Sigrid Brell-Cokcan :author
Johannes Braumann
II Architects int :organization
Vienna University of Technology
Austria :country

A New Parametric Design Tool for Robot Milling

This paper proposes the use of parametric design software, which is generally used for real-time analysis and evaluation of architectural design variants, to create a new production immanent design tool for robot milling. Robotic constraints are integrated in the data flow of the parametric model for calculating, visualizing and simulating robot milling toolpaths. As a result of the design process, a physical model together with a milling robot control data file is generated. :abstract

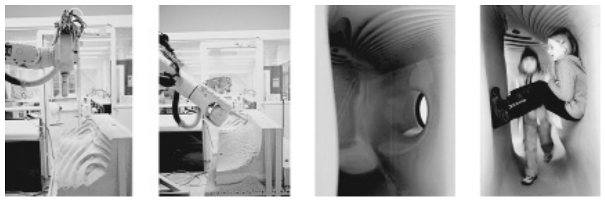
Figure 1. Robot milling for freeform surface design at TU Vienna: a 1:1 prototype is rough-cut (left), fine-cut (middle-left), finished surface (Polyurethane) (middle-right), in use (right).

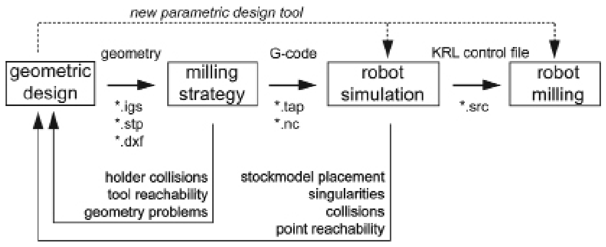

Figure 2. Existing CAD – CAM – Robot Workflow.

1 Motivation

Industrial robots have lately fascinated architects. Robots are multifunctional machines for mass-production: they can load, unload, deburr, flame-machine, laser, bond, assemble, inspect, sort, and mill, to name but a few of their functions. And, as seen e.g., in projects from Gramazio & Kohler, robots can even execute "multiple and varied tasks to create unique and carefully crafted objects" *(Edgar, 2008)*.

Gramazio & Kohler´s bricklaying robot does not just replace manual work but it provides the architects with the possibility to alter their design parametrically to produce e.g., an "informed wall" (Bonwetsch et al., 2006). Unlike robots, Computer Numerically Controlled (CNC) laser cutters, 3D-printers, and CNC-milling machines are already state-of-the-art manufacturing methods in architectural building workshops, architecture schools, and architectural offices *(cf. Iwamoto, 2009; Kolarevic, 2001,2005; Schodek et al., 2004; Scheurer 2008)*.

One of the major reasons why milling robots (Figure 1) are hardly used in education or architectural offices lies in the geometric complexity and difficulty of controlling the kinematics of a CNC-robot *(cf. Pottmann et al., 2007; Spong et al., 1988)*. Further reasons for the thus far limited use of milling robots in architecture are the complex workflow (discussed in Section 2) or the necessity of controlling the robot directly via on- or offline programmed toolpaths, which we argue is inappropriate for a dynamic architectural design environment. Most architects find it limiting to their work process that machine constraints must be considered with on- or offline programming, teaching, or scripting right at the beginning of an architectural design.

The goal of our ongoing research at TU Vienna is to create an easy-to-use design tool via data flow programming to dynamically control a Kuka KR60H milling-robot and to integrate production immanent constraints into a dynamic design environment with a maximum of manufacturing and aesthetic design control. In this paper, we propose interactive, production immanent tools that still allow the user to dynamically "explore" design variations throughout the whole design-to-fabrication process.

2 Existing Workflow

The following design-to-production workflow (Figure 2) is provided for robot milling applications (in our case for the Kuka KR60H) and can be described with the following steps:

2.1 Geometric Design in Computer-Aided Design Software

Usually, geometric models (parametric or not) are generated in a Computer Aided Design (CAD)-system and then exported to Computer Aided Manufacturing (CAM) software via, e.g., STEP, IGES, or DXF data exchange files. These geometric models do not automatically incorporate manufacturing constraints. Thus far, the creation of a millable 3D-geometric model has depended mostly on the experience of the designers: they must consider manufacturing relevant properties such as the right scale of the model, the tool and point reachability, or possible undercuts.

