Visual Programming for Robot Control: Technology Transfer Between AEC and Industry

Johannes Braumann, Karl Singline, and Martin Schwab

Abstract For a long time, the construction sector has been considered a field with a low degree of digitization and automation with architects and designers looking for inspiration in other industries. Today, the construction sector is steadily innovating and automating, prompted by the lack of skilled labor. While robots are gradually starting to be used in situ for construction, robotic arms—also referred to as industrial robots—have already created new ways for the creative industries to develop innovative machinic processes at 1:1 scale. As the field of architecture eagerly moved towards robotics with an open mindset and little existing infrastructure or established protocols, architects and designers were quick to adapt the key themes of Industry 4.0 for their purposes. A core enabling factor has been the field's expertise in advanced, geometry-focused visual programming tools, which have since been adapted for robotic fabrication in order to enable individualized fabrication processes and mass customization. This chapter explores this development through several case studies and provides an outlook how visual programming and robotics may lead to a more sustainable, local, decentralized, and innovative post-industrial manufacturing in the creative industries and beyond.

Keywords Visual programming · Technology transfer · Creative industries · Robotic fabrication · Mass customization

United Nations' Sustainable Development Goals 9. Industry, Innovation and Infrastructure · 11. Sustainable Cities and Communities · 12. Responsible Consumption and Production

J. Braumann (✉)
Creative Robotics and Association for Robots in Architecture, 4020 Linz, Austria
e-mail: johannes@robotsinarchitecture.org

K. Singline · M. Schwab
Creative Robotics, University of Arts and Industrial Design Linz, 4020 Linz, Austria

M. Barberio et al. (eds.), *Architecture and Design for Industry 4.0*, Lecture Notes in Mechanical Engineering, https://doi.org/10.1007/978-3-031-36922-3_26

1 Introduction

For a long time, the construction sector has been considered a field with a low degree of digitization and automation [1], with architects and designers looking for inspiration in other industries. Today, the construction sector is steadily innovating and automating, prompted by the lack of skilled labor. While robots are gradually starting to be used in situ for construction, robotic arms—also referred to as industrial robots—have already created new ways for the creative industries to develop innovative machinic processes at 1:1 scale. Startups from the field of architecture are developing new additive fabrication technologies [2] or radically rethinking existing processes like shotcrete [3] while established construction companies innovate their fabrication workflows (Fig. 1). This innovation can partly be attributed to universities, where many architecture and design students are today exposed to those technologies during their studies.

This leap in competence, from a few key research institutions to a much wider range of users, coincided with the definition of Industry 4.0 at Hannover Fair 2011, as marked by the first conference on Robotic Fabrication in Architecture, Art, and Design, ROB|ARCH a year later in 2012.

As the field of architecture eagerly moved towards robotics with an open mindset and little existing infrastructure or established protocols, architects and designers were quick to adapt the key themes of Industry 4.0 for their purposes.

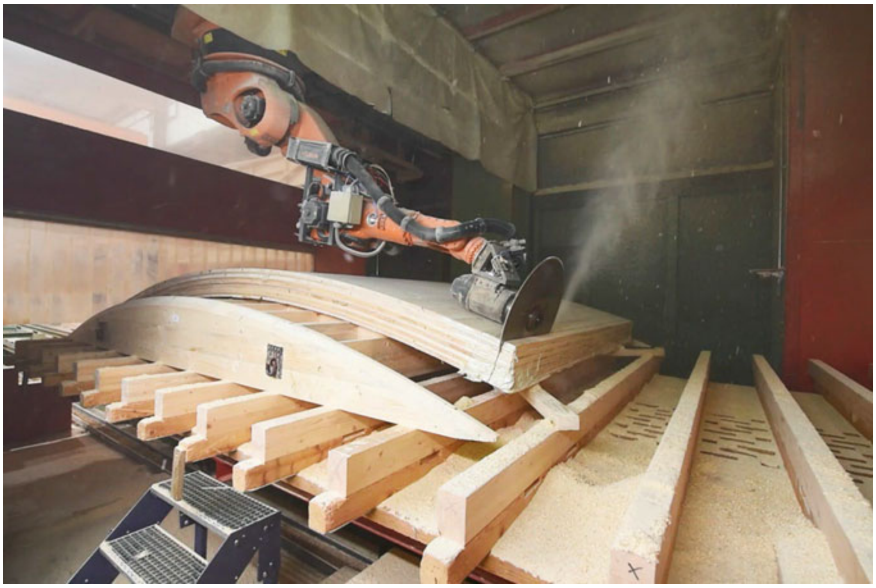

Fig. 1 Large-scale robotic fabrication process defined in a visual programming environment at ZÜBLIN Timber, Germany

This chapter traces the connection between Industry 4.0 and robotic fabrication in architecture, with a specific focus on the role of visual, flow-based programming as a driver for algorithmic thinking, resulting in innovative robotic fabrication workflows that are today also increasingly adopted by other industries.

