Automation of Post-Processing in Additive Manufacturing with Industrial Robots

P. Becker[1] C. Eichmann[1] A. Roennau[1] R. Dillmann[1]

***Abstract*— The use of industrial robots for the production of small scale manufacturing or even single pieces is rarely economical. The high investment of time and money required to teach a collision-free trajectory under consideration of all boundary conditions prevents the usage of robots until now. That is why it is still common practice in the industry for individual post-processing steps to be carried out manually, although they could be automated with today's technical possibilities.**

In this paper, we present an approach on how to use existing production data (STL and G-code) to generate trajectories to automate post-processing steps. These paths can then be executed by an industrial robot, for example to post-process an additive manufactured object and remove its support structures. The object may not be damaged during the process, so all movements of the robot and its tool are checked for collisions with certain parts of the object and the environment. While material is being removed, the corresponding data structure is updated accordingly to always provide a realistic representation of the current state. This approach was evaluated by removing support structures from multiple and different-shaped objects successfully. Furthermore, we used the same approach to mill pockets in material just by changing the input data.

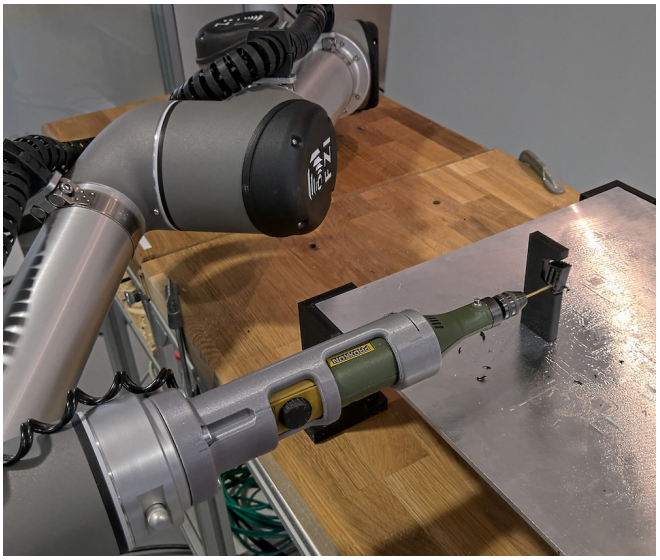


Fig. 1: The generated trajectory is executed with a milling tool to cut a form inside a block of material. The only input is the geometric description of the block as well as the model of the desired object after milling.

I. Introduction

3d printing, milling processes or spray cast all have in common that the object has to be checked by manual work after the manufacturing. This is necessary to remove undesired parts like edges or support structures from the manufacturing processes. For high quantity productions, practices already exist and the effort to automate them is worth it. But for small quantities, fast and flexible product adaptations or even unique manufactured objects the effort to teach a robot a certain post-processing trajectory exceeds the benefit [1]. That's why the task is still done manually despite current possibilities.

In this paper, we propose an approach and show the proof-of-concept to automatically generate trajectories for industrial robots based solely on the difference of the original computer-aided design data (STL) and the machine code (G-code). The two input files are used to calculate the difference between designed and finally produced object and based on this deviation a trajectory for the post-processing is generated. In the case of 3d printing, some parts have to be supported from below to be printable without any flaws. Those structures are printed non-solid and their only purpose is to prevent the upper layers from deforming due to gravity. These so-called support structures are fragile and easy to remove afterwards.

After the print is finished and the material has cooled down, those additional structures have to be removed to get the originally desired object. This removal process is nowadays still done manually by workers during the post-processing step which is time-consuming as well as unhealthy.

The approach of this paper is to automate this process and use only existing data to generate a robot trajectory to execute this task. To perform our experiment, the framework uses the CAD data and the machine code to calculate the difference between them. We decided to use a voxel data structure to be able to represent any kind of three-dimensional object as well as change the resolution depending on the current use case. The whole voxel handling and computation is done by GPU-Voxels [2], an open-source framework for real-time data processing on graphics processing units. The actual object shall not take any damage throughout the whole execution of the trajectory. That's why the path planner uses a voxel representation of the robot and the tool as well as a simplified environment to plan a collision-free path. It should be noted that contact with the object's support structure is desirable, especially during the removal process. However, contact between tool and object is only allowed within the support structures. Therefore, the

