Real-time data exchange (RTDE) robot control integration for Fabrication Information Modeling

Martin Slepicka[1], Jalal Helou[1] and André Borrmann[1]

[1]Chair of Computational Modeling and Simulation, Technical University of Munich, Germany

martin.slepicka@tum.de, jalal.helou@tum.de, andre.borrmann@tum.de,

Abstract -

In response to its stagnating productivity and growing demand for sustainability, the architecture, engineering, and construction (AEC) industry is increasingly pushing ahead regarding digitization and automation. Building Information Modeling (BIM) and Additive Manufacturing (AM) are two technologies that contribute significantly to this development. Although these technologies have similar basic principles — both are based on computer-aided methodologies and used for automation purposes — they have long been developed independently in the construction industry. However, recently, studies on integrating AM into BIM methodology have gained momentum. One approach is the Fabrication Information Modeling (FIM) methodology which aims at interfacing digital design with automated manufacturing by providing a fabrication-aware intermediate layer, a digital model that contains all the information necessary for automated fabrication. This study focuses on extending FIM by providing a new robot control methodology that can directly utilize FIM data to control an AM system and is better suited to extrusion-based concrete 3D printing needs. The proposed robot control method aims to be more flexible regarding the recalibration of printing parameters and prepares the basis for developing systems for automated quality control.

Keywords -

Fabrication Information Modeling; AM; Robotics

1 Introduction

The architecture, engineering, and construction (AEC) industry has been stagnant for a long time regarding the industry's productivity. In response to this, and to satisfy the growing demand for sustainability, the industry is increasingly pushing ahead in digitization. Critical components of this development include Building Information Modeling (BIM) and Additive Manufacturing (AM). On the one hand, there is BIM, a methodology developed to enable seamless use of digital data throughout the lifecycle of a building [1]. On the other hand, there is AM, a term that refers to various additive manufacturing processes that can resource-efficiently generate complex geometries from building materials based on digital data. However, although both technologies are based on computer-aided methodologies, they have long been developed separately in the industry. Among other things, this has also prevented the development of uniform data modeling guidelines, which has made it challenging to synergize the two technologies. This fact has led to various efforts to combine the two technologies [2]. Among other things, a methodology was developed to derive manufacturing information from BIM data for use with extrusion-based AM methods, i.e., Fabrication Information Modeling (FIM) [3].

With the help of FIM, tool paths for 3D printing of building components can be generated based on parametric patterns. In addition, with interfaces to various robot control frameworks, printing can be included in the digital workflow, regardless of which robot system is used. However, it should be noted that the FIM data are mostly not read directly or conversion-free by the robot control system but are interpreted via the interface.

Another problem is that concrete printing is a highly delicate matter. For fresh concrete to be processed using extrusion-based 3D printing, it must be pumpable, extrudable, and buildable [4]. These three properties contradict each other to some extent. On the one hand, the concrete must be flowable to be pumped. However, on the other hand, it must remain dimensionally stable as it is extruded through the nozzle and must be load-bearing fast enough to withstand further printing layers. In addition, external influences can change these rheological properties; among other things, pumping too fast can negatively affect the material's composition, while printing too slow can lead to poor inter-layer bonding [4]. Further negative influences are time-dependent due to the long curing time of the concrete. Others originate from, e.g., inertia effects during rapid changes of direction [4] or simply from environmental influences, such as temperature or humidity [5].

For these reasons, AM processes in construction are heavily dependent on human supervision. Concrete printing often fails without quality control and recalibrations, resulting in inferior component quality. In order to increase the degree of automation, this study proposes a robot control methodology that is significantly more adapt-

able than other comparable systems. On the one hand, the system can process the FIM data directly without intermediate steps. On the other hand, it is suitable to plan a velocity profile taking inertia effects at direction changes into account. Furthermore, the system enables adaptation of the fabrication information during the printing process and thus provides new possibilities to integrate sensor systems into the AM system for automated quality control. It can quickly be recalibrated, and parameters relevant to 3D printing can be changed by user input.

In the following, we will first show how conventional methods are carried out. Then the proposed methodology is described in detail, and improvements compared to conventional methods are discussed.