2.2 Milling Strategy in Computer-Aided Manufacturing Software

The imported geometric model (either as Non-Uniform Rational B-Splines (NURBS) or as meshes) is first checked for inconsistencies, bad faces, overlapping surface patches, or surface discontinuities. The stock model size can be generated automatically by defining a bounding box or referencing geometry. Toolpaths are generated by choosing milling strategies. (See also Section 4.3) Problems of the geometric model as outlined above,

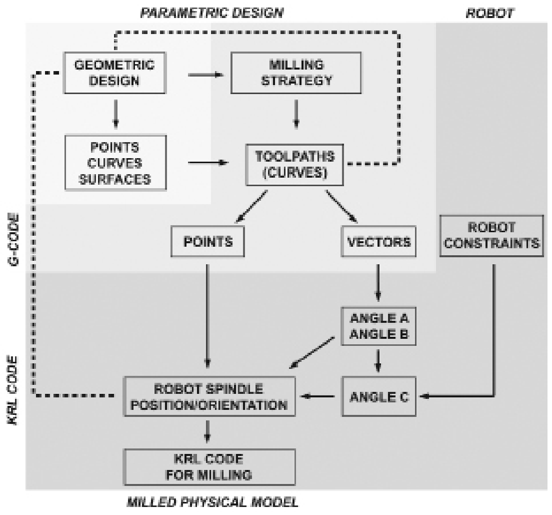

Figure 3. Robot milling data flow in Grasshopper.

unreachability of the toolpath, or tool/model collisions very often force the user to reshape the model in the geometric design step (2.1).

While CAM software is not applicable for geometric design, its advantages are the automated tool collision evasion and the possibility of simulating toolpaths and material removal. Poor support for robotic milling is a disadvantage of current CAM software: collision control and simulation feedback are only provided for the tool and the geometrically defined tool holder. The output of the CAM Software is the so called "G-Code", a numeric control data file for CNC milling machines. The robot and the robot´s kinematics are neither considered in the G-code nor in the milling simulation.

2.3 Post-processing the G-Code for Simulating the Robot's Kinematics

To simulate toolpaths and robot movements, so-called postprocessors are available from robot manufacturers. These postprocessors import G-code (see Section 3.1) and check the movement and positions of the robot´s end effector for unreachable points, collisions, and singularities which would all cause an error in the robot´s kinematics. A singular position means that two or more joints no longer independently control the position and orientation of the end effector *(see also Baker & Wampler 1988)*.

Often wrong positioning or incorrect orientation of the workpiece in the robot´s workspace or unreachable points of the toolpath force the user to go back to the geometric CAD-model and reconsider the initial design. After simulating the milling process within the postprocessor, a KRL (KUKA Robot Language) file can be written and executed at the robot.

The lack of a toolpath design according to the robot's constraints and the accurate simulation of the whole robot milling process, which is only available at the very end of the workflow, interfere with a fluent design process.

2.4 Production Immanent Modeling

In our previous research (Brell-Cokcan et al., 2009), the outlined workflow was optimized through eliminating the possible re-design for manufacturing (looping between Section 2.1 and 2.2) by using production immanent parametric modeling (see also Sakamoto et al. 2008). This strategy implements the relevant production constraints such as tool length, tool diameter, and stock model size as design parameters in the generative modeling tool. One advantage of using a parametric modeling environment such as Grasshopper (Section 4.1) is the possibility of tracking aesthetic design and manufacturing feasibility via a real-time preview. In an earlier design interface, only the tool parameters were controlled, but the CNC-machine--in our case the Kuka robot--was not considered.

To recapitulate the workflow outlined above, the inclusion of the robotic system in a more sophisticated design tool is necessary.