[1] Department of Interactive Diagnosis and Service Systems (IDS), FZI Research Center for Information Technology, Haid-und-Neu-Strasse 10–14, 76131 Karlsruhe, Germany. {pbecker, eichmann, roennau, dillmann}@fzi.de

different structures are all handled differently during the path planning. While executing the generated trajectory, the voxel representation is updated based on the current joint values of the robot and the consequent tool position. To achieve a high reachability, an approach orientation of the robot-mounted tool in the object space is sampled. The orientation able to reach the most target points is used to remove as much material as possible. If there remains undesired material after a first iteration, the process is repeated and a new orientation is sampled. Through the updated obstacle representation, new solutions can be made possible with each iteration.

As the framework is based on the Robot Operating System (ROS) [3] as well as the Open Motion Planning Library (OMPL) [4], it could be widely used with ease in industrial applications.

The evaluation of this approach was done by removing structural support from different 3d printed objects. Therefore, the designed model (STL file) and the resulting machine code (G-code) was given as an input and the algorithm calculated the necessary trajectory to remove the support structure.

The structure of this paper is as follows. In Section II, we present the related work. In Section III, we describe our computation approach to handle all data and generate the collision-free trajectory. In Section IV, we describe multiple application scenarios developed and experiments used to evaluate the system. Finally, we provide conclusions and perspectives in Section V.

II. Related Work

Robotic systems in bigger industrial plants are almost common today. Especially as the amount of manufactured products exceeds a certain threshold, a robot is profitable. But for small scale productions, the programming and teaching of a robot can easily be an overhead compared to manual work [1]. The planning and flexible adaptation of robot trajectories is still an ongoing research topic with many challenges [5], [6]. There are multiple methods to achieve this, like programming by demonstration [7], [8], instructive or learning systems [9], [10] as well as combination of those three [11]. New available technologies are also used to improve the way a user can teach trajectories like virtual and augmented reality [12], [13] or haptic devices [14].

Path planning is also an important research topic in machining. In an article from 2012, Chen and Dong give an overview of current research in robot-assisted machining [15]. They identify three main topics: Development of robotic machining systems, path planning for machining robots and vibration and chatter analysis and compensation. The authors name the inaccuracy of robotic arms as their main weakness. Industrial machining tasks can have strict tolerances that are currently unachievable for robots. This is especially clear in milling applications. The low stiffness of robot arms compared to classical Computerized-Numeric-Control (CNC) machines can be traced back to their serial construction. This can be attributed to the intrinsic compliance of gears and other mechanical components of each robot joint [16]. On the other hand, Chen and Dong perceive the flexibility of robot arms with many degrees of freedom as one big advantage. It allows for the machining of complex structures. They name objects with internal features as one application where robot milling can outperform traditional methods. For tasks not associated with strict tolerances, robots are already a valid tool. Chen and Dong name polishing, grinding and deburring as tasks where robots are shown to perform well. Robots achieve great quality of work in this fields with their ability to align an end-of-arm tool with the surface normal of the processed surface.

In 2019, Ji and Wang published a review on industrial robotic machining covering more than 30 years of research [17]. They focus on hard material machining. The existing research for machining with a low material removing rate like polishing and deburring aims mainly to achieve human-like operation and high surface quality. As of today, robot stability and machining vibrations are still an important research topic for high material removing rate processes like milling. The authors see the need for further improvements before industrial robot machining can rival the established CNC machines.

III. Concept

Figure 2 shows a simplified architecture highlighting the major components of the software stack. The first of them is the object segmentation, which produces a voxel representation from incoming data like the STL and G-code file or even point cloud sensor data. For clarity, we focus on STL and G-code data within the next pages while we describe the algorithms. The flexible framework is still able to handle more types of CAD or sensor data, but the necessary converters are currently not implemented.

The object representation, in form of several voxel sets, is used by the path planning to generate the desired post-processing trajectories. A visualization of a parsed G-code file can be seen in Figure 3. This kind of data representation has been chosen as it is close to the real world representation needed for the collision detection while planning the robot trajectories and is also able to perform operations like adding and subtracting multiple voxel layer. During material removal, the object representation is updated based on the tool's position. For this, the separation of polymer chips by the tool is simulated. The trajectory can be executed using a real robot manipulator or by running the whole process within a simulation.

