Real-Time Robot Simulation and Control for Architectural Design

Johannes Braumann[1], Sigrid Brell-Cokcan[2]
Association for Robots in Architecture, TU Vienna, Austria
http://www.robotsinarchitecture.org
[1]johannes@robotsinarchitecture.org, [2]sigrid@robotsinarchitecture.org

Abstract. *Industrial robots for architectural fabrication have not yet been directly linked to the design process, as current research focuses mostly on the automated generation of robot control data for mass customization. In this paper, we will discuss the use of a real-time programming environment for robot simulation/control and introduce a virtual robot, that allows architects to digitally prototype fabrication processes. While such a real-time approach is also suitable for mass customization, the main advantage is that this interaction with the virtual-robot can be used to intuitively solve complex fabrication problems.*
Keywords. *Industrial Robots; Inverse Kinematics; Virtual Robot; Mass Customization; Simulation; Parametric Design.*

INTRODUCTION

Robots have always fascinated architects, artists and designers with their aesthetic kinematic and unpredictable "humanlike" movements (see Lynn and Rappolt, 2008). By design, industrial robots would have to be considered ideal tools for architectural design and fabrication, due to their comparably low costs, large workspace, and inherent multifunctionality. Looking back to the developments of the last decade, we believe the main reason why robots are only slowly progressing into architectural offices and building workshops to be the perceived uncertainty between control (*action*) and movement (*reaction*).

ROBOTIC ACTION AND REACTION

Comparing the tool movements of robots and Cartesian machines reveals significant differences: For Cartesian machines that can only move in X, Y, and Z direction, the relationship between the movement command and the movement itself is very clear: When the NC (Numeric control) code instructs the machine to move in X direction, the corresponding motor engages until it has arrived at the defined position, making up a clear relationship between *action* and *reaction*. This relationship is much harder to grasp in the case of industrial robots. Robotic arms consist of up to seven axes and have to be considered non-Cartesian machines (Hägele et al. 2008): Any straight movement requires the simultaneous activation of several actuators. Therefore, the robot's multiple degrees of freedom provide great reachability and agility, but also increase the machine's complexity. While the workspace-safety of a Cartesian machine such as a CNC router can be checked by comparing all XYZ positions of the tool centre point with the workspace's boundaries, an industrial robot offers mathematically infinite possibilities to approach a given point XYZ.

One approach towards solving this problem is the so called online robot programming (Biggs and MacDonald, 2003) or teaching. Instead of relying on CAD (Computer Aided Design) data, the robot is moved into position and the rotation values of each axis are saved. This process is repeated until all positions are recorded, and can then be replayed ad infinitum. Similarly, Payne (2011) built a Robotic Digitizer that allows users to transfer the pose of a scale robot model to the actual industrial robot. Using these strategies, robot positions can be intuitively and non-ambiguously defined - however, the programming speed is relatively slow and therefore only suitable for production runs where the same movements are repeated many times - unlike architecture, where the fabrication of many individual parts is the main challenge.

Despite not being an ideal tool for architectural fabrication, online programming represents an interesting approach, as the user does not design a toolpath *for* the robot, but rather designs *with* the robot. Because the robot is handled in real time, potential problems are easily spotted and corrected. However, the optimization process is comparably slow, as the robot has to be manually moved from one position to another.

In this paper, we propose the use of real-time feedback in a virtual environment to understand and process the *action/reaction* relationships of complex systems such as robot kinematics. We will present projects, where a real-time system is used to directly translate collected input data into robotic movement.

REAL TIME PROGRAMMING IN ARCHITECTURE

Within the scope of this paper, *realtime programming* refers to programming systems that immediately present the result of an algorithm – limited only by the available processing power – as opposed to traditional programming that requires active compiling of code. One such tool with significant relevance in architectural design is Grasshopper, developed by David Rutten.