3 Improved Workflow

To allow for seamless design and robot control, the development of a new design tool, where a parametric geometric model is directly linked to the robot's constraints, is vital (Figure 3). The following parameters are integrated: tool properties (type, length, diameter) with dynamic geometric repositioning of the end effector, stock model size, geometric position in the robot´s workspace, and feeding speed. The robot´s kinematics between axis A1 and A6 can be neglected in the generated design tool due to the subsequent automatic calculation by the robot control unit during the milling process.

3.1 G-Code

The most common way of creating a G-code is by the use of CAM-software (see also Section 2.2). As most machines cannot deal with spline curves, the toolpaths are approximated by a series of straight line segments and circular arcs and formatted in the machine specific G-code syntax. Below is a single line of G-Code in a five-axis milling process:

```
N20X148.867Y188.395Z-1.739I-0.54361197J0.04997284K0.8378476
8F5000
```

N20 marks the 20th line of the code, followed by the X, Y, and Z coordinate of a point p in a predefined Cartesian coordinate system. The IJK values define a unit vector, which gives the direction of the tool axis at the point p. F5000 sets the feeding speed to 5000 millimeters per minute. Most machines are able to directly load the G-code and immediately start the milling process. However, an industrial robot with complex kinematics requires more spatial information than just a point and a vector–at least the end effector has to be clearly defined in three-dimensional space, which is not possible in G-code.

3.2 KUKA Robot Language

To solve the robot´s kinematics, it is possible to use a postprocessor, as described in Section 2.3. The conversion of CAM-software generated G-code to executable KRL-code is not an automated process, but requires multiple user input to import G-code files, to define the tool and coordinate system, to choose an appropriate kinematic strategy and to verify the kinematic simulation. An alternative to using a postprocessor is the direct generation of KRL-code in Grasshopper (refer to Section 4). The resulting KRL-code can then be executed with the KUKA robot.

```
LIN {X 539.459,Y 990.285,Z -60.006,A 8.14713,B -89.64397,C -63.14729}
C_DIS
```

Again, XYZ are the coordinates of a point p in 3D-workspace. However, A, B, and C do not define an orientation vector, but yaw, pitch, and roll angles. Similar to computer graphics, these angles enable the user to state the exact orientation of an object in space: the A-angle is equivalent to a rotation in the XY-plane around the Z-axis; the B-angle states the inclination of the end effector, while the C-angle defines the rotation

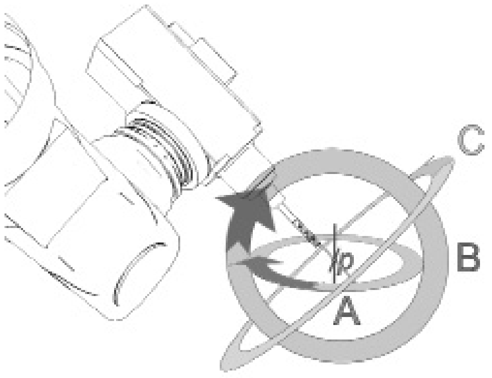

Figure 4. End effector orientation according to yaw (A), pitch (B) and roll (C).

along the axis of the end effector (Figure 4). LIN stands for "linear movement"–prompting the robot's tooltip to move towards this position with linear interpolation. Opposed to linear movements are PTP (i.e. Point-To-Point) movements, where the robot tries to maneuver from one point to another one with the least amount of rotation on any of its axis.

When KRL-code is executed at the robot´s terminal, the remaining kinematics is solved by the robot in real time, without requiring any further input.

4 Implementation of a Production-Immanent Design Tool

4.1 Parametric Design

Grasshopper (GH) is a data-flow programming plug-in for the CAD-software Rhinoceros (McNeel) written by David Rutten. It allows the user to create a directed acyclic graph out of pre-defined or custom made components. This is done in a visual way by pulling wires from the output of one component into the input of another component, allowing for a very intuitive approach to scripting procedures. A feature which sets the Grasshopper approach to programming apart from conventional scripting (e.g., Rhinoscript) is the constant real time visualization. Changes to referenced geometry or to the history tree itself are reflected in a preview window without delay, enabling the user in turn to react to those changes, without having to reset or restart the script.