GPU-Voxels [2] is a centerpiece for the trajectory generation presented in this work, as it is used to model the workspace in which the robot operates and also the different representations of the 3d printed object. There are multiple data sources contributing to the voxel environment: the STL import, G-code interpreter, the robots current position derived from its state, and planned trajectories represented by swept volumes. The main beneficiary of this representation is an OMPL planner [4]. It utilizes the detailed object representation and the more coarse grained robotic workspace to validate possible robot poses and motions. This is done

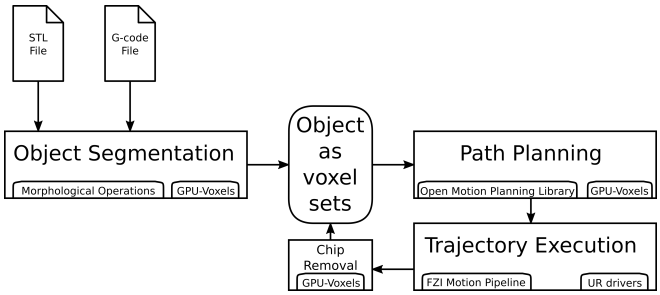


Fig. 2: High-level architecture diagram. The input data is segmented and voxelized and depending on the use case the trajectory is generated with integrated collision check. The calculated trajectory can than be executed by a motion stack like the FZI Motion Pipeline or MoveIt! [18].

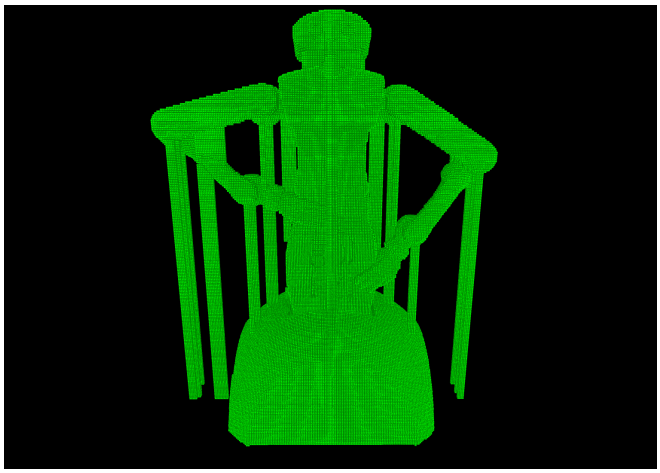


Fig. 3: Result of G-code voxelization containing the desired object and the support structre. This is the base for path planning with OMPL as well as collision checking.

by checking for self-collisions and collisions between the robot and its environment. The planner is wrapped in a ROS nodelet providing an action for motion planning.

Planned paths are composed from sequences of valid robot poses. To execute such a path, the FZI's own framework for linear interpolation of robot trajectories, the Motion Pipeline, is used. The inverse kinematics solver TRAC-IK [19] is utilized to transform cartesian target positions into robot poses. TRAC-IK combines two common inverse kinematic solvers to provide fast solutions to simple problems while also being able to solve complex ones.

Utilizing ROS, the system is optimized for automation and re-use. User interaction is not needed at runtime, input files can be specified as parameters beforehand. ROS launch configurations are given and provide the option to adapt various parameters like build size or voxel resolution in one place.

The most common file type for representing objects for 3d printing is STL. For this work, a process to voxelize STL files has to be developed. To import a model into a GPU-Voxels data structure, the STL file is first converted using binvox [20], [21], a program capable of creating compressed binary voxel grids from STL files. This step is encapsulated in a stand-alone ROS node, which provides a ROS action to analyze and convert a given STL file. From the created binvox file, a GPU-Voxels voxel set can be populated.

To operate on the voxel sets obtained by STL import and G-code interpretation, the sets have to be matched in their position and rotation in the GPU-Voxels environment. This can be done automatically by calculating the needed translation from the known position of the objects bounding boxes. If the object is rotated in relation to the STL before slicing, this simple method is not applicable. Rotations by multiples of 90 degrees around the X or Z axis of the object can be compensated by using invocation arguments of the binvox program. Other, more complex rotations require more advanced matching strategies. Algorithms for point cloud matching like Interative Closest Point (ICP) [22], [23] could be applied.