2 Industry 4.0

2.1 Individualization

While research into robotic fabrication in architecture has been ongoing since as early as the 1980s [4] the results of that research did not reach the larger community of the creative industry but was primarily rooted in academic research and engineering applications. Now in the fourth industrial revolution, innovative architectural applications using robotic arms are no longer just trailing the perfectly orchestrated mass-fabrication in industry but matching or even exceeding the state of the art of industry regarding individualization and mass-customization.

While the "smart factory" as the overlaying idea of Industry 4.0 does not directly apply to many architectural applications, it contains a range of sub-topics that strongly resonate with the creative industries. Custom manufacturing enables individualization and small lot sizes, while digital twins facilitate process simulation and rapid (design) iterations.

An enabling factor for that development can be found in the repurposing of digital tools originating from the creative industries for automation and robotics, which allow architects and designers to apply their deep knowledge and understanding of working with small lot sizes to the—for them—still new field of robotics, particularly robotic arms, or industrial robots.

The development and repurposing of tools has become important because while individualization is of course actively used in industry for a variety of applications [5], these developments are mostly built on custom, purpose-built software, rather than accessible programming environments. Though there are commercial software packages for a variety of tasks like milling, 3D printing, 3D scanning, etc., they only cover the most common, commercially relevant robotic tasks.

2.2 Digital Twin

Currently, a main selling point of industrial manufacturing software is the digital twin, which promises to speed up the development of new production processes as well as the quality of their output by enabling an accurate simulation of entire production lines.

When looking closer at these processes, it becomes apparent that while their scale and orchestration makes them highly complex, the individual actions making up those processes are often comparably simple, moving elements from A to B, performing spot welding, or dispensing glue along a curve.

The limitations of digital twins become apparent once non-standard processes are involved that go beyond the state of the art. Even industrially applied processes like fused filament fabrication still cannot be efficiently simulated due to the complex interactions of heat and material [6]. Thus, these fabrication processes rely on their software making viable assumptions about the process parameters, along with the process expertise of the machine operator that is often the results of years or decades of working with a given manufacturing method (Fig. 2).

2.3 Collaboration

However, when working with completely new processes, no viable assumptions or simulation frameworks yet exist. In industry, interdisciplinary teams solve these challenges in a collaborative approach with each discipline contributing their expertise, from geometry to material to robotics. This approach can also be seen in the early robotics projects in the field of architecture, where mathematicians collaborated with architects to realize complex brick stacking patterns [7].

Beyond large-scale industry, it often is not feasible for smaller enterprises, especially in the creative industries, to assemble interdisciplinary teams, instead having to rely on the local process experts with a deep understanding of a given material, but less programming and robotics expertise.

Geometry-focused, visual programming like McNeel Grasshopper and Autodesk Dynamo today provides a pathway for these user groups to apply their process knowledge to a robotic process through an accessible, responsive interfaces that fosters experimentation and by design facilitates iteration and individualization.

3 Visual Programming

Today, visual or flow-based programming is often associated with “low-code” [8] strategies that make complex processes accessible to non-experts, such as allowing designers to create complex, parametric geometries [9] or non-coders to automate processes and workflows [10]. However, visual programming is today also frequently used in industry, from complex factory automation to the definition of robotic processes.