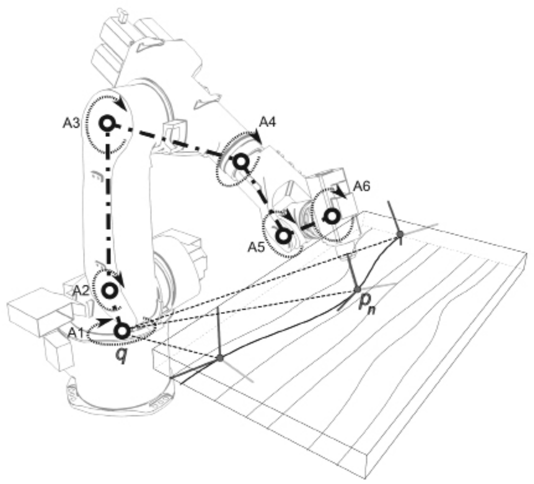

Figure 5. End effector orientation towards the robot's base point.

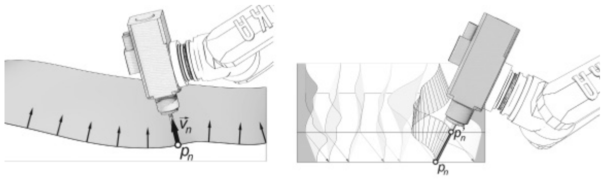

Figure 6. KRL definitions: a) parametric curve 1 at point pn and vector vn (left), b) parametric curve 1 at point pn, vector defined by curve 2 at point p´n (right).

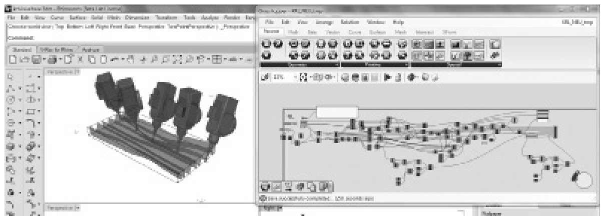

Figure 7. KRL-Visualisation in Grasshopper.

4.2 KRL-Code

For writing KRL-code in Grasshopper, the first step is to create the toolpaths. These consist of the parametrically generated tooltip curves and the tool axis orientation vectors. The points on the tooltip curve can be regarded as the XYZ-coordinates of point pn as described in Section 3.2 and may be used as direct input for the KRL-code.

Converting the vector data into angles A, B, and C is the key to writing KRL-code: The A- & B-angles (yaw & pitch) are calculated using trigonometric functions, while angle C (roll) requires a different approach according to the robot´s workspace (Figure 5):

angle A: $arctan\frac{z}{\sqrt{x^2+y^2}}$

angle B: $2\,arctan\frac{y}{\sqrt{x^2+y^2}+x}$

angle C: The third angle defines the end effector´s rotation around the tool axis, to achieve an optimized orientation to the robot´s basepoint. A separate coordinate point q for the above mentioned basepoint is introduced, which is placed at the center of axis A1. Using this additional information, the final orientation angle can be calculated for each point pn on the toolpath.

The KRL-Code in this paper controls the robot´s end effector and can be seen as a tool-component for various fabrication methods and designs (see Section 1), not just for robot-milling. Even though the parametric design tool has been built for milling purposes and robots, similar toolpath designs (see Section 4.3) can be used for any robot tooling compared to milling cutters. Due to the parametric tool diameter definition, even laser cutting with a tool diameter close to zero could be executed via the proposed KRL-Code. The advantage of a Grasshopper definition to commercial software is the ease in "unplugging" and replacing individual input components defined by constraints such as machining, tool definitions, or various design parameters.

4.3 Parametric Milling Toolpath Design for Robot Control

The toolpath layout to be generated by the parametric milling design tool depends on the desired milling strategy and the stock model geometry (see also Section 5). A typical milling job consists of two steps: rough cut, which coarsely removes material layer by layer, and fine cut (see Figures 1 and 6) where the tool-tip precisely processes the remaining material, thus producing the surface finish. In contrast, the method of flank-milling uses the whole cutting length of the cutter instead of only the tool tip to remove material from the stock model which results in an advantageous optimization of the milling process with minimum tool engagement rates and high material removal rates *(see also Schindler 2009)*.