2 Background

In this study, the focus is on extrusion-based concrete printing methods. One of the first, and still representative, processes developed is Contour Crafting (CC), in which concrete is forced through a nozzle to form a strand (or filament) and then deposited layer by layer according to a digital model. CC was developed in analogy to the Fused Deposition Modeling (FDM) 3D printing method, processing a thermoplastic in the same functional principle.

As with FDM, the corresponding digital model is created as a volume model using computer-aided design (CAD) and stored in a data format suitable for the subsequent slicing operation, usually as Stereolithography (STL) file. The subsequent slicing process cuts the volume model into several equidistant 2D slices. These slices are then converted into tool paths that describe the movement of the nozzle and, thus, the location where the material is applied. Usually, these paths first follow the contour of the slice and then fill the rest of the surface following a specific pattern (hatching). After that, the planned path is transformed into a robot trajectory, i.e., a series of robot poses that the machine must traverse, and communicated to the 3D printer along with the extrusion rates via G-Code or other specific interfaces [6]. Executing this robot control code will then start the printing process.

This trajectory translation depends on the selected robot system, e.g., for an industrial robot arm with six degrees of freedom (6-DoF), six coordinates describe each pose. In the Cartesian representation, three coordinates (X, Y, Z) describe the position of the robot's end-effector, and another three coordinates (R_X, R_Y, R_Z) describe its orientation. However, the robot requires these coordinates as a list of joint positions (θ_1, θ_2, θ_3, θ_4, θ_5, θ_6), which must be calculated by inverse kinematics (usually done by the robot control unit) [7].

Inverse kinematics transforms a robot's end-effector coordinate frame from Cartesian space into Joint space using Denavit-Hartenberg-Transformations. For a 6-DoF robot arm, six chained transformations must be solved for the respective Joint angles, which is a very time-consuming calculation and usually provides multiple results (more than one possible pose). The transformation from Joint space to Cartesian space is called forward kinematics, a more straightforward problem that provides only one solution.

2.1 Digital model

The representation of the geometry (digital model) is the output of the design process that usually describes the product's final form. As previously stated, the digital model is usually created using the user's preferred CAD tool. However, with CAD, it is only possible to generate the geometry of the component to be manufactured. Additional information, e.g., the material to be used or varying feed rates, cannot be considered in this process [8]. Additionally, the representation requires several preparation steps for fabrication, such as meshing into the STL format and layer-wise slicing, introducing multiple areas of deviations, and hindering automation.

Furthermore, an STL mesh exhibits numerous redundancies and cannot model curved segments accurately except with significantly large file sizes [9]. Thus, the aspects needed for fabrication are handled after the design process rather than incorporated into it. Nevertheless, with the extensive knowledge and support for CAD files regarding meshing, optimization, and simulation, concrete 3D printing (C3DP) processes based on regular CAD are the current standard [6, 10].

On the other hand, BIM has been proven to offer multiple advantages to the construction industry and C3DP, such as the support for additional parameters for fabrication and standard support for Industry Foundation Classes (IFC), which is advantageous for AM applications [8, 9]. Several studies have taken advantage of the flexibility of BIM and developed automated BIM-based AM fabrication processes [8, 11].

The proposed methodology in this study uses the FIM methodology developed by Slepicka et al. [3] for generating the required digital model instead of using a simple CAD tool. FIM offers a pattern-based approach where voids, inserts, supports, and other functional inserts can be integrated directly into the design. Another advantage of using FIM is that the path-planning tasks are already incorporated in this step. Therefore most of the tasks are semi-automatically applied already in the design phase.

2.2 Planning and Execution

As mentioned before, for path planning, it is standard practice to use slicing software, which converts an STL file into tool paths that can be exported as robot control code. The slicer program will use a specific in-fill pattern,

density, and material based on user input, specifying how the end-effector's tool path is generated. Typical slicing software solutions provide different output formats, but the old industry standard DIN 66025, commonly known as G-Code, is often used for 3D printing [8, 11]. A different approach gaining much interest is using the Robot Operating System (ROS) framework, which provides MoveIt!, a motion planning framework, for path planning [12]. MoveIt! can plan the path, calculate inverse kinematics, and detect collisions, among other features.

When the path planning is complete, the manufacturing information is transmitted to the robot, which performs the 3D printing. There are two different methods for this task, offline and online programming. In this context, offline programming refers to transmitting manufacturing information as a whole, which is then executed sequentially. Online programming, on the other hand, describes the continuous transmission of manufacturing information in small packages, which enables more control and room for feedback [13].