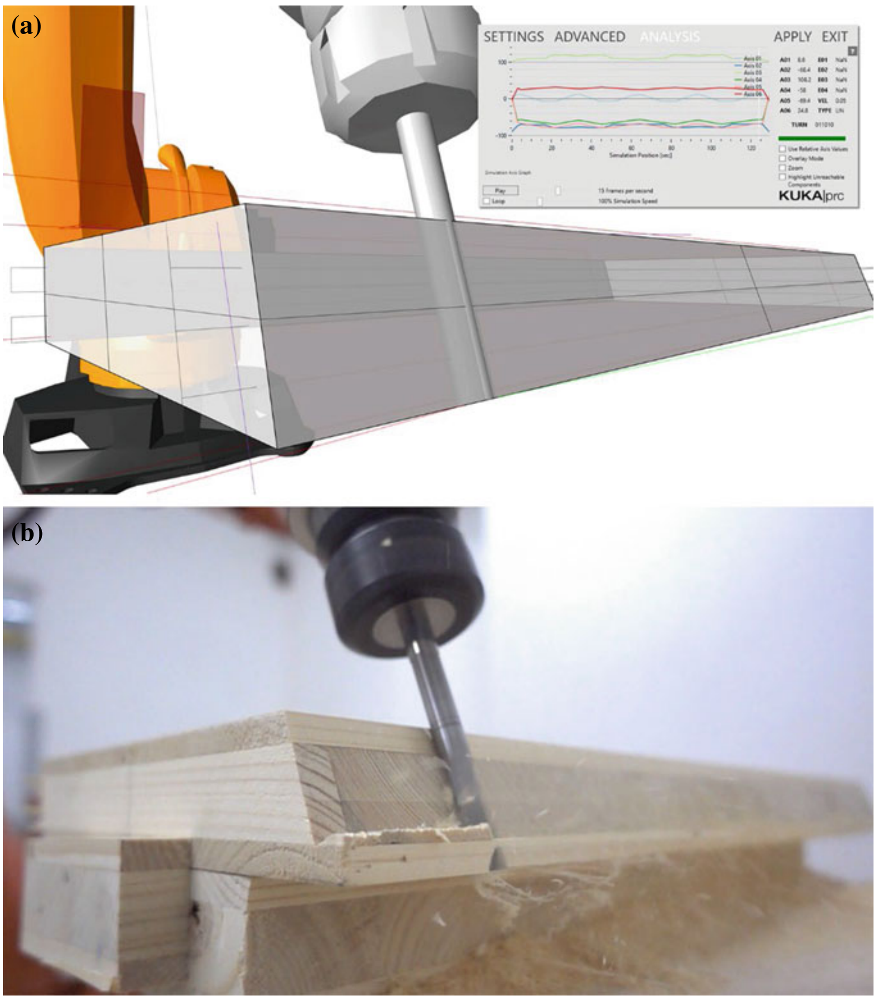

Fig. 2 Digital, idealized representation of a robotic milling process (Fig. 2a), actual complexity of timber as a cross-laminated, anisotropic material (Fig. 2b)

3.1 Visual Programming in Industry

The underlying programming paradigm of flow-based programming was first developed by Morrison [11] in the late 60 s for IBM, building upon the concept of co-routines that was sketched out even earlier [12].

The concept of flow-based programming is that the process of programming does not happen in a textual, but in a graphical way. This often results in a flowchart-like appearance, where modules, containing certain functionality or processes, are connected via lines or arrows, thus defining their parametric relationship (Fig. 3, left).

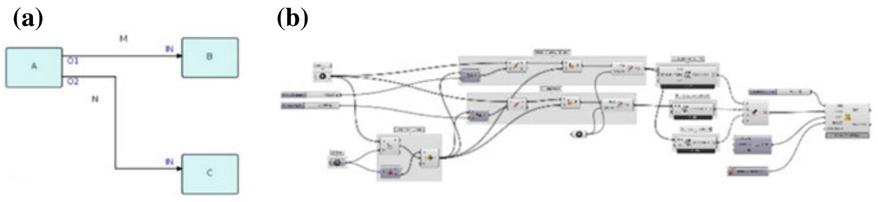

Fig. 3 Flow-based programming by Morrison dating back to the 60 s (Fig. 3a). CAD-oriented visual programming through McNeel Grasshopper (Fig. 3b)

Within the area of robotics, a basic visual programming language for PLCs is the Function Block Diagram (FBD) which was standardized as part of IEC 1131–3 [13] and allows the user to e.g. create individual functions via ST (Structured Text) and then make them more easily useable via the graphical representation as a FBD.

More recently there has been a large number of new, visual programming environments presented within the scope of industrial robots, such as drag&bot [14], Fox|Core by Faude, and the Desk software used to program Franka Emika robots, all of which are running within the browser. Within the greater area of robotic research there are also educational environments like Scratch [15] and Blockly, whose block-based visual programming has been expanded to include industrial robots [16–18].

These environments are highly flexible and can be used to program a wide variety of tasks, even allowing the easy integration of a wide array of external sensors and data sources. However, their primary use-cases can be found in the area of pick-and-place, bin picking and assembly, as they lack CAD integration and more complex geometric functionality that would be needed to e.g., derive toolpaths from an imported surface geometry.

3.2 Visual Programming in Architecture, Art, and Design

Geometry-focused approaches towards visual programming are instead often found in the creative industry. Users in the field of architecture and design utilize tools such as McNeel Grasshopper and Autodesk Dynamo to define parametric objects, artists create real-time visualization and digital art through VVVV and Max/MSP and entire video games are developed through Unreal Blueprints and Unity Visual Scripting. In the area of 3D-modelling, visual programming has been pioneered for the definition of photorealistic materials, defining how different textures are blended and linked via a node-based system, but has since been expanded to geometric operations, e.g. via Blender Animation Nodes, XPresso, Houdini and others.

A fundamental difference in these systems is whether they are real-time based or not. Programs like VVVV, but also FBDs in PLCs, are generally running in real-time, i.e., the graph is constantly being refreshed at a high rate, ideally at 60 Hz for live-visuals and potentially much more frequently for PLC-programs that might be triggered every millisecond or less, for 1000 Hz and more.

CAD-oriented programming environments like Grasshopper (Fig. 3, right) on the other hand are optimized for longer-running, more complex processes and by default not constantly updating: operations are represented as function blocks with inputs on the left and outputs on the right side, creating a directed, acyclic graph. Only when an input changes, the downstream components refresh automatically to reflect the change in input values, creating a highly reactive system that lends itself to a very intuitive interaction with data and geometry. Commonly, such function blocks include geometric operations like e.g., creating a point out of three numeric values representing its XYZ coordinates, but through plugins like KUKA|prc, HAL, RoboDK, and Robots, the range of function blocks can be expanded to include robotic fabrication, thus for example defining the robot's programmed position through a coordinate system.