In Grasshopper, the user creates algorithms via visual programming. This is done by connecting the output of one component with the input of another component, translating into an acyclic, directed, graph. These components are "black boxes" that contain operations such as transformations or evaluations, which are performed on the geometrical or numerical data that is plugged into that component. What sets Grasshopper apart from traditional scripting is that the user does not have to compile the code, but can see the result immediately in the viewport, limited only by the amount of CPU cycles that are required to compute the solution. As Grasshopper treats each component separately, even invalid solutions are shown: If a component divides by zero, the output of all previously processed components is shown and only that component is marked as invalid. This process enables a much more fluent workflow, compared to programming languages where only the erroneous line is displayed in the debugger.

Real-time processing therefore allows users to explore the relationships between constraints by interactively working on the parametric model and analysing the results in real-time. Understanding action/reaction is of great pedagogical value and allows much more intuitive approach towards complex systems such as geometry or kinematics.

THE VIRTUAL ROBOT

Integrating industrial robots into a real-time programming system that allows the designer to directly interact with the machine requires the development of a Virtual Robot that reacts to commands similar to a real robot. As the main challenge of working with robots is the complex kinematics, the integral component of a virtual robot is its kinematics solver. There are two relevant types of kinematic solvers: Forward and inverse kinematics. Forward kinematics are used when the code supplies axis rotation values. Starting from the base of the robot, these rotations are iteratively performed on the kinematic chain. After the final axis has been rotated, the position (XYZ) and orientation (ABC) of the tool centre point can be extracted.

Much more relevant and complex is the task of calculating the robot's joint position when only the position and orientation of its endeffector is given.

This is the most common type of robotic simulation and referred to as inverse kinematics (Figure 1). Solving inverse kinematics for industrial robots is made easier by the fact that the rotational axes of the front axes A4 to A6 intersect in extension in a common point, thereby creating a virtual spherical wrist. Because of that, inverse kinematics calculations for most industrial robots can be divided into two manageable segments: The three joints before the spherical wrist are calculated using trigonometric functions, while the rotation values of the remaining three joints require transformation matrices to get the rotation angles that map the robot's base coordinate system of the spherical wrist to the coordinate system of the tool centre point, defined via XYZABC.

Packing such a solver into a component for Grasshopper therefore allows the designer to integrate an interactive robotic simulation into his parametric definition. Any changes performed to toolpaths and programming can then immediately propagate down to the virtual robot, enabling dynamic collision and reachability checking. This programmed realtime-environment will be used in the following chapter for an intuitive design to robotic fabrication process - streamlining design *action* to robot *reaction*.

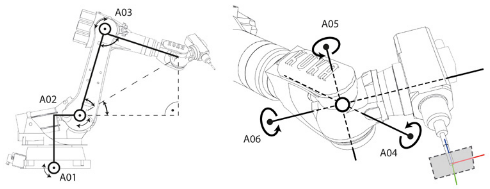


Figure 1
Virtual Robot: Inverse kinematic solver for industrial robots.

INTUITIVE ROBOT CONTROL

Industrial robotic applications usually deal with a fabrication problem in such a way that a problem is defined, a solution programmed and the resulting control data file executed at the robot. Interaction within this process is not necessary because the design is usually finished at the point when robot code is generated. We therefore argue that robotic arms have not yet been used to their full capabilities in industry applications, as they are never linked directly to a design process.

In our research we focus on *intuitive robot control* and *intuitive simulation* for a bottom up design process. To develop fluent action and reaction scenarios for industrial robots in an architectural context we focus on two different ways how this intuitive process can be described: *Static* workflows where the initial parametric input data can be quickly exchanged to generate new robot control data, and *dynamic* processes in which the robot interacts with continuously changing data, resulting in a comparably unconstrained system that cannot be automatically solved.

Static workflows for customized mass production

An established way of automatically programming robotic arms is to create a tightly constrained system with a parametric input data source. Such variable input data can be for example 3D geometry that may be imported from external software or derived e.g. from pixel graphics or 3D scanning data (refer to Brell-Cokcan and Braumann, 2010, and Braumann and Brell-Cokcan, 2012). Putting such an intuitive *action-reaction* scenario in an architectural and/or industrial context this substitutability of data inputs is generally known as *mass customization*.