Using MoveIt!, offline programming is the standard, where the complete path and the joint coordinates are precalculated. Although this approach allows collision checking, it does not allow real-time checks and adaptations to the printing path.

With FIM, the planning process for the fabrication is, as mentioned in the previous section, shifted to the design phase, when the digital model is created and, in part, automated. Using FIM, BIM data is extracted and processed by applying design patterns, generating semi-automatically (the user must choose design parameters) fabrication information for a selected component. While the outer shape of the component is retained from the BIM design, the pattern is applied to generate the interior structure of the component. In contrast to conventional slicing methods, with FIM, the voids are designed by path planning, not the other way around [3]. The fabrication information (path and parameters for material, process, and machinery) is stored in the BIM exchange format IFC (cf. fig. 5).

When combining FIM with the methodology proposed in this study (cf. section 3), the manufacturing information can be directly transmitted with the online programming strategy to an AM system. As discussed in section 3.1.2, an object-oriented planning algorithm is proposed that complements the FIM-IFC representation [3], enabling direct data access and the generation of an adapted velocity profile. The proposed high-frequency online communication uses the Real-Time Data Exchange (RTDE) offered by Universal Robots™ (UR), which synchronizes the UR controller with an external client over TCP/IP. While this communication protocol is specific to UR CB-Series and e-Series robots, other industrial robots offer similar protocols for data exchange, e.g., KUKAs Robot Sensor Interface (RSI) [14].

3 Methodology

The proposed methodology consists of several modules described in detail in this section, including their execution. In general, the RTDE methodology is supposed to directly process digital models generated by FIM [3] and execute the contained fabrication information. Thus, it extends the FIM framework, providing means to generate system-specific velocity profiles and to directly control AM systems. At the same time, the system also opens up possibilities to feed data back to the FIM model during the printing process. Figure 1 summarizes the different modules and their connections to each other.

3.1 Modules

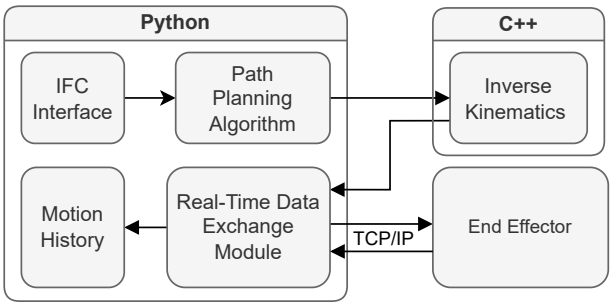

Figure 1. A Flowchart of the Different Modules used within the Methodology

3.1.1 IFC Interface

As discussed in [3], FIM uses the IFC format to store the fabrication information (printing path and other variables relevant to the printing process). Using the available ifcOpenShell library for Python, the IFC file's contents can be read and queried easily. The code queries for layers that are geometrically represented as `IfcCompositeCurve` entities, which in turn contain `IfcCompositeCurveSegment` describing the geometry of each layer segment. While the specific geometry can be of any IFC representation, the current code is equipped to handle `IfcTrimmedCurve`, `IfcPolyLine`, and `IfcSpline` entities. All the `IfcCompositeCurve` entities are stored as layer objects, whereas `IfcCompositeCurveSegment` entities are translated into segment objects within the program. Segments are parametrized and can be evaluated using $t \in [0, 1]$ and discretized into N equal segments or based on an array of step sizes utilized within the path planning module.

This object-oriented approach allows for a more straightforward adaptation of other IFC entities and allows control over the required discretization in later workflow stages. The layer and segment objects can be offset, scaled, and trimmed using their respective method functions.

3.1.2 Velocity profile

Initially, the velocity profile algorithm assumes a constant velocity for the entire layer. It then adds a gradual acceleration and deceleration at the start and end, respectively, represented by a hyperbolic tangent profile. An exemplary velocity profile is shown in Fig. 2 for one layer.