To facilitate support removal, the voxels need to be separated into different groups. Of main interest is the differentiation between parts that are desired and support that needs to be removed. The following paragraphs discuss the separation of voxels into meaningful sets.

All voxels generated by the STL import provide the set STL, while all voxels from G-code interpretation form the set $GCODE$. Binvox voxelizes a given STL as a solid voxel set, therefore all voxels representing the outer shell of the object (the set $SHELL$) and, if existing, the infill pattern ($INFILL$) are included:

$$SHELL \subseteq STL \quad \text{and} \quad INFILL \subseteq STL \tag{1}$$

Every voxel from G-code interpretation can be classified as either object shell, infill, or support, which leads to Equation 2. Every voxel that is not desired (not part of the shell or infill) but printed regardless is classified as support, as formalized in Equation 3, which also means that the STL model does not contain support, as shown in Equation 4.

$$GCODE = SHELL \cup INFILL \cup SUPPORT \tag{2}$$

$$\begin{aligned} SUPPORT &= (GCODE \setminus SHELL) \setminus INFILL \\ &= GCODE \setminus (SHELL \cup INFILL) \end{aligned} \tag{3}$$

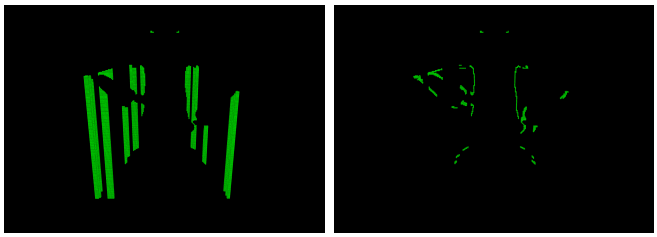
$$STL \cap SUPPORT = \emptyset \tag{4}$$

From Equations 2, 3 and 4, it can be concluded that a set containing all support voxels is the relative complement of STL in $GCODE$, as presented in Equation 5.

$$SUPPORT = GCODE \setminus STL \tag{5}$$

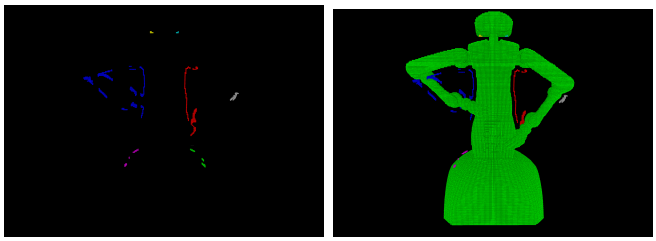
In reality, quantization errors occur from limited voxel resolution and from limited printer resolution. The layer height acts as limit in Z direction while the smallest printable unit in the XY plane is defined by the nozzle diameter. Therefore it is advantageous to dilate STL before executing the aforementioned Equation 5 to reduce the amount of voxels wrongfully classified as support. Q in Equation 6 is a structuring element designed to counteract quantization errors.

$$STL' = STL \oplus Q \tag{6}$$



(a) $SUPPORT'$ (b) $CONTACT$

Fig. 4: Voxel sets generated by Equations 8 and 9 from the HoLLiE STL and G-code files



(a) Voxel cluster (b) Voxel cluster with HoLLiE model

Fig. 5: Voxel cluster of support contact regions in different colors

Equations 1 is also valid for STL'. Equations 4 and subsequently 5 are generally not valid for STL' because voxels from $SUPPORT$ that are close to STL get included into STL' through the dilation. These voxels can be expressed through the error set formulated in Equation 7. A new, reduced set $SUPPORT'$ can be defined as shown in Equation 8 and visualized in Figure 4a.

$$ERROR = SUPPORT \cap STL' \tag{7}$$

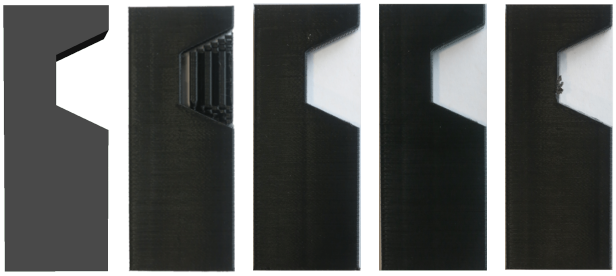
$$\begin{aligned} SUPPORT' &= SUPPORT \setminus ERROR \\ &= GCODE \setminus STL' \end{aligned} \tag{8}$$

Experiments with different object models show that the proposed dilation of the STL set is valid approach to counter quantization error.

