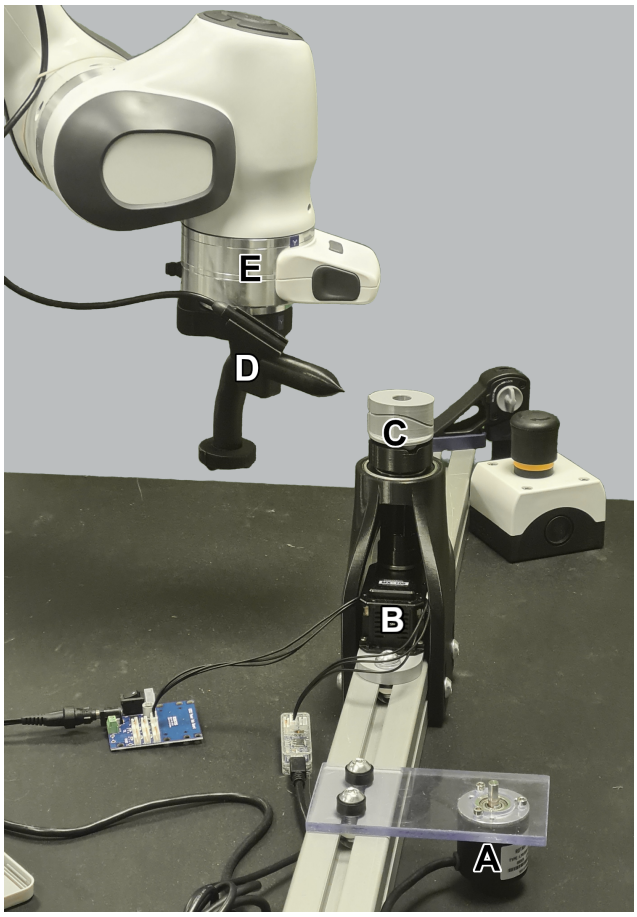


Figure 3: Case study: Setup. **A** Rotary encoder to drive the servo motor during demonstration. **B** Servo motor for rotating the workpiece. **C** Workpiece/Sensor mock-up. **D** Glue gun mock-up. **E** Robot flange.

3.2 CAM-based Programming Approach

This programming approach has much in common with conventional off-line programming of industrial robot manipulators and the programming of CNC machines. The idea is to create a controlled environment that can be modelled with sufficient precision, and then rely on the model and planning algorithms to generate toolpaths for the machine or robot. Just as in CNC machining, we use a CAD model of our workpiece to predefine where and how the tool is supposed to move along the part. This process of generating machine instructions based on geometries of a CAD model is commonly referred to as CAM. Professional CAD software often offers integrated CAM environments; we use the manufacturing environment integrated in AutoDesk Fusion 360. This CAM environment offers various machining operations for standard manufacturing techniques such as laser/water jet cutting, additive manufacturing, turning and milling along three to five axes.

Fusion 360 does not offer gluing as a manufacturing technique, but the "Multi-Axis Contour" is a 5 axis milling operation that follows 3D curves lying on the surface of a CAD model while keeping normal to

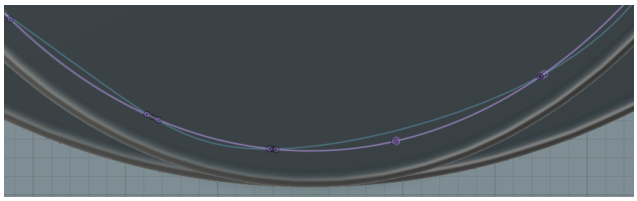
the model surface, and can be adjusted to accommodate manipulating a glue gun with a robot arm. Fully defining the movement of the robot's end effector in Cartesian space requires a toolpath with 6 instead of 5 degrees of freedom. With Multi-Axis Contour, the rotation about the tool axis remains undefined, since this is irrelevant for milling with tools that are rotational symmetrical around the tool axis. In our case, we have to define the rotation about the tool axis in a customized post-processing step in Matlab.

The curve which is traced by the operation can either be an existing feature on the model surface, or can be created by drawing a spline onto the surface. When drawing the spline onto the model, it is important that the spline lays exactly on the model surface at all points, otherwise the toolpath will have gaps during sections where the spline lifts off of the surface. We ensure that the spline we draw lays on the surface by first including the relevant surfaces of the model as 3D geometries into the sketch, drawing the spline onto the included surfaces, and finally projecting the spline onto the model surface. Projecting the spline onto the surface is necessary, since even when placing the spline anchors on the included surfaces, the sections between the anchors will deviate from the surface if the curvature of spline and the surface do not match (Fig. 4).