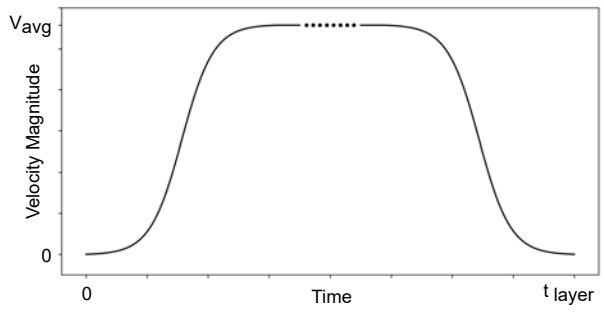

Figure 2. Schematic of the Assumed Velocity Profile of one Layer within the Path Planning Algorithm

The profile is entirely defined by the user input values of the layer printing time and maximum acceleration/deceleration values. Based on the frequency of the established connection with the end-effector, an array of step sizes is calculated, defining the distance between the discretized points. These step sizes are fed sequentially to each segment's `discretize` method to obtain a list of waypoints for the end-effector to follow. The velocity and acceleration in each direction are also calculated to ensure that the machine limits are not exceeded.

3.1.3 Inverse Kinematics

With the printing layer segments discretized into an array of cartesian points, inverse kinematics is performed to obtain an array of joint coordinates that the specific end-effector can execute. These calculations are the most time-consuming among the modules and thus are implemented using C++, utilizing the mathematical library *Eigen* for matrix operations and *pybind11* for Python bindings. The compiled program offers a `kinematicCalculator` object that can be initialized for a specific end-effector currently limited to UR5, UR5e, UR10, and UR10e but can be extended to any end-effector with known Denavit–Hartenberg parameters. The C++ object also has the methods `forward` and `backward` for executing the respective kinematic transformations.

Using C++ ensures the real-time capability of the program, where a look-ahead time of a few seconds can be safely maintained. For instance, at a 125 Hz frequency of communication, the program's ratio of execution time to planning time varies between 1.8 and 2.5 on the reference machine.

3.1.4 Real-Time Data Exchange

As mentioned, the RTDE controller communicates with the robot and performs the printing process. While using the RTDE communication protocol, the user can perform several tasks. The user can manipulate the robot's joints, read its TCP and joint position, and use its general-purpose digital and analog I/O registers. These parameters are continuously transmitted using a high-frequency connection reaching update rates of 125 Hz on CB-series robots and 500 Hz on e-Series robots [15]. Within the control loop of the end-effector, the UR controller has priority over the RTDE communication while allowing the robot to skip packages that interrupt its computations, which minimizes the jerk of the robot.

Additionally, to this end, the inverse kinematics step is done prior, as described in section 3.1.3, allowing greater control over the computational speed. RTDE for UR-robots is provided in C++ and Python APIs and is used in Python for this work. The API works with a URScript running on the robot, which receives the path information and governs the movement of the end-effector.

With the proposed RTDE methodology, the robot motion is controlled by transmitting the individual poses that the robot traverses as joint coordinates at a fixed transmission rate (z.B. 125 Hz). The robot is moved to the pose just transmitted in time until the subsequent transmission (0.008 s at 125 Hz). Accordingly, the speed at which the distance of each successive pose describes the robot moves. Thus it is possible to influence the position and speed of the robot in each of the time intervals, either to insert an unscheduled stop or to react to deviations based on sensor data. In addition, the corresponding data can be processed graphically and made usable for process monitoring (cf. fig. 7).

The packages transmitted are defined using a configuration XML file, with outputs from the robot defined in the `state` recipe, while inputs are defined in the `setp` recipe. The outputs `runtime_state`, `actual_TCP_pose`, `actual_q`, and `output_int_register_0` are used to monitor the robot's state, positions in Cartesian and joint coordinates, and an integer to communicate with the URScript running on the robot, respectively. Meanwhile, inputs are defined as six double registers, `input_double_register_*`, which transmit the joint positions to the URScript.