3.3 Towards Robotics

Since 2007 Grasshopper has established itself as the core platform for applications utilizing industrial robots within the field of architecture and design. There are several advantages of using such a platform for robotic processes: Geometry and toolpath strategies can be generated at the same time, so that it is possible to intelligently have the geometry respect certain fabrication-related parameters, while the fabrication-related data can be automatically assigned to the relevant geometric features. This automatically translates both into the potential to be used for the automated batch-generation for mass-customization, as well as providing immediate feedback on the feasibility of the developed fabrication process, thus speeding up the learning process for new users. Furthermore, as the user is defining their own fabrication strategy, the potential scope of robotic fabrication is expanded beyond the functionality offered by CAM software such as milling, wire-erosion and plasma cutting.

3.4 Limitations

However, going beyond the state of the art also leads to an increased complexity for the user, as the development of such strategies requires both in-depth robotic knowledge, as well as material expertise and proficiency with the visual programming environment. Unlike many CAM environments, visual programming frameworks generally offer little official documentation and training, instead relying on third party and community-led initiatives.

A related general criticism of flow-based visual programming in Grasshopper by Davis et al. [19] is the lack of modularization to facilitate efficient code re-use as well the infrequent use of clearly named parameters.

Integrated development environments (IDE) for text-based programming like Microsoft Visual Studio support the user in creating a clear and easily readable syntax, often offering refactored code automatically. This process is based on consistent coding conventions for programming languages, which do not yet exist for visual programming.

Finally, especially within the scope of robotic fabrication, a limitation of CAD-oriented flow-based programming environments is that while they are very efficient in defining data flows, they are less optimized for process flows, i.e., conditional clauses, parallel processes [20]. This can be especially an issue for flexible fabrication processes that incorporate sensors and feedback loops (see Sect. 4.2).

4 Robotic Workflows and Dataflows

4.1 Defining a Robotic Process

A core appeal of using robotic arms lies in the abstraction of complexity. Rather than having to design and fabricate a bespoke, complex machine, robotic arms instead allow users to deploy an extremely reliable, well-tested and readily available manipulator, that then must be equipped with a suitable tool to perform a given task. The robot therefore forms the basis of a larger, robotic setup (Fig. 4). While they may not reach the level of performance of a purpose-built machine [21], their capabilities generally exceed the tolerances required at construction sites.

Their programming consists mostly of a sequence of movement commands and IO operations, structured by conditionals and logical expressions. A movement command defines where to move in relation to the current position of the robot,

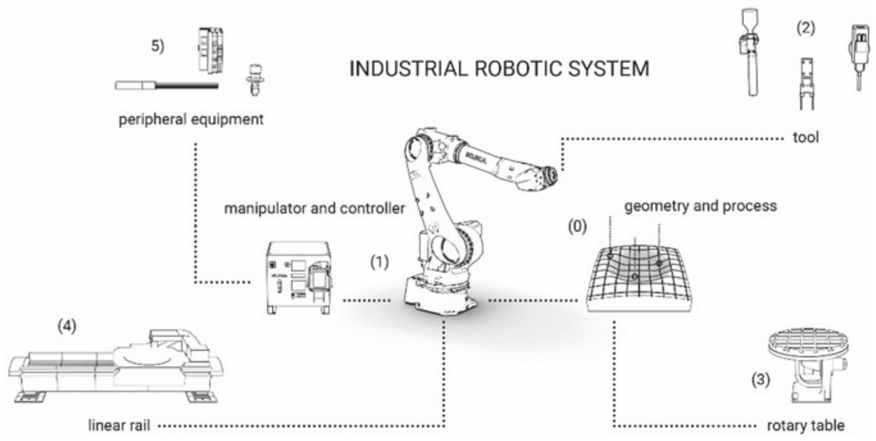

Fig. 4 Overview industrial robotic system—schematic layout for robot-based fabrication

the speed of the movement, and how to interpolate between the current and subsequent position. The robot's position can either be expressed in axis values, with a numerical value corresponding to each of its degrees of freedom, or in a Cartesian format, expressing a coordinate system through an XYZ coordinate and three Euler angles ABC in the case of KUKA robots. Common interpolation methods are linear interpolation, that moves in a straight line, circular interpolation that creates an arc through an auxiliary point, and point-to-point interpolation that interpolates at the axis level, creating a highly efficient trajectory.

4.2 Robotic Fabrication Through Visual Programming

In a visual programming environment, these movements can be represented by individual nodes. In most robotics-focused, accessible programming environments, the axis values or Cartesian position of the robot's tool is extracted from the current position of the—simulated or real—robotic arm. Therefore, the robot is moved, the position saved, and then the process repeated until the final program exists—similar to teaching by demonstration [22] done on a physical robot, but through the flow-based programming with an easier control over the structure of a program.

Within a geometry-focused visual programming environment like Grasshopper, the toolpath logic can be extracted from the underlying geometry: A NURBS curve is projected onto a free-formed surface and then segmented into a series of point objects. Based on the parametrization of the surface closest to each point, a coordinate system is defined. Along with a numerical value for the movement speed, each coordinate system results a linear movement, that is then traced by the robot, moving along the given surface while keeping the tool axis perpendicular to it.