In the KRL Grasshopper definition we distinguish between these two milling methods to control the robot´s end effector at a point pn of a designed toolpath (Figure 6): The first strategy is to define a parametric tooltip curve, Curve 1, which defines the toolpath represented by the exact location of the tool tip at point pn and a vector vn. This vector originates from a point pn on Curve 1 and can be for example a normal vector of the freeform surface to be milled. First tests show the need to reduce the geometric data load, which is caused by usual rough and fine cuts with multiple and parametrical step-downs. Heavy geometric data interferes with the real-time toolpath calculations resulting in a delayed design feedback. Instead of just "replacing" commercial CAM software for rough and fine-cuts we recommend using this design method for e.g. "engraving" surface structures. *(cf. Aigner & Brell-Cokcan, 2009)*.

The second strategy of flank milling (Figure 6, right and Figure 7) requires two curves to define the toolpath and the resulting cutting surface: Curve 1 which represents the toolpath at the exact location of the tool tip at point pn, and Curve 2 which gives the position of the according point p´n on the tool axis. When milling, the endmill moves through these two curves, where points pn and p´n correspond, by the curve parameterization. The two curves together define a ruled surface where the rulings connect the corresponding points pn and p´n on the two curves.

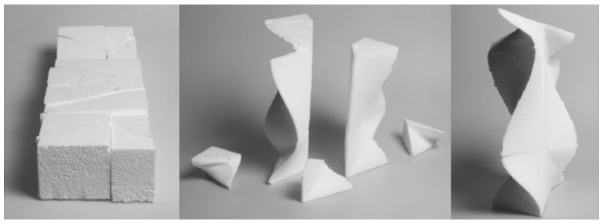
Figure 8. StackIt: A freeform wall resulting from 2x4 parametric toolpaths; cut stockmodel (left), 12 geometric entities (middle), mounted result (right). Design by Christoph Müller and Christian Vladikov.

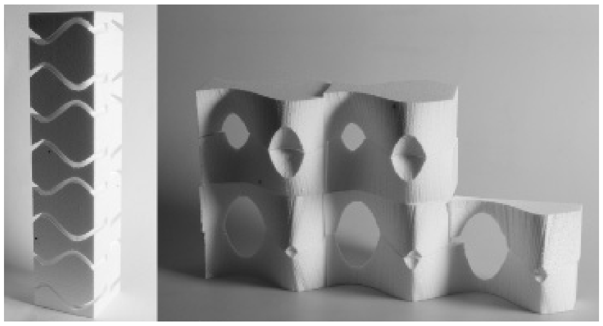
Figure 9. StackIt: A freeform wall resulting from 2x4 parametric toolpaths; cut stockmodel (left), mounted 18 geometric entities (right). Design by Sara Hammar.

5 Design Results

In our courses at Vienna University of Technology we teach students Production Immanent Design and Architectural Geometry to enable them to create their own individual design tools with a maximum of surface control and a minimum of geometric input. In addition to the production immanent components described in Section 2.3 and 4, the design components developed depend on the architectural design task, the geometric entities to be produced, variants and possible combinations of the end object (Figures 8 and 9). A desired minimum of geometric input data for a good toolpath design also results in less machine time, process, and material efficiency.

6 Conclusion

In this paper we have presented an optimized digital workflow that performs the toolpath calculations in real time and thereby provides an additional layer of instant design feedback. The process of flank-milling lends itself particularly well to the proposed method, as the geometric input–opposed to the usual rough and fine cut milling–consists of comparatively few control points of the toolpaths. Therefore, these calculations can be done in real-time and the user immediately sees the modified toolpaths when he alters the object's geometry, while the robot's orientation and KRL-code representation are calculated. Furthermore, the user can access each point individually and basically simulate the entire milling process within the CAD-software without using additional CAM-tools. While it would not be necessary to simulate our KRL-code in the provided robot´s postprocessor, we strongly recommend doing so to avoid unforeseen singularities.