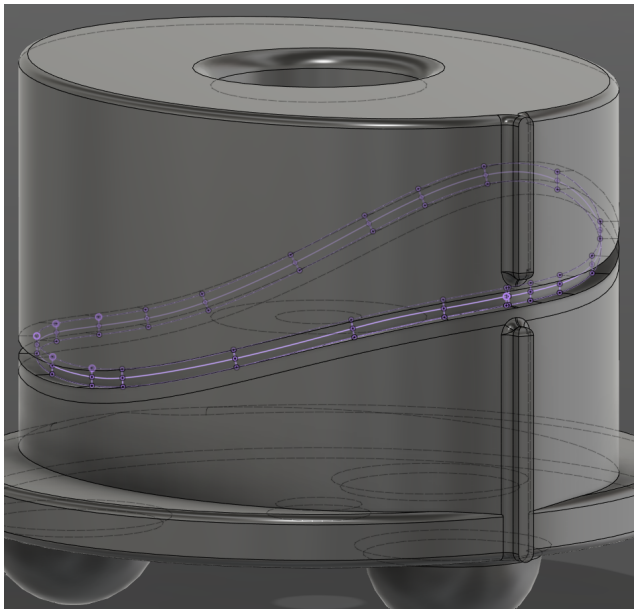
The available tools in Multi-Axis Contour are milling cutters, but we can modify the dimensions of a generic ball end mill to match the dosing needle on the glue gun. Some of the remaining settings in Multi-Axis Contour are only relevant when milling (e.g. feed & speed). For our case study, we adjust the tolerance to control how fine-grained the toolpath is going to be. We also adjust the entry position and the lead-in and -out movements to define where and how the tool approaches the part. After computing the Multi-Axis Contour operation, the resulting toolpath can be previewed within Fusion 360 as shown in Fig. 5.

The toolpath now embodies our desired operation, but it is still a generic set of instructions within Fusion 360 that can't be used to control a specific piece of hardware. To convert the toolpath to machine code that can be interpreted by the controller of the machine or robot at hand, an additional post-processing step is required. The post-processors allow to make hardware specific settings, convert the generic toolpath into machine-readable code and output it in the correct file format.

Fusion 360 comes with a library of post-processors for some common machines and some robots. However, there is no post-processor for the Franka Emika Panda robot we use, and in any case, a standard post-processor for the Panda would not work in our custom setup with an external axis controlled by a servo motor.



(a) Spline (blue line) deviating from the model surface included in the sketch (purple line).



(b) Final result after projecting the spline (purple) back onto the model surface.

Figure 4: Drawing a spline as reference for the "Multi-Axis Contour" operation.

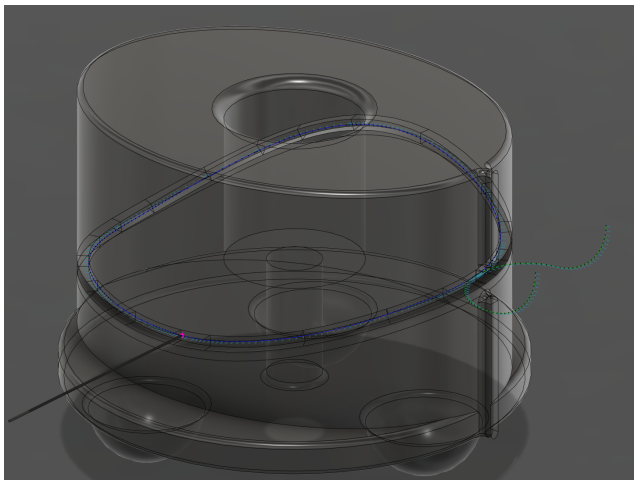


Figure 5: Toolpath generated with "Multi-Axis Contour". The dark blue line shows the toolpath, the light blue lines show the orientation of the tool, the green lines show the lead-in and lead-out movements.

Therefore, we use a generic post-processor from AutoDesk that exports the toolpath as a text file containing a list of Cartesian coordinates and orientations of the tool axis in terms of unit vectors, and perform additional, custom post-processing steps in Matlab. We use the Mathworks Robotics System Toolbox to build and solve the kinematic model of our setup.

To obtain the complete reference trajectory for our setup from the exported toolpath, we compute the closed-loop inverse kinematics from the toolpath frames on the workpiece attached to the external axis to the dosing needle on the glue gun mounted to the robot's end effector. We modify the kinematic model of the robot by moving the Tool Center Point (TCP) to the glue gun tip; this transformation can be obtained from the CAD-model of the glue gun. We also add the external axis as a revolute joint to the model; the transformation from the robot base frame to the external axis is determined with our gauging procedure detailed in Sec. 3.3.

We set up a generalized inverse kinematics solver with three constraints: The first constraint is on the joint bounds of the robot and the servo motor, the second one is a revolute joint constraint that aligns the z-axis of the glue gun with the tool axis from the toolpath, the third constraint limits the tip of the glue gun to stay within a small working envelope to ensure that most of the relative rotational movement between the workpiece and the glue gun is done by rotating the servo motor. We iterate through the toolpath by attaching the tool frames one after another to the external axis, updating the revolute joint constraint to the current frame and solving the inverse kinematics (Fig. 6). The previous inverse kinematic solution serves as the initial value of the next computation.

After obtaining the joint positions of the robot and external axis, we compute the forward kinematics to obtain the movement of the glue gun in Cartesian space. The gluing operation is now represented as a list of servo motor positions of the external axis and homogeneous transformation matrices of the glue gun.