To achieve a reliable simulation, the entire robotic setup must be known: As there are barely any standardized tools for industrial robots, calibration of each tool is required so that the robot is aware of the position and orientation of the tool center point in relation to the robot's flange. Similarly, local coordinate systems are defined in relation to the robot's world coordinate system—commonly at its base—to define the program origin and orientation. This modularity can also be well represented in a visual programming environment, coupling the robot node with different tools or external equipment to define the so-called robot cell. The typical setup for robot-based manufacturing shown in Fig. 4 consists of the product and associated geometry and process parameters (0), the industrial robot unit consisting of manipulator and controller (1), the end effector—with tools like gripper, extruder or spindle (2), a static or dynamic base for the workpiece—e.g. a rotary table for increased reachability (3), a static or dynamic robot foundation—e.g. a linear rail for expanding the working volume (4) and peripheral elements like industrial control systems, sensors, actuators and safety equipment (5).

4.3 Dataflows for Robotic Fabrication

Having both the accurate robotic setup and the parametric design within a single environment allows architects and designers to constrain the toolpath generation to parametric geometry, automatically creating an individual robot control data file for each design variation and thus fostering individualization.

Thus, a robotic process in a geometry-focused visual programming environment commonly consists of three parts (Fig. 5): The generation of the parametric geometry, the extraction of toolpaths, most commonly as sets of coordinate systems, and finally the robotic simulation and code generation. This sequence forms a directed graph where the user can easily interact with a complex, parametric system that covers both design and fabrication. Once a process is free of collisions and other problems, the resulting file can be copied to the robot, or streamed to the robot in real-time.

As geometry-focused visual programming environments create acyclic, directed graphs, they are best suited for bringing parametric designs to fabrication. However, fabrication processes that incorporate sensor feedback or user interaction are challenging to represent within such a system, as it becomes necessary to differentiate between data flow and process flow. This can be achieved implementing behavior models like state machines. Recent projects have implemented Unity Visual Scripting for that purpose as it differentiates between flow-graphs—creating similar graphs like Grasshopper—and state graphs—controlling the process flow from one state to another, e.g., informed by user interaction [20].

Due to the flexibility of the paradigm of visual programming, it has become the predominant way of programming applications that go beyond the state of the art in robotic fabrication in the creative industry, at academia and startups as well as within the established industry.

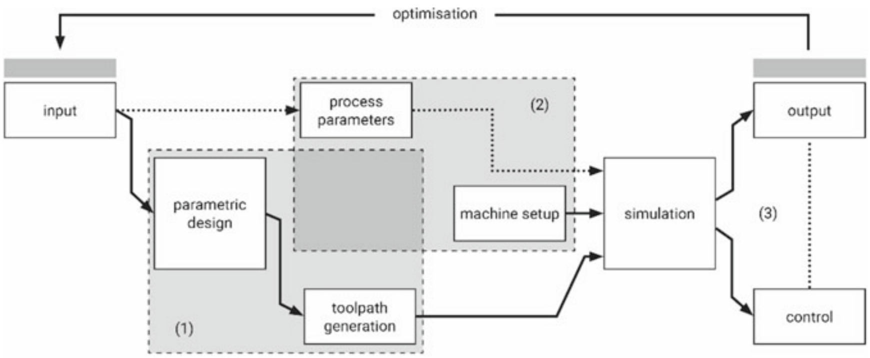

Fig. 5 Schematic workflow diagram—visual programming setup for robot-based fabrication: (1) parametric design and toolpath generation (2) digital representation of physical machine setup and process parameters (3) simulation, code generation and manual interpretation

5 Case Studies

This section provides an overview of innovative robotic applications within the field of architecture, where visual programming has become a core enabling factor for the realization of innovative processes. From large-scale, multinational construction companies to architectural offices, innovative construction startups and the wide dissemination at maker spaces.

5.1 Architecture and Design Office: Matter Make, Malta

Located on the island country of Malta, holistic design, and fabrication studio Matter Make identified the complexity of operating within the disconnected region regarding the lack of easy access to building materials (Fig. 6). This absence of access to a singular raw material led them to reconsider traditional architectural and interior design techniques and incorporate a workflow built upon a flexible visual programming as a core component of their working environment focused on complete processes based on the products available for import.

This approach prompted Matter Make to add value through means of design creativity that was supported by the novel workflow and flexibility of an industrial robotic arm which allowed a range of materials and processes. This meant projects could differ vastly between materials such as copper to plywood to high density foam without requiring additional software or equipment.

Within their visual programming environment, the same digital model is created and evolved from initial conception through to finalized design form can feature layered data of varying complexities which is used for a multitude of tasks such as quantifying a bill of materials, while simultaneously generating toolpaths required to produce their designs robotically.