In our future research, we will pursue another use for the presented production immanent design tool, namely parametric mass customization. The latter allows fabricating variations of the same parameterized geometry. As this geometry is generated with a particular history, it is possible to assign milling processes to certain geometric features, a strategy referred to as feature-based CAM design.

Acknowledgements

This research has been supported by the Vienna University of Technology via the innovative project "Geometric Model Building with CNC Technology". See http://www.iemar.tuwien.ac.at/content/robot_milling.aspx for more details.
We would also like to thank Georg Franck-Oberaspach for being supportive of our research, Michael Hofer, Heinz Schmiedhofer and Georg Suter for co-lecturing Architectural Geometry at TU Vienna, Walter Fritz and Christian Kern for open hours at the building workshop and our students for being so enthusiastic and hard-working.
The children´s playground in Figure 1 was designed by Klaus Molterer and Joachim Hackl

References

Aigner, A., & Brell-Cokcan, S. *(2009). Surface structures and robot milling—The impact of curvilinear structured architectural scale models on architectural design and production. In Innovative design & construction technologies—building complex shapes and beyond (ed.). I. Paoletti, 433-445. Milano: Maggiooli Editore*
Baker, D.R., & Wampler II, CH.W. *(1988). On the inverse kinematics of redundant manipulators. In M.W. Spong, F.L. Lewis, & C.T. Abdallah (Eds.), Robot control dynamics, motion planning, and Analysis (468-486). New York: IEEE Press*
Bonwetsch T., Gramazio, F., & Kohler, M. *(2006). The informed wall, applying additive digital fabrication techniques on architecture. Proceedings of the 25th Annual Conference of the Association for Computer-Aided Design in Architecture. Louisville, KY: 489-495*
Brell-Cokcan S., Reis, M., Schmiedhofer, H., & Braumann, J. *(2009). Digital design to digital production: Flank milling with a 7-Axis robot and parametric design. Proceedings of the 27th eCAADe Conference—Computation: The New Realm of Architectural Design. Istanbul: 323-330*
Edgar B. L. *(2008). A short biography of KR 150 L110. In Digital Materiality in Architecture, eds. F. Gramazio & M. Kohler, 49-56. Baden: Lars Müller Publishers*
Iwamoto L. *(2009). Digital fabrications: Architectural and material techniques, New York: Princeton Architectural Press*
Kolarevic B. *(2001). Digital fabrication: Manufacturing architecture in the information age. Reinventing the discourse-how digital tools help bridge and transform research, education and practice in architecture. In Proceedings of the 21st Annual Conference of the Association for Computer-Aided Design in Architecture, Buffalo (New York): 268-278*
Kolarevic B. (ed.). *(2005). Architecture in the digital age. Design and manufacturing, New York: Taylor& Francis.*
Pottmann H., Asperl, A., Hofer, M., & Kilian, A. *(2007). Architectural Geometry. Exton, Pa: Bentley Institute Press.*
Sakamoto, T., Ferre, A., & Kubo, M. (Eds.). *(2008). From Control to Design: Parametric/Algorithmic Architecture, Barcelona: Actar.*
Scheurer F. *(2008). Architectural CAD/CAM: Pushing the Boundaries of CNC Fabrication in Building. In B. Kolarevic, K. Klinger K. (Eds.), Manufacturing material effects: Rethinking design and making in architecture, 211-222. New York: Routledge.*
Schodek D., Bechthold, M., Griggs, J.K., Kao, K., & Steinberg, M. *(2004). Digital design and manufacturing: CAD/CAM applications in architecture, New York: John Wiley & Sons.*
Schindler C. *(2009). Flankenfräsen. In Ein architektonisches Parametrisierungsmodell anhand Fertigungstechnischer Kriterien, dargestellt am Holzbau, 213. Zürich: ETH Zürich.*
Spong, M.W., Lewis, F.L., & Abdallah, C.T. *(1988). Robot control dynamics, motion planning, and analysis. New York: IEEE Press*