It is advantageous to segment the contact zones between the STL model and the supports for a targeted removal of the connecting polymer with a robot mounted tool. Equation 9 shows how to determine this contacts, with N being a $3 \times 3 \times 3$ neighborhood and Figure 4b visualizes the result.

$$CONTACT = SUPPORT' \cap (STL' \oplus N) \tag{9}$$

After the contact zones are extracted, it's advantageous to cluster them into groups of contiguous voxels. This allows to segment the distributed contact voxels by spacial relations and by the support structure they are a part of. The clustering algorithm applied follows an iterative region-growing approach. A is the voxel set which shall be clustered, e.g. the set containing all contact voxels. B_0 is a set containing one arbitrary voxel a with $a \in A$. B_{n+1} is calculated following Equation 10. The structuring element S_r is chosen as sphere with radius r. r is the length a region grows in one step and the minimal distance between two distinct clusters.



(a) (b) (c) (d) (e)

Fig. 6: The sample object as STL model (a), printed with support (b), post-processed by the robot (c) and a human (d) and with a possible processing error (e)

$$B_{n+1} = (B_n \oplus S_r) \cap A \tag{10}$$

A cluster is found if $B_{n+1} = B_n$, after which B_{n+1} is removed from A and the process is repeated with a new B_0 from the now reduced A. All clusters are found if $A = \emptyset$.

Figure 5 shows seven voxel clusters determined by the presented algorithm, both alone and with the model generated from STL file.

IV. Evaluation

The system was evaluated using a Universal Robots UR5 lightweight robotic manipulator [24] and a standard drill mounted along the z-axis of the robot's tcp. The robot itself is mounted to a workbench, which also includes mounting points for the 3d printers build plate at the same level. The plate can be fitted in place without additional tools as the plate mount is already calibrated in the robotic system. The whole setup can be used in simulation with Gazebo [25] for checking beforehand and is also implemented as hardware demonstrator.

The discussed system is used to post-process a 3d printed object with contiguous support volume. Figure 1 shows the tool in contact with the support structure while the removal trajectory is executed. The result of this process is shown in Figure 6, together with the original model of the desired object and a specimen finished manually. Additionally, a part with a forced error is provided for comparison with the untainted object. This can happen when the tool is misaligned and the side of the drill touches the object, while the drill tip travels to the nearest points on the inner back. Such a processing error can be an indicator for incorrect calibration.

In order to show the diversity of the approach, several different objects, each with its own special features were post-processed. For example, Figure 7 shows a dry-run in simulation. This can be used to validate the planned trajectory without any damage. Especially for difficult objects with multiple clusters or with dimensions that force the robot near its joint limits, this can be helpful to avoid harmful damages. The algorithm plans the trajectory so that the contact surfaces from the support structures are removed first to reduce the time of post-processing. When the support structure is directly between a part of the build plate and the

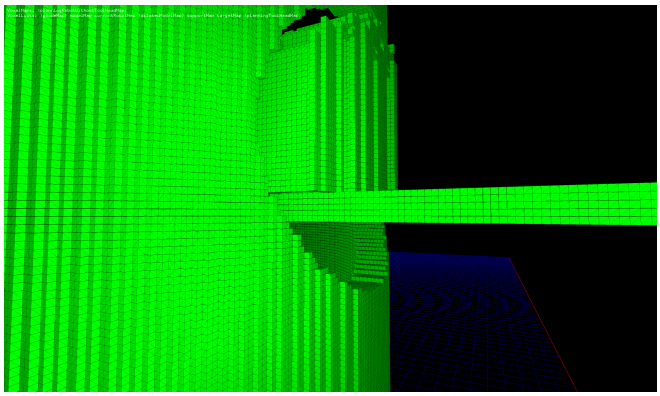


Fig. 7: Simulate the removal of the support structure inside an cylindric object. The lower part of the support structure was already removed and the voxel representation updated accordingly.

object the robot will not try to remove the lower attachment to the build plate. Only the structures attached to the objects will be removed and so the printed object is still freed from the undesired structures.