Since the position controller of the Franka Emika Panda is prone to acceleration discontinuities and throws errors easily if the reference trajectories are not perfectly smooth, we encode the trajectories in a continuous-time regression model. With the regression model, the trajectory samples for the current time step can be computed online and we do not have to rely on that the discretization of the reference trajectory and the sample rate of the controller match. We use a linear regression model with Gaussian basis functions just as it is used in probabilistic movement primitives

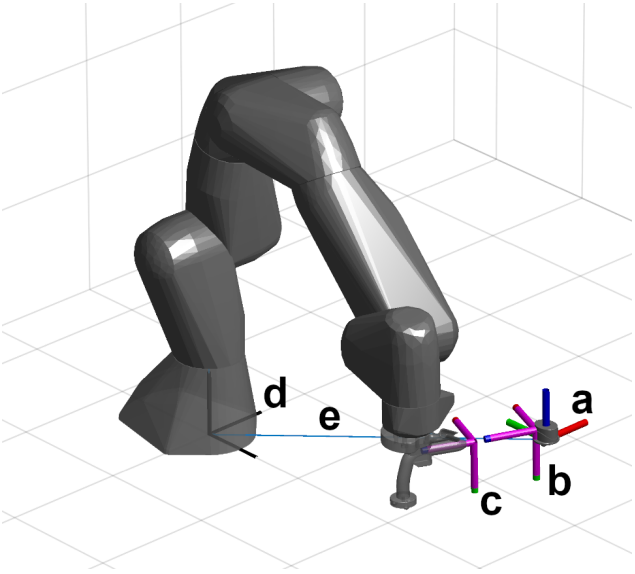


Figure 6: Kinematic model of the robot and the external axis in Matlab. **a** Origin of the external axis, modelled as a revolute joint rotating about the z-axis (blue). **b** Toolpath frame at one time instance, attached to the revolute joint of the external axis. **c** Toolframe attached to the tip of the dosing needle on the glue gun. The constraint between the z-axes (blue) of frames b and c is not fully engaged in this picture, since the frame origins do not coincide. **d** Robot base frame. **e** Transformation from robot base frame to origin of the external axis.

(ProMPs) (Paraschos et al., 2013). The orientations are converted to unit quaternions prior to encoding. An overview of the work flow is shown in Fig. 7.

3.3 Gauging the Work Cell

The CAM-based approach generates Cartesian trajectories based on a CAD-model of the workpiece localized in the robot base frame. For an ideal reproduction of the movement, the pose of the workpiece in the robot base frame used in software during the generation of the trajectory and the pose of the workpiece in the robot base frame in the real world set up should be identical. Determining the pose of the workpiece is a common issue in OLP and CNC machining, and often solved with touch probing or optical measurement systems.

For the purpose of our case study, we use a kinematic coupling on the end effector together with the robot's proprioceptive sensors to estimate the pose of the workpiece in the robot base frame. Kinematic cou-

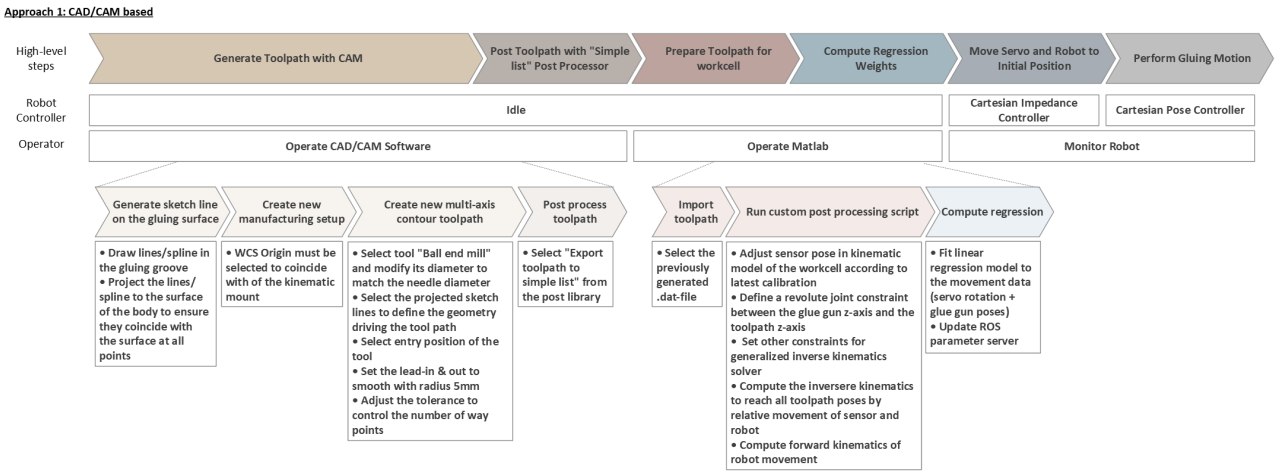


Figure 7: Schematic description of the CAM-based programming approach.

plings are fixtures used in exact constraint design which constrain parts with the minimum of necessary contact points (e.g. 6 contact points to constrain 6DOF), by which they ensure a precise and repeatable placement of the part (Blanding, 1999). The pattern we use in our coupling is a common choice in exact constraint design, and consists of a female-side with three radially oriented v-slots and a male-side with three spherical surfaces. We integrated identical male-sides of the coupling into the bottom of the workpiece and into a custom-made gauging tool. The female-side is mounted to the external axis. We assume that we can manufacture the male-sides on the gauging tool and the workpiece with acceptable accuracy using a 3D printer, such that a common reference frame placed at the center of the mount will be identically located with respect to the female-side when the mount is engaged with either the workpiece or the gauging tool.