A recent industrial project showing mass customization was realized together with artists for the *Red Bull Ring* racing track in Spielberg/Austria, where 88 different robotic milled EPS (Extruded Polystyrene) pieces were cast in aluminium and mounted to form a 17x32m arch (Figure 2). Remarkable in this project is that all 88 positive forms with 14mm cross bars and a finished freeform surface were fabricated by the artists themselves using a KUKA KR150 industrial robot. Even though the artists had never worked with robotics, CAM or CAD before, they were able to control and generate robot data immediately. This was made possible by the substitutability of all geometric parts and a predefined Grasshopper definition for KUKA|prc that required just the bare minimum of input values. The manageable amount of inputs and the intuitive simulation with the virtual robot in GH was the key for the artists to plug & play their robot according to their needs.

Scan-To-Mill

The project Scan to Mill utilizes the 3D-scanning capabilities of a low-cost Kinect sensor. Using Microsoft's official SDK (software development kit) in the Visual Studio 2010 developing environment, we developed a custom component for Grasshopper that can extract the Kinect's depth image sensor feeds. Similar to the robotic punching (see Section 6), the depth data is stored in a twodimensional list. However, while the 0-255 value of robotic punching refers to the brightness of the image, the depth sensor's 800-4000 range is equal to each point's distance from the Kinect sensor, measured in millimetres. These depth values are then mapped onto a stock model, with a value of 800 resulting in a cutting depth of 0mm and a value of 4000 resulting in the maximum cutting depth, depending on the tool/material/spindle used. The result is an instant, physical snapshot of the depth data as captured by the Kinect sensor.

All of the mentioned processes work in near-real time, meaning that a threedimensional snapshot of the environment taken with the Kinect is immediately converted into robotic toolpaths that can be simulated with the virtual robot. The developed components allow various settings, such as restricting the range of the depth data to a subset of its maximum 800-4000 range, setting the maximum cutting depth and changing the size of the stockmodel.

SOLVING COMPLEX PROBLEMS THROUGH INTUITIVE INTERACTION

The two projects presented in the previous sections have in common that they are working with sets of constrained data. For the Red Bull arch, the threedimensional geometry is always topologically similar, consisting of a freeform frontside and a constructive backside and by design does not exceed the workspace of the industrial robot. Similarly, the Kinect's 3D depth scanning data always results in a surface with no undercuts, making automated fabrication feasible.

However, in many cases the range of input data is not constrained, making the automated generation of robot control data difficult, as aspects such as (self) collisions and reachability cannot be guaranteed based on the input data. A common way to solve such problems is to create elaborate code structures for solving every thinkable problem that may arise out of such data. This exhibits several problems: First of all, solving a problem using code requires an exact definition of the problem, which in the case of ill-defined problems is hard to achieve. Furthermore, the effort of accounting for every possibility rises with each additional parameter, making it extremely time-consuming.

We believe that, rather than attempting to create an ultimate algorithm for a hard-to-define problem, a preferable approach is to create an accessible, intuitive, real-time interface, that the design can use to intuitively find a solution for a problem that would have been extremely difficult to put into code (refer to Khatib et al. 2011).

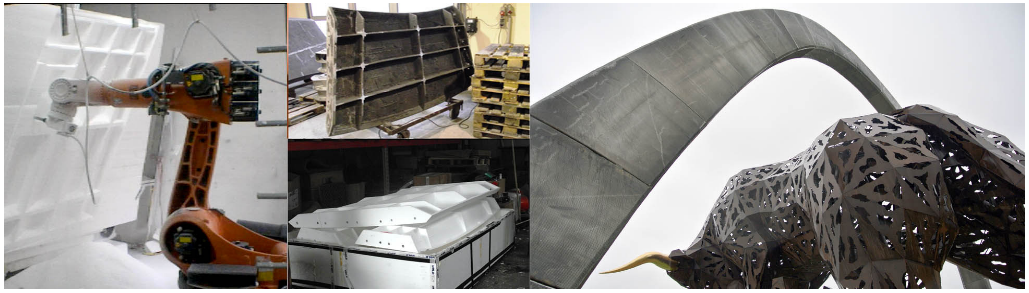


Figure 2
Red Bull arch: KUKA KR 150 milling (left), finished positive EPS forms (lower middle), cast aluminium form (upper middle) finished Red Bull arch (right) project & image courtesy of Neugebauer & Kölldorfer.