In Figure 7 the lower part of the structure has already been removed, these voxels are no longer visible anymore. The voxels are updated continuously, so that this free space can be used by the path planning in further process steps.

The proposed system is able to create trajectories removing multiple disjointed support structures by clustering and iterative processing. The generated trajectory executes cuts without collisions. A more complex example is to remove the support structures from the small humanoid robot object. The challenge here is the fragile structure of the object and the multiple support structure columns with lower attachment on the build plate and some at the object itself. The post-processed object is shown in Figure 8.

There is no direct time saving for this robot-based processing compared with manual work. An advantage can be derived from the ability of the robotic system to operate both day and night without the need of human supervision. This and the reduced cost compared to a human worker on the long run can offset the increased processing time.

The system and its object description via voxel sets is flexible enough to easily integrate different manufacturing procedures. A logo is milled into a polystyrene foam tile using the same pipeline described above. The desired tile including the logo is voxelized instead of the STL file and the voxel set generated from G-code is replaced by one from a model of the unprocessed tile. It was sufficient to adjust the input data accordingly. A change of the source code was not necessary. Three independent clusters of material to be removed are extracted, as shown in Figure 9. The trajectory is executed by the robot and the final product can be seen in Figure 10. Three trajectories were generated by the system, each considering one cluster individually and thus the rest of the model was not damaged. Between clusters, the robot returns to a defined position from which it moves to the next cluster.

During experimentation, it became clear that the reach of

Fig. 8: The robot model of HoLLiE while post-processing and removing multiple columns of support structure from the print. The first columns have already been removed.

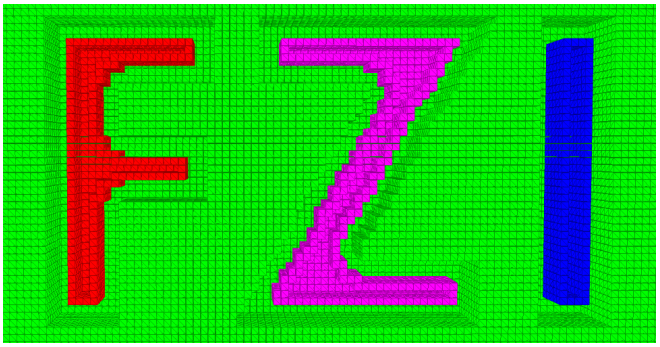


Fig. 9: The voxel representation of a material removal trajectory. The different voxel clusters to be removed by the robot are colored in red, violet and blue.

Fig. 10: The logo milled utilizing the proposed system. No adaption of source code was necessary just the data was changed to the one of the logo and the block.

Universal Robots UR5 is constraining the type and size of the processed objects considerably. A change in hardware is proposed to address this issue. A robot with a bigger workspace could be used or mounting the robot on a linear axis could also increase its reach. Lastly, the working area including the mounting points for the build plate could be located on a turntable to utilize the given robot workspace better.

V. Conclusions

This work shows that the automation in additive manufacturing of certain manufacturing steps, especially post-processing, can be done just by using the already existing production data and therefore, it is even affordable for small scale productions. Previously, the robotic automation of manufacturing steps for small series or individual pieces was very time-consuming, but with this approach it is possible to plan robot trajectories easily for post-processing steps especially needed in the uprising additive manufacturing.

In this publication we addressed the challenges of the real world interaction of a robot and a printed object just by using the already existing process data to execute the necessary post-processing. Based on GPU-Voxels in combination with ROS and OMPL, the solution can provide data-based and collision-free trajectories for industrial robots. Especially in difficult cases with structures that had no direct connection, multiple clusters were identified by the algorithm and based on those suitable trajectories could be generated that did not damage the object. The evaluation of the system showed that the post-processing like the removal of support structure can be done automatically by a robot even for small scale manufacturing. Different objects have been tested to evaluate the generalization of this approach. At the same time, the framework can also be used to do subtractive manufacturing by milling or drilling of material.