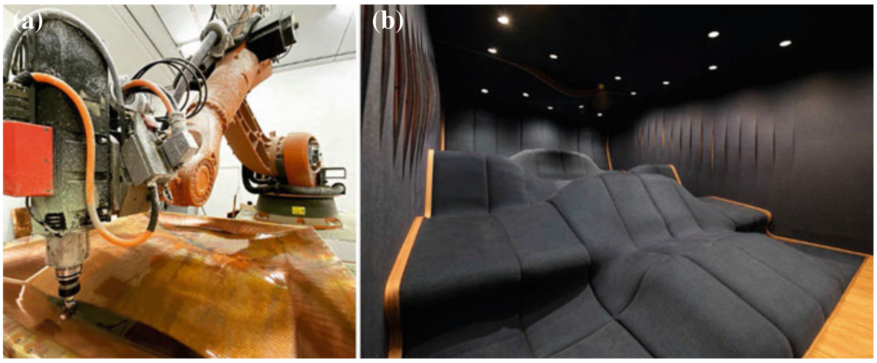

Fig. 6 Robotic incremental sheet forming of copper (Fig. 6a), free-formed home theatre by Matter Make (Fig. 6b)

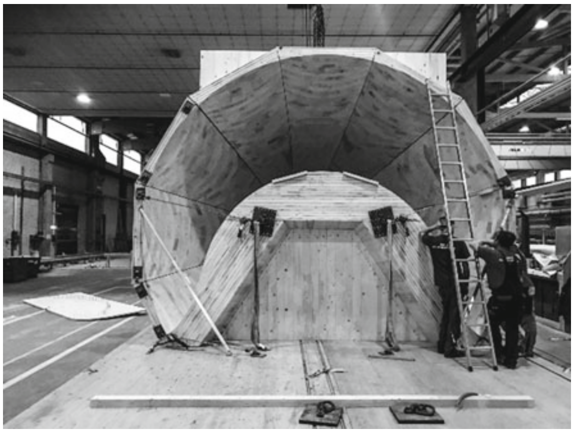
Fig. 7 Large-scale robotically fabricated timber formwork for free-formed concrete columns

5.2 Large-Scale Construction Company: ZÜBLIN Timber, Germany

ZÜBLIN Timber is a leading timber construction company based in Germany that has worked on high-end projects such as the Metropol Parasol in Sevilla and Stuttgart 21 in Germany (Fig. 7). A pioneer in robotic fabrication in architecture, ZÜBLIN Timber set up their first robotic timber fabrication setup already in 1995 and has since upgraded it to be able to cover 160m^2 with a large robot and two linear axes.

ZÜBLIN Timber uses visual programming in-house for both general production preparation and the definition of robotic processes. This facilitates the integration of advanced algorithms coming from architects and fabrication consultation offices and greatly streamlines the workflow from design to production: Where parametric geometry is often reduced to plan drawings that are then manually processed at the fabricator, ZÜBLIN Timber can keep the entire process within a single environment.

5.3 Construction Startup: REPRECT, Austria

Austrian based start-up, Reprect, is a research and development group with a focus on innovating precast concrete through adaptation of novel technologies. In 2020 Reprect explored the challenges and advantages of that the introduction of an industrial six-axis robotic arm could be for digital fabrication across multiple aspects of precast concrete manufacturing techniques. This was completed over several steps: research into existing workflows for traditional precast concrete fabrication, identifying key areas where automation could play a significant role and frameworks that would be required to support such an integration.

A significant challenge was the countless variables introduced when handling custom building components, rarely were two precast elements identical. To ensure the workflow was feasible, it had to accommodate precast panels of various dimensions, penetrations, and modifications: horizontal and vertical drilling, surface milling and contouring. The visual programming environment Grasshopper was used for its ability to process traditional CAD data that was combined with domain-specific meta-data in the form of a JSON file. Based on that, toolpaths are calculated to provide visual feedback in the form of a full robotic simulation almost immediately. This research resulted in a software solution was designed to allow for the growth of projects without a requirement of significant retooling.

5.4 Education

Educational institutions take up a central role in the popularization of new technologies. Within the field of construction robotics, there are now dedicated Master programs at several universities such as RWTH Aachen, ICD Stuttgart and ETH Zurich where visual programming is used as a core tool to get students exposed to robotics.

A different approach is to specifically use robotics as a cross-sectional, interdisciplinary tool that can be applied in a wide variety of creative disciplines. At the research department Creative Robotics in Linz, Austria, students from a variety of programs such as architecture, industrial design, fashion, and interactive media take robotic courses where they are encouraged to apply robotic technologies within their own field of expertise.

Visual programming environments like Grasshopper and Unity Visual Scripting are therefore taught to students with widely varying degrees of experience in working with CAD software and digital software tools. Through that, robotics and visual programming put together become an interface that encourages interdisciplinary collaboration and experimentation, that has already resulted in several startups, such as Print-a-Drink for robotically fabricated cocktails and YOKAI Studios for innovative textile processes (Fig. 8, left). Research-led teaching also contributed to experimental projects like a mobile 3D-printing platform that was exhibited at the Ars Electronica Festival (Fig. 8, right).

That process is facilitated by the close collaboration of Creative Robotics with the Grand Garage, Europe’s largest maker space, embedding academic research in a semi-public environment that encourages interdisciplinary thinking.

5.5 Case Study Summary

In summary, the collated projects described above demonstrate the discrete capabilities of incorporating visual programming and automation techniques already utilized in industry 4.0 within a contrasting range of architectural and related fields, from education to small bespoke design and large-scale fabrication for construction.