For gauging the setup, the operator first defines the robots TCP to the reference frame of the coupling on the gauging tool; the transformation from the flange to the reference frame is taken from the CAD model of the gauging tool. The operator then removes the workpiece from the external axis, moves the external axis to its zero position and hand-guides the robot with the gauging tool into the kinematic coupling (Fig. 8). Once the coupling is engaged, the transformation from the robot base to the TCP can be captured. This transformation corresponds to the pose of the reference frame on the workpiece in the robot base frame. The transformation is then used to adjust the workpiece pose in the kinematic model (Fig. 6).

3.4 LfD-based Programming Approach

This programming approach draws on practices from physical human-robot interaction (pHRI) to create an immediate and undiluted connection between the operator, task and robot. We use a LfD related technique to transfer the expert operator's gluing performance to the robot.

Technically, we do not use LfD in the conventional sense (Siciliano and Khatib, 2016, Chap. 74), since we only record and play a single demonstration and do not arrange for generalization. We argue that this is a valid approach since we consider our case study as a form of robot programming rather than robot learning, and since we keep the task and environment controlled and unchanged. However, it must be considered for discussion that multiple demonstrations, maybe in connection with incremental learning, could help the operator to provide better toolpaths to the robot.

For recording a demonstration, the robot is put into a compliant hand-guiding mode where it can be moved around easily by the operator. Similarly, the external axis is also put into hand-guiding mode such that it can be actuated by the operator using the rotary encoder. Once the operator has moved the setup into an appropriate starting configuration, the recording of the robot joint states and the position of the servo motor is started. The operator proceeds to perform the demonstration of the gluing operation: moving the glue gun with one hand while turning the nob of the rotary encoder with the other (Fig. 9). After completing the gluing path, the operator stops the recording.

The operator runs a Matlab script to import the recorded data into Matlab where the joint states and servo positions are extracted and the servo positions interpolated such that the sampling of the joint states

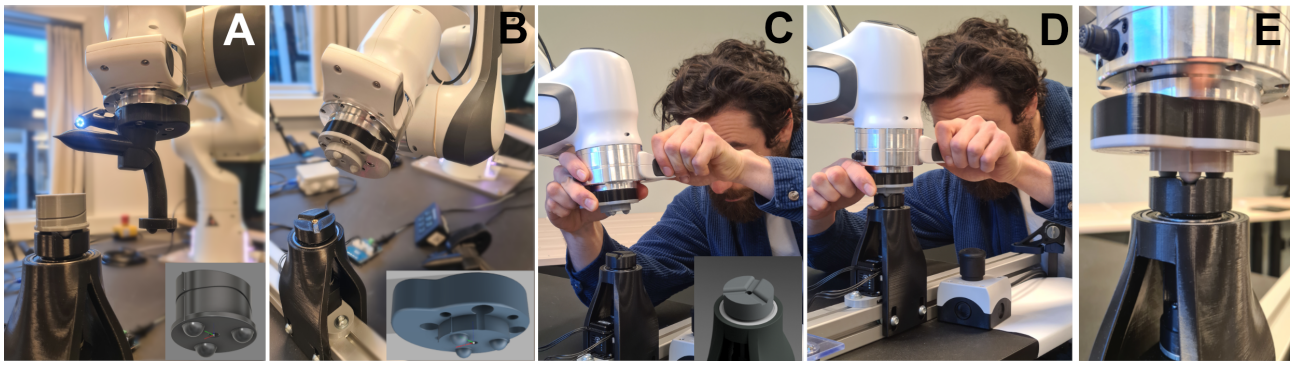


Figure 8: Gauging the setup for running the CAM-generated trajectories. The pose of the workpiece relative to the robot base frame is determined through a kinematic coupling. The coordinate frames in the center of the couplings are used to match the transformations of the gauging tool and the workpiece. **A:** Initial situation with workpiece mounted to the kinematic coupling on the external axis and glue gun attached to the robot flange. Bottom right corner shows the CAD model of the workpiece with the male-side of the kinematic coupling and the reference frame on the bottom. **B:** The workpiece is removed from the external axis and the glue gun replaced by the gauging tool featuring the identical male-side of coupling as the workpiece. The CAD model of the gauging tool with its reference frame is shown in the bottom right corner. **C:** The operator hand-guides the gauging tool onto the external axis. In the bottom right corner the CAD model of the female-side of the coupling on the external axis is shown. **D:** The operator ensures that the kinematic coupling is fully engaged, i.e. the three spherical features on the gauging tool are each in touch with one of the slots, by pushing the end effector gently downwards. **E:** Close up of the engaged coupling. The engaged coupling ensures that the location of the reference frames on the gauging tool and the workpiece coincide.

Figure 9: Operator providing a demonstration of the gluing task to the robot for the LFD-based programming approach.

and servo positions match. The glue gun poses are obtained by computing the forward kinematics from the joint states. From here on the steps are similar to the CAM-based approach (Sec. 3.2): Converting the glue gun orientations to unit quaternions and fitting a linear regression model with Gaussian basis functions to the data. An overview of the work flow is shown in Fig. 10.