A URScript program, illustrated in Fig. 3, must be exe-

```
▼ BeforeStart
    write_output_integer_register(0, 0)
    tmp:=[0,0,0,0,0,0]
    setp = get_actual_joint_positions()
    Popup
    write_output_ integer register(0, 1)
    rtde_set_watchdog("input int register 0", 1, "STOP")
▼ Robot Program
    servoj(setp, 0, 0, 0.008, 0.1, 300)
▼ Thread 1
    setp :=get_actual_joint_positions()
  Loop
      tmp:=[0,0,0,0,0,0]
      tmp[0] = read_input_float_register(0)
      tmp[1] = read_input_float_register(1)
      tmp[2] = read_input_float_register(2)
      tmp[3] = read_input_float_register(3)
      tmp[4] = read_input_float_register(4)
      tmp[5] = read_input_float_register(5)
      x = norm(tmp)
      If x
        setp:= tmp
        write_output_integer_register(0, 0)
```

Figure 3. URScript running on the End-Effector

cuted on the robot's end for the system to receive and apply the transmitted waypoints. The program starts with initializing variables and waits for a pop-up message box to be closed by the user. Then the execution process starts as soon as the RTDE communication from the Python code starts. After that, the script continuously executes `servoj` along with a thread in which it sets the joint position `setp` to the `input_double_register_*`, read by the function `read_input_float_register`. `servoj` is a real-time control function of joint positions, which blocks the computation for a specified parameter t, determined by the frequency of the communication. It uses `look-ahead time` and `gain` to predict the future trajectory [16]. If changed, these parameters can further fine-tune the reaction time of the robot and the smoothness of the robot's trajectory. For this study, the standard settings recommended in the documentation were used and proved sufficient. The parameter t can be set to a value as low as 0.002 s (500 Hz) for e-Series robots but was set to 0.008 s in this study to ensure that the inverse kinematics calculations can be performed in parallel to execution (cf. section 3.1.3).

3.2 Program Execution

With each of the modules described, the overall flow and execution of the methodology are illustrated in Fig. 4. After initializing and reading the IFC file using the IFC Reader module and after initializing the RTDE communication via the configuration XML file, three threads are activated. The planning thread is responsible for discretizing the segments and layers provided by the reader into three-dimensional Cartesian coordinates. The inverse kinematics thread receives the Cartesian coordinates and calculates the joint coordinates. Finally, the communication thread relays the joint coordinates to the URScript with the specified frequency of 125 Hz to maintain real-time control. The communication thread also communicates with the extrusion module, controlled by a serial connection.

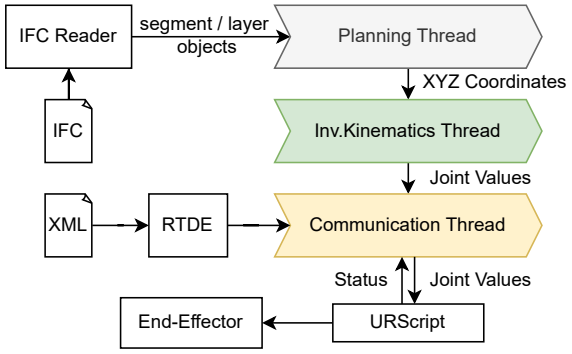

Figure 4. Execution Flowchart of the Proposed Methodology

The Planning and the inverse kinematics thread are put on hold to avoid unnecessary computation when they have calculated enough joint coordinates, defined by look-ahead time limits, and resume when the robot has caught up. Using this sequence, the segment, layer, and coordinate information can be changed and fed into the thread within or after the look-ahead time has passed. Accordingly, this also enables the user to change the layer time, extrusion rate, and other parameters depending on the results at the start of the printing process. If changes are made, these will only take effect after a few seconds (depending on the hardware) so as not to interrupt printing.

4 Experimental Validation

The presented robot control methodology has been developed for a small-scale test setup, including a Universal Robot UR10e industrial robot arm and an extrusion-based clay printing tool. The following validation shows that the developed planning and control modules can directly process FIM models into robot motion in a robust and smooth process. From the preparation of the design to the finished printed component, only a few steps are required. For controlling the robot, a regular laptop with the following specifications was connected via TCP/IP to the robot:

- i7-6700HQ 2.6 GHz Processor (8 Cores)
- 12 GB of Random Access Memory (RAM)
- Ubuntu 22.04 LTS

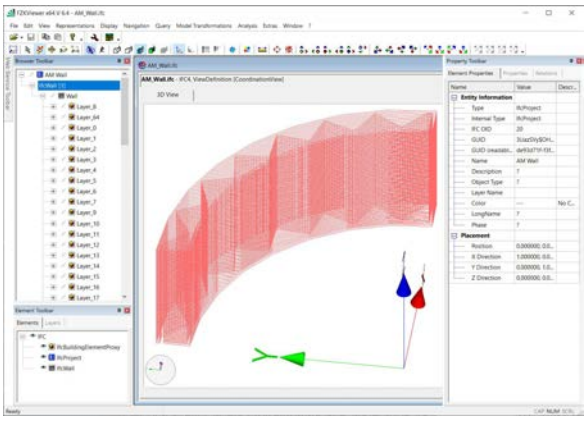

Figure 5. Fabrication Information Model used for testing the robot control method [3].