An example for such a strategy is the robotic fabrication strategy shown in Section 6.2, where a SplineTex element is held by two robots and has to be manipulated so that it best approximates a digital design. Using a virtual environment that simulates both robots and the SplineTex material allows an intuitive approach towards solving that complex problem.

Robotic and material simulation

In previous research (Brell-Cokcan and Braumann 2010) we discussed the limitations of current CAD-CAM workflows and proposed the use of real-time programming environments such as Grasshopper. In these environments, changes to the digital design can immediately propagate through the parametric model and affect the toolpaths and robot control data files, instead of having to go through three different software tools, as was the case before. Our research resulted in the development of the KUKA|prc software for Grasshopper, which allows an intuitive exploration of the relationships and interconnections of design-action and fabrication-reaction, making robotic fabrication much more accessible.

The addition of a Virtual Robot, as described earlier, greatly augments the capabilities of KUKA|prc, as its kinematics solver can simulate the robot's movements in real time. With these tools, the user can modify a design and in real-time analyze the effects on the robot's posture, general reachability, collisions and axis limits. So far, the robot's interaction with the environment was not considered in such a workflow.

These interactions can be divided into three groups: Subtraction, addition and manipulation. In the scope of this research, we are mostly interested in the last group, manipulation, where material is not removed (e.g. milling) or added (e.g. 3D printing) but rather manipulated, by bending, folding, etc. By analyzing the performance of the chosen material and implementing it as a digital simulation into a real-time programming environment, we expect to be able to virtually prototype complex interactions of material and robots.

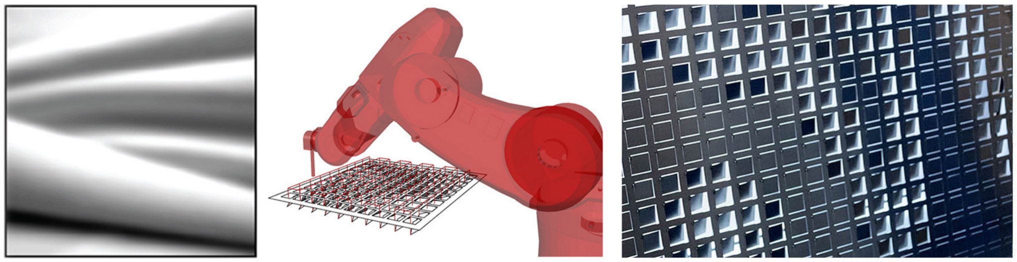


Figure 3
Robotic Punching: Transforming a 2D image to toolpaths, followed by kinematic simulation and final fabrication.

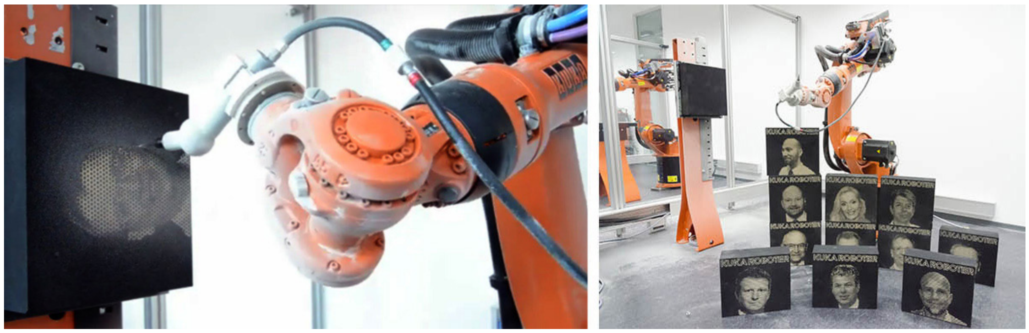


Figure 4
Robotic milling: Parametric toolpaths generated from raster images - a conical mill with variable milling depth creates the grey-scale effect.

The following two projects represent custom design tools created within a real-time programming environment, where not only the industrial robot, but also the material is simulated. Users can directly interact with the design tools by either manipulating the external data input or the internal fabrication parameters and simulate the resulting robotic toolpaths in real time.