3.5 Details on Hardware Integration

The setup is integrated with the `Robot Operating System` (ROS) running on a desktop computer. The servo motor is connected to the PC through a TTL to USB converter; motor commands can be sent from ROS using the `Dynamixel SDK`. The rotary encoder is connected to an Arduino Uno microcontroller board, which counts the encoder impulses using the `Encoder library`, and makes the encoder readings available to the ROS network using the `rosserial` protocol over a USB connection to the PC. In the LfD approach, we use an Elgato Stream Deck as a physical user interface, such that the operator can easily switch into hand-guiding mode and start and stop the demonstration recording with one hand while working in the setup. The robot is connected to the PC via the Franka Control Interface (FCI). Real-time control commands

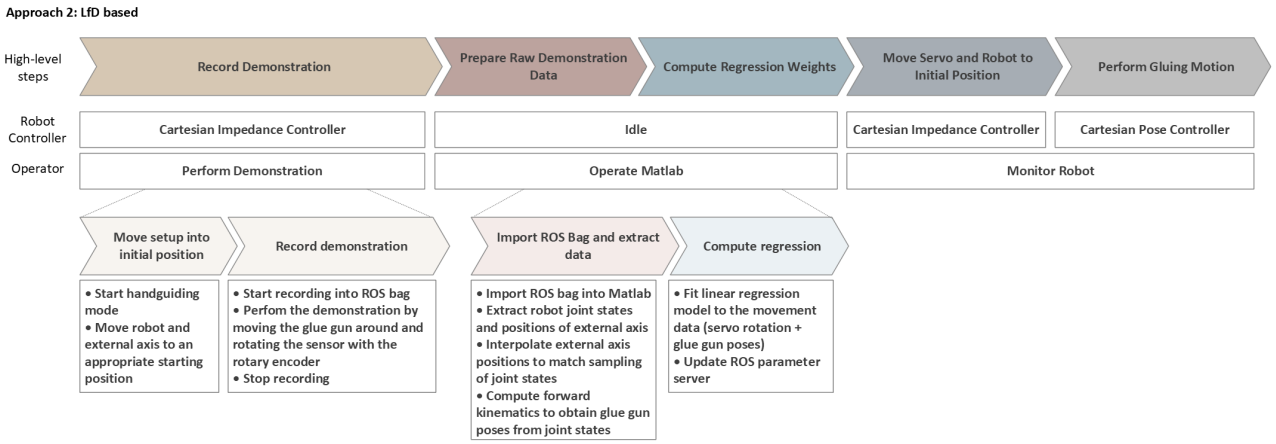


Figure 10: Schematic description of the LfD-based programming approach.

can be sent, and the robot states be read, at a sample rate of 1000Hz from ROS using `franka_ros`, which is the ROS integration of `libfranka`, an open-source C++ interface Franka Emika provides for their robots. `franka_ros` provides several control interfaces implemented in the `ros_control` framework, from which we use the Cartesian pose interface in a custom pose controller. The pose controller commands both glue gun poses to the robot and positions to the servo motor.

4 Comparison and Discussion

In the case study presented in this paper we compared two opposing approaches to programming fine manufacturing tasks on a collaborative robot. We wanted to explore what the different approaches could offer in terms of accuracy and easy of use, and get an impression of what requirements where present on the operator side to use them effectively. We also aimed at getting an impression if it would be realistic to use them in practice for programming fine manufacturing tasks.

For comparing the two approaches, we generated a toolpath with the CAM-approach as described in Sec. 3.2, and recorded toolpath demonstrations for the LfD-approach as described in Sec. 3.4. We let the robot perform these toolpaths while recording the joint states of robot and workpiece as well as videos from three perspectives.

4.1 Comparison of Performance

For the comparison, we selected a LFD toolpath we consider to be representative for the task (neither worst, nor best performance), based on our experience from setting up the experiment and previous trials.

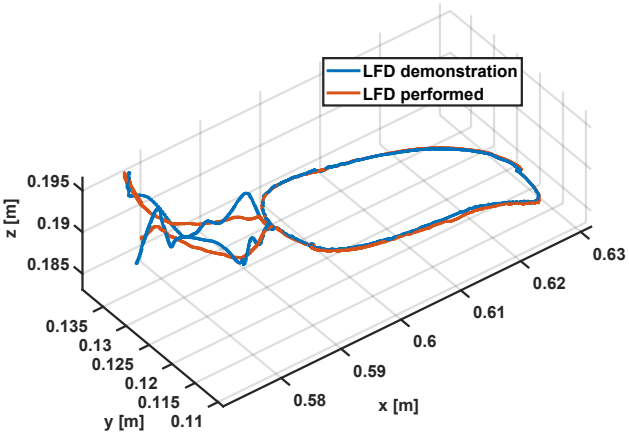


Figure 11: LFD Approach: Demonstration and performed movement.

Fig. 11 shows the selected demonstrated and performed toolpath. The paths are reconstructed from the relative movement of the glue gun tip and the workpiece. Except for the faster lead-in and -out movements the performed path is quite similar to the demonstration. Some minor deviations can be explained by errors in the regression model and tracking errors caused by inaccuracies of the robot and by physical contact between workpiece and glue gun. The demonstration shows some minor jiggle which is a sign of inaccuracies in the human demonstration.

Fig. 12 shows the toolpath generated from the CAD model of the workpiece and the same path performed by the robot. There are sections with considerable deviations between the planned and performed path. However, compared to the reference trajectory of the LFD-based approach, the planned path of the CAM approach is perfectly smooth.