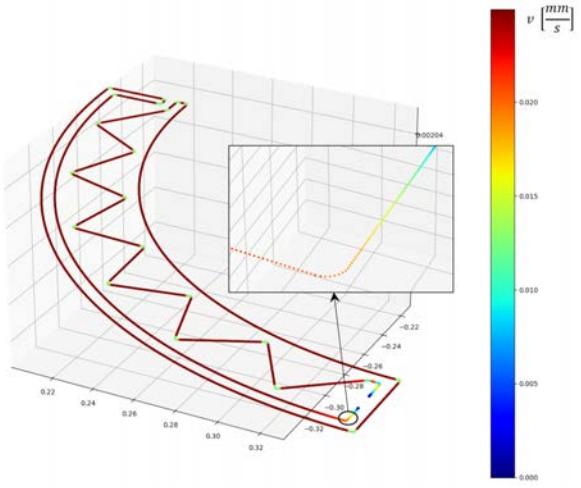

Figure 6. Velocity profile for the imported FIM model.

Using the FIM framework [3], a BIM-based design – a quarter-circular wall with a zigzag in-fill – was created to be fabricated with the clay printer (cf. Fig. 5). As a first step, a suitable velocity profile was generated for this printing path, which was limited by a maximum acceleration of $250\frac{mm}{s^2}$ and by a layer time of $70s$. As shown in Fig. 6, using a single layer as an example, the speed is reduced in regions where a rapid change of direction occurs. In addition, the slow acceleration and deceleration at the beginning and end of the printing path can be seen. The magnification shows that the path is discretized into individual waypoints, and each waypoint gets a velocity value assigned.

Before executing the clay print, the control system was tested on a simulated robot,illustrated in Fig. 7, also depicting the GUI developed for this system. All the relevant information can be seen on the GUI, and some values can be adapted during print time. Finally, the system was tested with the clay extruder, as shown in Fig. 8. The simulated robot and the actual clay print were executed using an update rate of 125 Hz, thus sending a new waypoint every $0.008s$. The resulting robot movement seemed much smoother than with an offline programmed test print using URscript, even at higher printing speeds. With the proposed system, it is thus possible to produce components faster compared to other control methods.

The RTDE control system was also successfully coupled with the extruder control system, allowing the extrusion rate to adapt automatically to the robot's speed. The general print quality of the system still needs to be assessed but will be done in a coordinated test setup. Most importantly, the system processed the manufacturing information piece by piece while controlling the robot at a 125 Hz update rate without any jerking.

5 Conclusion

At its core, the proposed RTDE control method is similar to other robot control frameworks, e.g., ROS, but it follows a more streamlined approach, which fits the overall concept of FIM. It can directly process FIM models and convert the data into robot movement without data conversion. Additionally, since the FIM data is not processed all at once before printing begins but piece by piece, the system remains adaptable if the printing path or any print parameters need to be changed. Although the path planning and robot control modules are separate in the current implementation of FIM, both systems could be run simultaneously. Any changes to the print path could be done automatically based on sensor input. Only a few steps are required to process a digital model into a finished component.

However, the main advantage of the proposed RTDE robot control method is its simplicity. Although open-source robot control frameworks are being developed to provide a common platform for controlling different robots, these systems are mainly developed for scientific purposes, i.e., mainly for lightweight robots. For example, in ROS, various Kuka robot systems are integrated up to the size KR210/150, but none of the larger robot models are supported. The control method presented in this study can always be used, provided that parameters can be transmitted to the robot via a suitable communication interface and the Denavit-Hartenberg parameters corresponding to the robot are available. For UR robots, the RTDE functionality has been demonstrated in this study. However, in the same way, it can be done for Kuka robots, via the RSI interface or the KukaVarProxy tool [14], or for ABB robots, via the OPC UA or MQTT interface.