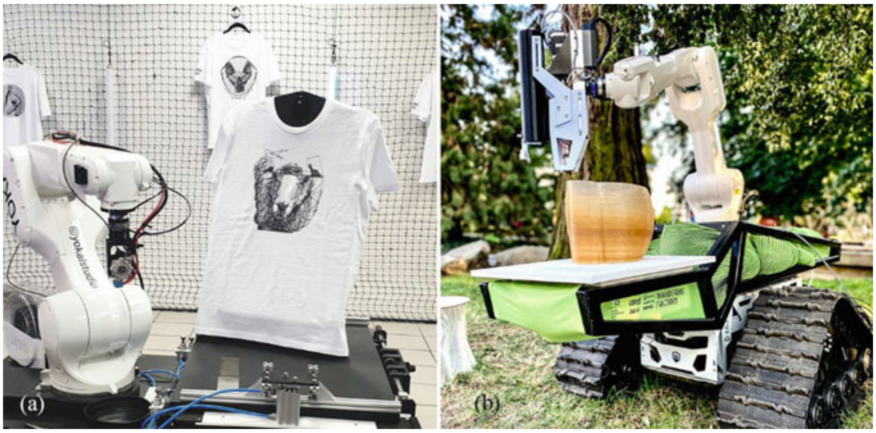

Fig. 8 Education resulting in student-led startups, like YOKAI Studios in the field of fashion and textiles (Fig. 8a), research-led teaching for the "Wandering Factory" at Ars Electronica Festival (Fig. 8b)

Although each case study is operating at a distinct scale and output, all actively demonstrate capabilities for non-standardized and mass customization construction methods facilitated by robotics that was previously deemed unavailable due to economic and efficacy constraints [23].

6 Outlook

The field of architecture has taken a pioneering role in the rapid adoption of robotic technology, questioning established standards and through that realizing advanced applications that have also caught the eyes of industry. While the ultimate goal of an automated construction site still lies in the future, current projects with industrial robots have moved beyond academic research, towards creating the foundation for innovative startups in the fields of architecture and design.

An enabling factor for this process has been the critical mass of users within architecture and design, often in academia, who have laid a foundation of knowledge for new users to build upon. Achieving this critical mass can at least partly be attributed to the development of domain-specific visual programming tools that allowed architects and designers to build upon their existing knowledge of parametric and generative design but expand it to cover also robotic fabrication. This can be seen as a testament to the efficiency of peer-to-peer teaching for knowledge transfer between different fields.

Other fields in the creative industries, such as fashion and textiles or crafts, are also starting to increasingly adopt digitization and concepts of Industry 4.0. However, their fundamental knowledge as well as their goals and the scale and valorization of their output differ significantly from architecture and design. It will therefore be necessary to adapt and modify the workflows and dataflows developed by architects and designs, towards again creating bespoke, domain-specific tools that have the potential of greatly accelerating the adoption and integration of new technologies.

The underlying technologies for simulation and control can be maintained, requiring mostly the interaction metaphors and concepts, including the degree of abstraction, to be adapted to new user groups. Building on the current approach of linking functional elements that often represent only basic geometrical methods, reduced workflows can be developed that group functionalities (Fig. 9) and automatically assign data to the correct inputs and outputs, guided by intelligent assistant systems. Having a clearly defined data structure opens possibilities for an accessible, automated optimization of robotic process parameters that can consider the individual degrees of freedom provided by each production process.

This reduction in complexity and scope enables new users to focus on the domain-specific input and associated process parameters, while providing them with a pathway to extend their level of control over a process by interacting with the underlying building blocks.

Ultimately, robotics combined with powerful and accessible tools for robot programming offers startups the potential to create highly innovative, disruptive applications, but also gives the much wider field of smaller architectural offices the possibility to take control of the fabrication process with the potential to realize high-end construction processes that strongly incorporate individualization, thus bridging the gap to much larger architecture and construction firms that employ specialized, interdisciplinary teams to fabricate their designs.

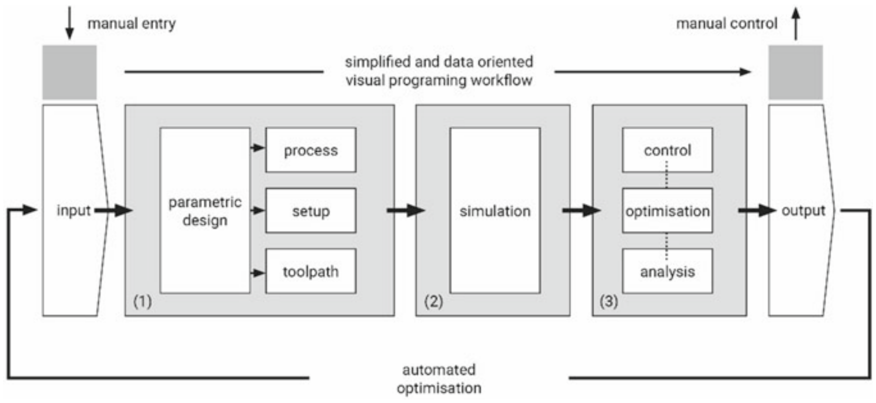

Fig. 9 Outlook for data-based workflow approaches: design-based geometry and process parameter generation (1) process related simulation (2) computer-aided analysis and automated optimization for design and process adjustment (3)

When looking at the impact of digital fabrication combined with visual programming on a higher level, the increasing accessibility of customization and multi-level-optimization leads to a potential reduction of energy consumption and use of resources whereas the low system costs enable and support a return to local and decentralized production.