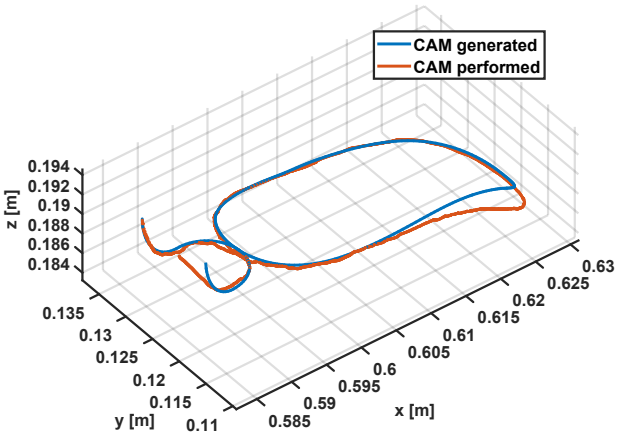


Figure 12: CAM Approach: Generated and performed movement.

For a better comparison of the approaches, their deviations and levels of noise, we plotted the Cartesian coordinates of the toolpaths in the robot base frame over the angle swept through by the relative rotation of workpiece and glue gun tip (Fig. 13).

We notice that the demonstrated path in our experiment is rather steady for a hand guiding operation, which is not always the case. During handguiding, the robot typically disturbs the operator in their movement due to imprecise compensation of the torques acting on the robot, such that the operator experiences friction and overshooting. In this task, the required movements of the glue gun are smooth and the groove on the workpiece helps the operator to stay on the desired path. A movement performed in free space, and containing more variations in acceleration, would likely show more noise. A characteristic inaccuracy for this task can be seen around $\frac{1}{2}\pi$ where the operator accidentally slips the glue gun tip out of the groove on the workpiece. Figure 14 shows a detailed view of the Z-coordinate of the LFD toolpaths together with corresponding video frames of the glue gun tip during the demonstration. The demonstration shows a "dent" where the operator slipped out of the groove. The dent is not visible in the performed path, since the glue gun tip is guided by and remains inside the groove during execution.

Figure 13 shows that the biggest deviation between planned and performed path occurs along the z-axis around $\frac{5}{4}\pi$ with the CAM approach. Based on the simulation of the generated path and the video data of the performed path, we know that the glue gun tip followed the groove on the workpiece during both planning and performance. Thus, we suspect that these deviations are a result of errors in the gauging procedure (Sec. 3.3) and tracking errors caused by physical contact between the glue gun and the workpiece: Due to small errors in the gauging procedure, the pose of the planned path does not perfectly match the groove in the physical workpiece. Since the Franka Emika Panda is not infinitely stiff even in position control mode (in fact, it implements position control internally through a stiff impedance controller), the glue gun tip can get caught in the groove and be guided by it, leading to deviations between the planned and performed movements. Figure 15 shows the z-axis plot of the CAM approach from Fig. 13 zoomed in to the section between π and $\frac{7}{4}\pi$, together with four video frames showing that the glue gun tip was inside the groove during the entire section.

4.2 Discussion

We find in our case study that the CAM approach lead to bigger path deviations due to insufficient gauging, as compared to the LfD approach. Yet, it must be kept in mind that the path deviation alone only permits conclusions about the reproduction quality of the planned path. It does neither provide information about the quality of the planned path, nor indicate how well the task was performed in terms of following the groove on the workpiece. We know from video data that despite the path deviations during the CAM approach, the task performance was acceptable, owing to the characteristics of the task and robot. For other tasks, without physical constraints facilitating the tracking/correcting errors in the gauging or with a stiff and powerful robot, the performance could be much worse or even dangerous. The case study emphasized the importance of a suitable gauging procedure to determine the pose of the workpiece in common reference frame. The procedure must be accurate to profit from the precise path planning with CAM, but it must also be fast and easy to use to enable the quick set-up of new tasks. Without such a procedure, the supposedly more precise CAM approach can perform worse than the LfD approach - even for fine manufacturing tasks.

The accuracy of the gauging procedure used in our case study may be limited by the manufacturing process (3D printing) and the size of the kinematic coupling. The coupling could be improved by machining it with high tolerances from metal, resulting in a more precise coupling and smoother surfaces. In addition, increasing the distance between the spheres on the male-side of the coupling could help to reduce the impact of manufacturing inaccuracies on the gauging.

We consider the quality of the demonstration in the LfD approach to be reasonable for such a fine task. We do not observe severe noise in the data in Fig. 13. The video data shows minor slip-ups while following the groove on the workpiece, but the overall accuracy is acceptable. Certainly, our case study is too limited