The system is easily expandable and opens up many pos-

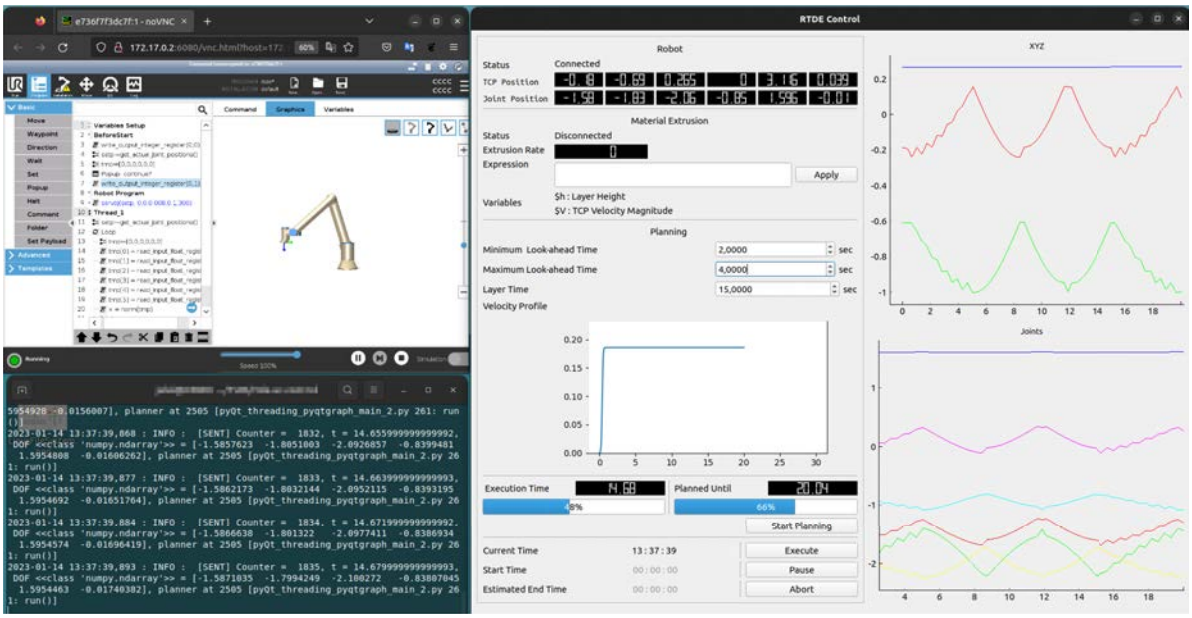

Figure 7. Using the robot control system with a simulated UR10e robot (left), showing the GUI of the system (right).

Figure 8. Clay print using the implemented robot control system. The shown clay extruder nozzle is mounted on a UR10e robot.

sibilities, such as integrating different (closed loop) control loops. For this reason, the proposed RTDE method is invaluable for setting up automatic quality control systems and thus increasing AM systems' automation level. The implementation shown in this study also provides a lightweight GUI and enables the user to interact with the printing process while it is running.

Compared to the offline control of the robot (using the URScript functionality), the printing process seems much smoother with the RTDE control, which is easily explained by the velocity smoothing at sharp corners in the print path impossible otherwise. A fact that is taken as an indication that velocity smoothing positively affects the overall print quality. However, as this study is mainly focused on robot control, the performance of the printed specimens was not evaluated quantitatively. Nevertheless, the RTDE control can be used to study the effects of speed smoothing when printing with viscous materials such as concrete or clay.

Acknowledgments

The research presented is part of the Transregio 277 'Additive Manufacturing in Construction – The Challenge of Large Scale', funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 414265976 – TRR 277.

References

[1] André Borrmann, Markus König, Christian Koch, and Jakob Beetz. Building information modeling: Why? what? how? In *Building information modeling – Technology Foundations and Industry Practice*, pages 1–24. Springer, 2018.

[2] Klodian Gradeci and Nathalie Labonnote. On the potential of integrating building information modelling (BIM) for the additive manufacturing (AM) of concrete structures. *Construction Innovation*, 2019.

[3] Martin Slepicka, Simon Vilgertshofer, and André Borrmann. Fabrication Information Modeling: interfacing building information modeling with digital fabrication. *Construction Robotics*, 6(2):87–99, 2022. ISSN 2509-8780. doi:10.1007/s41693-

022-00075-2. URL https://doi.org/10.1007/s41693-022-00075-2.