By combining high flexibility with scalability, we expect individualized robotic fabrication informed by visual programming to support the creation of sustainable and innovative post-industrial manufacturing, in architecture and beyond.

References

1. IFR: World Robotics 2018—Industrial Robots. (2018)
2. Branch Technology. https://www.branch.technology. Last Accessed 11 June 2021
3. Aeditive—Revolutionäre Effizienz im Betonbau. https://www.aeditive.de/en/. Last Accessed 11 June 2021
4. Bock, T.: Innovationen im bauwesen: Roboter auf japanischen Baustellen. Bauingenieur. **63**, 121–124 (1988)
5. Da Silveira, G., Borenstein, D., Fogliatto, F.S.: Mass customization: Literature review and research directions. Int. J. Prod. Econ. **72**, 1–13 (2001). https://doi.org/10.1016/S0925-5273(00)00079-7
6. Rashid, A.A., Koç, M.: Fused filament fabrication process: a review of numerical simulation techniques. Polymers (Basel). **13**, 3534 (2021). https://doi.org/10.3390/polym13203534
7. Bärtschi, R., Knauss, M., Bonwetsch, T., Gramazio, F., Kohler, M.: Wiggled brick bond. In: Ceccato, C., Hesselgren, L., Pauly, M., Pottmann, H., and Wallner, J. (eds.) Advances in architectural geometry 2010, pp. 137–147. Springer, Vienna (2010). https://doi.org/10.1007/978-3-7091-0309-8_10
8. Fryling, M.: Low code app development. J. Comput. Sci. Coll. **34**, 119 (2019)
9. Mamou-Mani, A.: Structural innovation through digital means: Wooden waves, galaxia, conifera, sandwaves, polibot, silkworm. http://mamou-mani.com. Last Accessed 27 April 2022
10. Brell-Cokcan, S., Braumann, J.: Industrial robots for design education: robots as open interfaces beyond fabrication. In: Zhang, J., Sun, C. (eds.) Global design and local materialization, pp. 109–117. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38974-0_10
11. Morrison, J.P.: Flow-based Programming: A new approach to application development. J.P. Morrison Enterprises, (2010)
12. Conway, M.E.: Design of a separable transition-diagram compiler. Commun. ACM. **6**, 396–408 (1963). https://doi.org/10.1145/366663.366704
13. Maslar, M.: PLC standard programming languages: IEC 1131–3. In: Conference Record of 1996 Annual pulp and paper industry technical conference, pp. 26–31. (1996). https://doi.org/10.1109/PAPCON.1996.535979.
14. Naumann, M., Wegener, K., Schraft, R.D., Lachello, L.: Robot cell integration by means of application-P'n'P. In: In: Proceedings of ISR 2006. (2006)
15. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y.: Scratch: programming for all. Commun. ACM. **52**, 60–67 (2009). https://doi.org/10.1145/1592761.1592779
16. Mateo, C., Brunete, A., Gambao, E., Hernando, M.: Hammer: An android based application for end-user industrial robot programming. Presented at the (2014). https://doi.org/10.1109/MESA.2014.6935597

17. Trower, J., Gray, J.: Blockly language creation and applications: Visual programming for media computation and bluetooth robotics control. In: Proceedings of the 46th ACM Technical symposium on computer science education, pp. 5–5. (2015)
18. Weintrop, D., Afzal, A., Salac, J., Francis, P., Li, B., Shepherd, D., Franklin, D.: Evaluating CoBlox: A comparative study of robotics programming environments for adult novices. Presented at the (2018). https://doi.org/10.1145/3170427.3186599
19. Davis, D., Burry, J., Burry, M.C.: Understanding visual scripts: Improving collaboration through modular programming. Int J Archit Comput. **9**, (2011). https://doi.org/10.1260/1478-0771.9.4.361
20. Braumann, J., Gollob, E., Bastan, A.: Towards AR for large-scale robotics. In: 2022 IEEE conference on virtual reality and 3d user interfaces. Christchurch, New Zealand (2022)
21. Perez, R., Gutierrez Rubert, S.C., Zotovic, R.: A study on robot arm machining: advance and future challenges. In: Katalinic, B. (ed.) DAAAM proceedings, pp. 0931–0940. DAAAM International Vienna (2018). https://doi.org/10.2507/29th.daaam.proceedings.134.
22. Biggs, G., MacDonald, B.: A survey of robot programming systems, vol. 10
23. Apolinarska, A.A., Knauss, M., Gramazio, F., Kohler, M.: The Sequential Roof. In: Advancing wood architecture. Routledge (2016)