The overall setup could be improved by reducing the calculation time during the path planning, especially the collision check routine. Furthermore, the reachability of target points inside the workspace could be diversified by adapting the current hardware setup. Therefore, changing the robot model to one with greater reach, adding a linear axis or placing the build plate on a turntable are possible options. In future, a combination of the presented approach and automatic unloading of machines [26] can increase the throughput of additive manufacturing and reduce the amount of manual work at the same time.

References

[1] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," in *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*. VDE, 2010, pp. 1–8.

[2] A. Hermann, S. Klemm, Z. Xue, A. Roennau, and R. Dillmann, "Gpu-based real-time collision detection for motion execution in mobile manipulation planning," in *2013 16th International Conference on Advanced Robotics (ICAR)*. IEEE, 2013, pp. 1–7.

[3] M. Quigley, J. Faust, T. Foote, and J. Leibs, "Ros: an open-source robot operating system."

[4] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[5] R. Alterovitz, S. Koenig, and M. Likhachev, "Robot planning in the real world: research challenges and opportunities," *Ai Magazine*, vol. 37, no. 2, pp. 76–84, 2016.

[6] V. Villani, F. Pini, F. Leali, C. Secchi, and C. Fantuzzi, "Survey on human-robot interaction for robot programming in industrial applications," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 66–71, 2018.

[7] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, 2004.

[8] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," *Springer handbook of robotics*, pp. 1371–1394, 2008.

[9] B. A. Pearlmutter, "Learning state space trajectories in recurrent neural networks," *Neural Computation*, vol. 1, no. 2, pp. 263–269, 1989.

[10] Y. Huang, D. Büchler, O. Koç, B. Schölkopf, and J. Peters, "Jointly learning trajectory generation and hitting point prediction in robot table tennis," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 650–655.

[11] P. Neto, D. Pereira, J. N. Pires, and A. P. Moreira, "Real-time and continuous hand gesture spotting: An approach based on artificial neural networks," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 178–183.

[12] F. A. Candelas, S. T. Puente, F. Torres, F. G. Ortiz, P. Gil, and J. Pomares, "A virtual laboratory for teaching robotics," *complexity*, vol. 1, pp. 10–11, 2003.

[13] C.-H. Chu, Y.-W. Liu, P.-C. Li, L.-C. Huang, and Y.-P. Luh, "Programming by demonstration in augmented reality for the motion planning of a three-axis cnc dispenser," *International Journal of Precision Engineering and Manufacturing-Green Technology*, pp. 1–9, 2019.

[14] A. Muxfeldt, J. N. Haus, J. Cheng, and D. Kubus, "Exploring tactile surface sensors as a gesture input device for intuitive robot programming," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2016, pp. 1–4.

[15] Y. Chen and F. Dong, "Robot machining: recent development and future research issues," *The International Journal of Advanced Manufacturing Technology*, vol. 66, no. 9, pp. 1489–1497, Jun 2013. [Online]. Available: https://doi.org/10.1007/s00170-012-4433-4

[16] A. Karim and A. Verl, "Challenges and obstacles in robot-machining," in *IEEE ISR 2013*, Oct 2013, pp. 1–4.

[17] W. Ji and L. Wang, "Industrial robotic machining: a review," *The International Journal of Advanced Manufacturing Technology*, Apr 2019. [Online]. Available: https://doi.org/10.1007/s00170-019-03403-z

[18] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[19] P. Beeson and B. Ames, "Trac-ik: An open-source library for improved solving of generic inverse kinematics," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, pp. 928–935.

[20] P. Min, "binvox," http://www.patrickmin.com/binvox, 2004 - 2019, accessed: 2019-05-24.

[21] F. S. Nooruddin and G. Turk, "Simplification and repair of polygonal models using volumetric techniques," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 191–205, 2003.

[22] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb 1992.

[23] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, Oct 1994. [Online]. Available: https://doi.org/10.1007/BF01427149

[24] "Ur5/cb3 original instructions (en)," Universal Robots A/S, Energivej 25, DK-5260 Odense S, Denmark, 2019.

[25] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sep 2004, pp. 2149–2154.

[26] P. Becker, E. Henger, A. Roennau, and R. Dillmann, "Flexible object handling in additive manufacturing with service robotics," in *2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA)*. IEEE, 2019, pp. 121–128.