[4] R.A. Buswell, W.R. Leal de Silva, S.Z. Jones, and J. Dirrenberger. 3D printing using concrete extrusion: A roadmap for research. *Cement and Concrete Research*, 112:37–49, 2018. ISSN 0008-8846. doi:https://doi.org/10.1016/j.cemconres.2018.05.006. URL https://www.sciencedirect.com/science/article/pii/S0008884617311924. SI : Digital concrete 2018.

[5] Nicolas Roussel. Rheological requirements for printable concretes. *Cement and Concrete Research*, 112:76–85, 2018. ISSN 0008-8846. doi:https://doi.org/10.1016/j.cemconres.2018.04.005. URL https://www.sciencedirect.com/science/article/pii/S000888461830070X. SI : Digital concrete 2018.

[6] Pinar Urhal, Andrew Weightman, Carl Diver, and Paulo Bartolo. Robot assisted additive manufacturing: A review. *Robotics and Computer-Integrated Manufacturing*, 59:335–345, 2019. doi:10.1016/j.rcim.2019.05.005.

[7] Serdar Kucuk and Zafer Bingul. *Robot kinematics: Forward and inverse kinematics*. INTECH Open Access Publisher London, UK, 2006.

[8] Lieyun Ding, Ran Wei, and Haichao Che. Development of a BIM-based automated construction system. *Procedia Engineering*, 85:123–131, 2014. ISSN 1877-7058. doi:https://doi.org/10.1016/j.proeng.2014.10.536. URL https://www.sciencedirect.com/science/article/pii/S187770581401902X.

[9] Alexander Paolini, Stefan Kollmannsberger, and Ernst Rank. Additive manufacturing in construction: A review on processes, applications, and digital planning methods. *Additive Manufacturing*, 30:100894, 2019. ISSN 2214-8604. doi:https://doi.org/10.1016/j.addma.2019.100894. URL https://www.sciencedirect.com/science/article/pii/S2214860419309029.

[10] Filipe Monteiro Ribeiro, J. Norberto Pires, and Amin S. Azar. Implementation of a robot control architecture for additive manufacturing applications. *Industrial Robot: the international journal of robotics research and application*, 46(1):73–82, 2019. doi:10.1108/ir-11-2018-0226.

[11] Omid Davtalab, Ali Kazemian, and Behrokh Khoshnevis. Perspectives on a BIM-integrated software platform for robotic construction through Contour Crafting. *Automation in Construction*, 89:13–23, 2018. ISSN 0926-5805. doi:https://doi.org/10.1016/j.autcon.2018.01.006. URL https://www.sciencedirect.com/science/article/pii/S0926580517307975.

[12] Xuchu Xu, Ruoyu Wang, Qiming Cao, and Chen Feng. Towards 3D Perception and Closed-Loop Control for 3D Construction Printing. In *Proceedings of the 37th International Symposium on Automation and Robotics in Construction (ISARC)*, pages 1576–1583, Kitakyushu, Japan, October 2020. International Association for Automation and Robotics in Construction (IAARC). ISBN 978-952-94-3634-7. doi:10.22260/ISARC2020/0219.

[13] Arturo Laurenzi, Enrico Mingo Hoffman, Luca Muratore, and Nikos G. Tsagarakis. Cartesi/o: A ros based real-time capable cartesian control framework. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 591–596, 2019. doi:10.1109/ICRA.2019.8794464.

[14] M.H. Arbo, I. Eriksen, F. Sanfilippo, and J.T. Gravdahl. Comparison of kvp and rsi for controlling kuka robots over ros. *IFAC-PapersOnLine*, 53(2):9841–9846, 2020. ISSN 2405-8963. doi:https://doi.org/10.1016/j.ifacol.2020.12.2688. URL https://www.sciencedirect.com/science/article/pii/S2405896320334509. 21st IFAC World Congress.

[15] Real-time data exchange (RTDE) guide - 22229, 2022. URL https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/.

[16] Universal robots - script manual - e-series - sw 5.11, 2022. URL https://www.universal-robots.com/download/manuals-e-series/script/script-manual-e-series-sw-511/.