Robotic punching

Robotic Punching is built upon the idea of using two-dimensional data to transform a twodimensional material into a threedimensional structure (Figure 3,4). Its base material is a stainless steel sheet with a thickness of 1mm and a waterjet-cut grid of 30mm quadrilaterals, with one corner fixed to the metal grid.

Within the Grasshopper environment, a custom component captures an image from either digital camera, iPad, or local file system. This image data is then sampled according to the brightness of each pixel. By mapping the brightness values onto the grid of the steel sheet, each quadrilateral is assigned a value between 0 and 255. The toolpaths are then laid out in a way that the robot's tool punches the quads, with the depth of each punch corresponding to the quad's value - similar to the project in Figure 4, where a conical milling tool was used to create patterns based on raster images. The orientation of the punching tool is set according to the normal vector of the quad, so that when a quad is bent, the tool automatically follows the material, reducing the contact area to a single point. The movement of the tool - and in extension the full kinematic movements of the robot - is therefore geometrically dependent on the material behaviour during the fabrication process. Figure 5 shows the relationship between punching position of the robot tool, punching depth (d) and tool inclination (a) that is simulated within the virtual robot layout. For collision avoidance, the user can vary between different tool positioning strategies in realtime.

Various forms of interaction are possible, such as adjusting the punching amplitude, setting the maximum tool inclination, or modifying the contrast of the initial image. Due to the real-time kinematics, the effects of these settings can be immediately evaluated and controlled within the design software. Therefore, in this project the robot reacts according to the material behaviour, as simulated in the virtual environment.

Robotic forming of SplineTex

SplineTex is an innovative fibre-reinforced composite material developed by SuperTex, a spinoff from University of Innsbruck's architectural faculty. Depending on the core material, SplineTex can already have different material properties in its soft state, behaving either similar to a rope, or stiffer and plastically deformable like steel rods. By resin injection or infusion, SplineTex can then be hardened and fixed in its current positon. Therefore, SplineTex allows architects and designers to achieve the aesthetic effect of digital wireframes, to be used for interior design, pavilion structures and other organic structures. (Figure 6)

As part of a mutual research project funded by the Austrian Research Association (FFG), we are investigating the robotic fabrication of SplineTex. At the moment, most SplineTex elements are formed with the help of complex jigs. However, while it is cost-efficient to build jigs for larger production runs, they are not suitable for many small, individual pieces. The goal of this project is to find a powerful and accessible workflow that allows SuperTex to use multiple, cooperating robots for forming individually shaped SplineTex elements.

For the simulation of the rope-like SplineTex elements, we developed a spring-based physics engine, while the simulation of the industrial robots is based on the Virtual Robot described in Section 04. The shaping of SplineTex is done by importing a reference curve, and creating a catenary with the curve's length. We then have to find out where the robots should position the SplineTex material and which SplineTex submaterial has the stiffness to best approximate the reference curve. It has been our experience that computationally finding a solution for

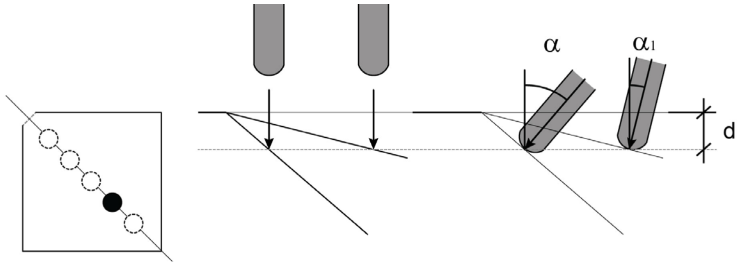
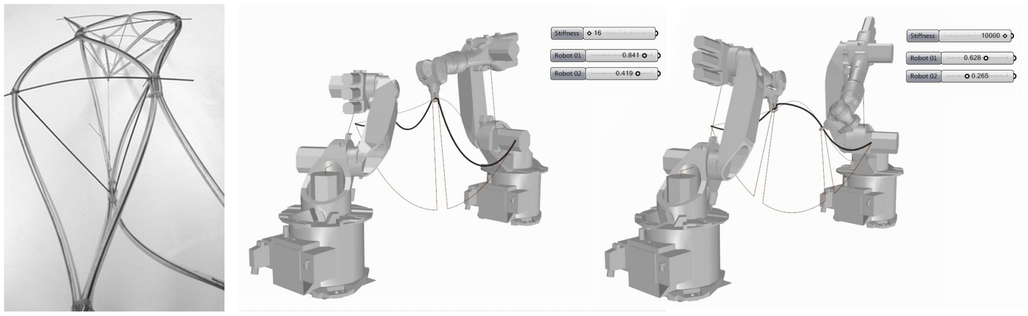