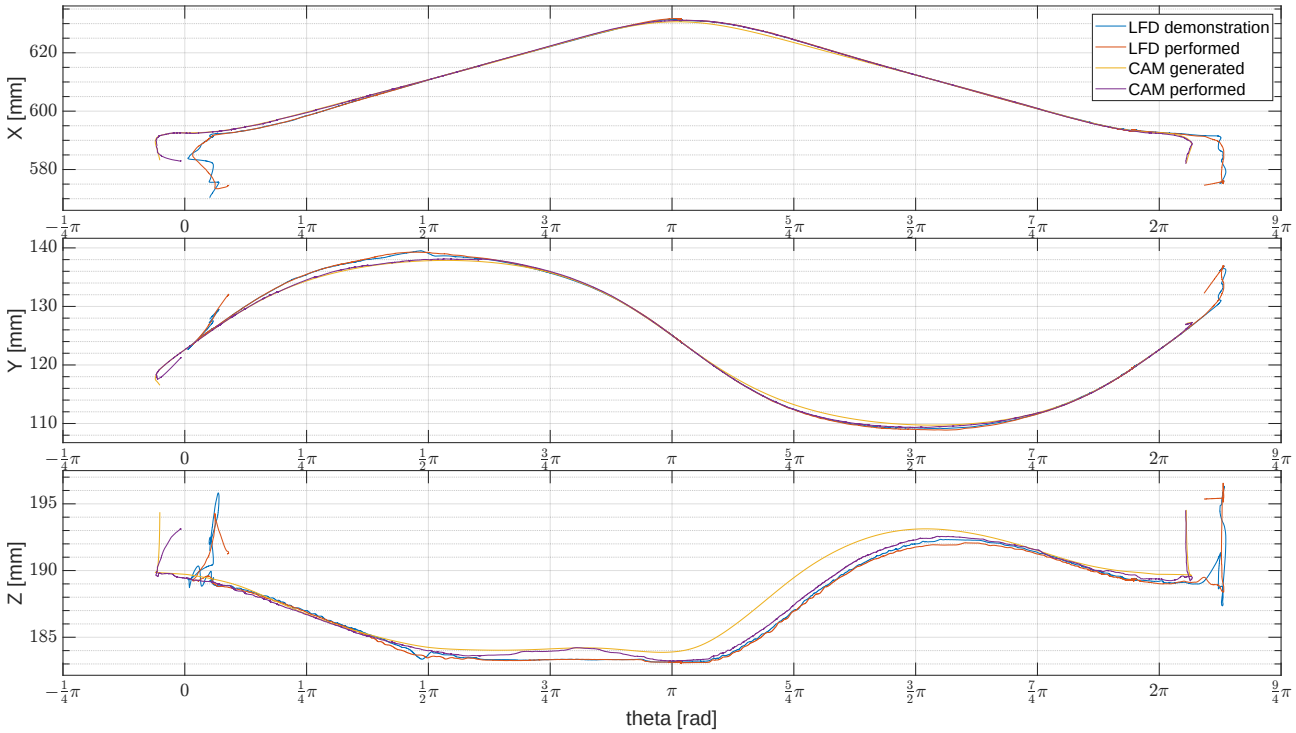


Figure 13: The Cartesian coordinates (in the robot base frame) of the toolpaths shown in Fig. 11 and Fig. 12 plotted over theta, the angle between the Y-axis of the workpiece frame and the radial vector connecting the origin of the frame and the toolpath samples.

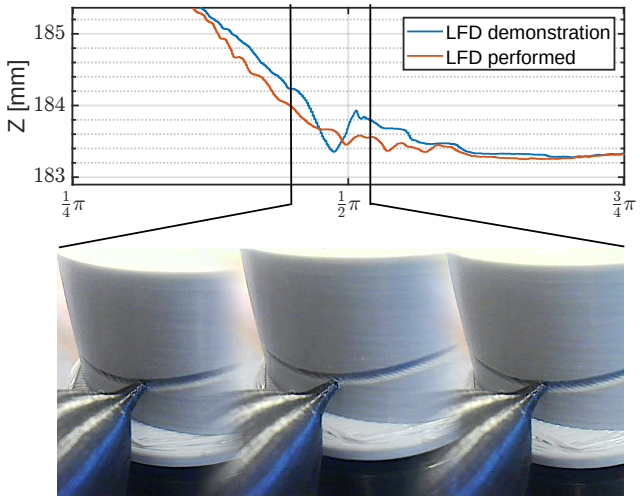


Figure 14: Detailed view of the z-axis plot from Fig. 13 and the corresponding video frames of the demonstration. The figure exemplifies a typical inaccuracy during a demonstration where the operator accidentally slips the glue gun tip out and beneath of the groove on the workpiece.

to make conclusions about the accuracy that can be achieved when automating fine manufacturing tasks on the basis of human demonstrations, yet we can say that it generally does not seem to be infeasible. Providing high-quality demonstrations may be more difficult in tasks without physical constraints that help the operator during hand-guiding, in more dynamic tasks, and in precise tasks with a larger scale where the operator can not sit still and rest their elbows as in our case study (Fig. 9).

From an operator's perspective, the LfD approach requires hands-on knowledge and the practical skills for doing the task. If these requirements are given, programming the robot by demonstration is straightforward and requires minimal preparation of the task set-up. The CAM approach, on the other hand, requires competence in CAD modelling and CAM, but not the ability to do the task manually. An understanding of the manufacturing processes involved in the task is still required to properly plan toolpaths with CAM, especially if the task involves control parameters that cannot readily determined based on the geometries of the workpiece. The accurate gauging and modelling of the setup can be challenging and can lead to production errors despite precisely planned toolpaths.