Figure 5
Robotic tool-and material simulation: Robotic tool following the physical bending of steel, the tool's contact position is set according to reachability criteria.


Stiffness 16
Robot 01 0.841
Robot 02 0.419

Stiffness 10000
Robot 01 0.628
Robot 02 0.265

Figure 6
SplineTex, image courtesy of www.supertex.at (left), material and robotic simulation (middle and right).

that problem is difficult, as the stiffness of the material has an impact on the robot positioning and vice versa. However, by creating an interactive, real-time environment, we were able to intuitively position the robots and choose a fitting SplineTex submaterial. While an evolutionary solver proved to be less than ideal for global optimization, it was very useful for fine tuning and ensuring an optimal result.

CONCLUSION

The presented projects should be seen as proof-of-concepts that demonstrate the power of using a real-time-programming environment for fabrication and address topics such as mass customization and material-informed manufacturing. As both design and fabrication are performed simultaneously in one continuous environment, parametric object intelligence can be preserved and toolpaths rapidly prototyped, without having to import and export data. In a general sense, Robotic Punching and Scan-To-Mill each represent a very constrained environment with self-similar geometries and a finite choice of parameters such as punching depth or stockmodel size. Similar concepts can therefore also be applied to large scale architectural applications, such as the Red Bull arch. As all small parts of a whole generally have the same topology, a parametric tool can extract each individual object, apply the same toolpath strategy, simulate every robot position along the toolpath and only ask for user feedback if the robot runs into collision or singularity positions.

However, if a problem is not clearly defined, these *static* approaches cannot be used. Instead of spending much time and effort on programming solutions for such cases, we propose the use of real-time interfaces that allow the designer to intuitively interact with the fabrication process in order to solve these *dynamic* processes.

Including this realtime functionality for fabrication automation and exploration in easy-to-use components for architectural design software enables architects to use robotic technology that by far exceeds the scope of conventional CAD/CAM tools.

ACKNOWLEDGEMENTS

This research has been supported by the Austrian Research Association (FFG). We want to thank Eric Dokulil for sharing his robot expertise, and KUKA CEE as well as SuperTex for providing the equipment needed to perform this research.

REFERENCES

Biggs, G and MacDonald, B 2003, "A Survey of Robot Programming Systems", in *Proceedings of the Australasian Conference on Robotics and Automation, CSIRO*. Brisbane.

Braumann, J and Brell-Cokcan, S 2012, "Digital and physical computing for industrial robots in architecture: Interfacing Arduino with industrial robots", in *Proceedings of the 17th International Conference on Computer Aided Architectural Design Research in Asia* (CAADRIA), Chennai, pp. 317-326.

Brell-Cokcan, S and Braumann, J 2010, "A New Parametric Design Tool for Robot Milling", in *Proceedings of the 30th Annual Conference of the Association for Computer Aided Design in Architecture* (ACADIA), New York, pp. 357-363.

Hägele, M, Nilsson, K, and Pires, JN 2008, "Industrial Robotics", in Siciliano, B and Khatib, O (eds), *Springer Handbook of Robotics*, Springer Verlag, Berlin Heidelberg, pp.963-986.

Khatib, F 2011, "Crystal structure of a monomeric retroviral protease solved by protein folding game players", in *Nature - Structural & Molecular Biology*, 18, pp. 1175-1177.

Lynn, G and Rappolt, M 2008, *Greg Lynn Form*, Rizzoli, New York.

Payne, A 2011, "A Five-axis Robotic Motion Controller for Designers", in *Proceedings of the 31st Annual Conference of the Association for Computer Aided Design in Architecture* (ACADIA), Banff (Alberta), pp.162-169.