We did not include the control of additional process

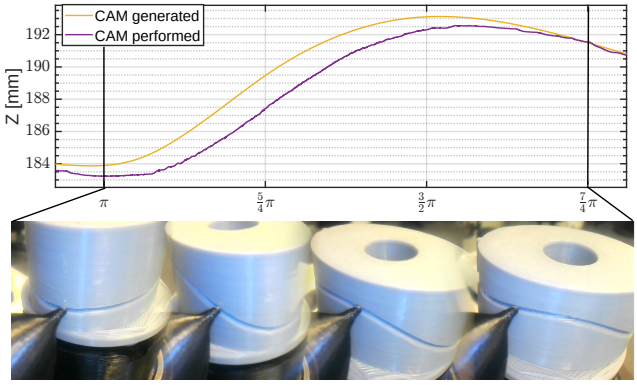


Figure 15: Detailed view of the CAM z-axis plot from Fig. 13 together with the corresponding video frames of the glue gun tip from π to $\frac{7}{4}\pi$. This figure shows how the relatively large deviation between the generated and performed CAM path between π to $\frac{7}{4}\pi$ arises: the video frames confirm that the glue gun tip stays in the groove during the entire segment, and thus, the performed path corresponds with the desired robot movement. However, the planned path does not match the performed path which indicates errors in the gauging process of the setup (Sec. 3.3).

parameters such as glue flow in our case study. Yet, similarly to programming the toolpath, the best interface for programming of such parameters depends on the nature of the task. If parameters are to be kept constant or a multitude of simple signals must be controlled simultaneously, programming them in software may be easier. If the task requires adaptive, closed-loop control of a parameter, exploiting the cognitive abilities of the operator with a demonstration-based approach may be at advantage.

5 Conclusion

Based our experience from the case study described in this paper, we believe that both the CAM-based and LfD-based approaches have something to offer for automating fine manufacturing tasks in cobot environments. The specific task characteristics that decide which approach is best suited are difficult to define on a general basis, and will depend on the operator's skills and preferences. The LfD approach is a quick and intuitive way for skilled operators to transfer their expert knowledge to the robot without any robot training, but the quality of the execution is highly dependent on how well the robot can compensate for torques, and act as a near perfect compliant device for the operator. The CAM approach requires competence in CAD-modelling and CAM in addition to an overall understanding of the manufacturing tasks, but provides a perfect geometric path for execution. However, the accuracy of the execution is highly dependent on the quality of the gauging between the workpiece mounting and the robot base frame.

For future work, we aim to investigate a hybrid programming interface that enables the operator to choose more fluidly which aspects of a task should be programmed through which approach. Such a hybrid interface allows operators to accommodate their skills and preferences, and leverages the strengths of the CAM approach and the LfD approach in programming fine manufacturing tasks on collaborative robots.

Acknowledgments

The contributions to this work according to the Contributor Roles Taxonomy CRediT are as follows: Daniel Schäle: *Conceptualization, Methodology, Investigation, Software, Writing - original draft.* Martin F. Stoelen: *Conceptualization, Writing - review & editing.* Erik Kyrkjebø: *Conceptualization, Funding acquisition, Writing - review & editing.*

This work was partially funded by the Norwegian Research Council under grant number 22071, and partially supported by the Polish National Centre for Research and Development under the project "Automated Guided Vehicles integrated with Collaborative Robots for Smart Industry Perspective" (Project Contract no.: NOR/POLNOR/CoBotAGV/0027/2019-00).

References

Blanding, D. L. *Exact constraint: Machine design using kinematic principles.* ASME Press, New York, 1999. doi:10.1115/1.800857.

Fossdal, F., Heldal, R., and Peek, N. Interactive digital fabrication machine control directly within a cad environment. In S. N. Spencer, E. Whiting, J. Hart, and C. Sung, editors, *Proceedings, SCF 2021.* The Association for Computing Machinery, Inc, New York, New York, pages 1–15, 2021. doi:10.1145/3485114.3485120.

Halim, J., Eichler, P., Krusche, S., Bdiwi, M., and Ihlenfeldt, S. No-code robotic programming for agile production: A new markerless-approach for multimodal natural interaction in a human-robot collaboration context. *Frontiers in robotics and AI*, 2022. 9:1001955. doi:10.3389/frobt.2022.1001955.

Heimann, O. and Guhl, J. Industrial robot programming methods: A scoping review. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, pages 696–703, 2020. doi:10.1109/ETFA46521.2020.9211997.

Hein, B., Hensel, M., and Worn, H. Intuitive and model-based on-line programming of industrial robots: A modular on-line programming environment. In *2008 IEEE International Conference on Robotics and Automation*. IEEE, pages 3952–3957, 2008. doi:10.1109/ROBOT.2008.4543818.

Iturrate, I., Kramberger, A., and Sloth, C. Quick setup of force-controlled industrial gluing tasks using learning from demonstration. *Frontiers in Robotics and AI*, 2021. 8. doi:10.3389/frobt.2021.767878.

Nemec, B., Likar, N., Gams, A., and Ude, A. Human robot cooperation with compliance adaptation along the motion trajectory. *Autonomous Robots*, 2018. 42(5):1023–1035. doi:10.1007/s10514-017-9676-3.

Neto, P. and Mendes, N. Direct off-line robot programming via a common cad package. *Robotics and Autonomous Systems*, 2013. 61(8):896–910. doi:10.1016/j.robot.2013.02.005.

Paraschos, A., Daniel, C., Peters, J. R., and Neumann, G. Probabilistic movement primitives. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2616–2624. Curran Associates, Inc, 2013.

Siciliano, B. and Khatib, O., editors. *Springer Handbook of Robotics*. Springer International Publishing, Cham, 2016. doi:10.1007/978-3-319-32552-1.

Villani, V., Pini, F., Leali, F., and Secchi, C. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 2018. 55:248–266. doi:10.1016/j.mechatronics.2018.02.009.