Adaptive Robot Control

New Parametric Workflows Directly from Design to KUKA Robots

Johannes Braumann[1], Sigrid Brell-Cokcan[2]
[1] Robots in Architecture | University for Arts and Design Linz [2] Robots in Architecture | RWTH Aachen University
[1,2] {johannes|sigrid}@robotsinarchitecture.org

In the past years the creative industry has made great advancements in the area of robotics. Accessible robot simulation and control environments based on visual programming systems such as Grasshopper and Dynamo now allow even novice users to quickly and intuitively explore the potential of robotic fabrication, while expert users can use their programming knowledge to create complex, parametric robotic programs. The great advantage of using visual programming for robot control lies in the quick iterations that allow the user to change both geometry and toolpaths as well as machinic parameters and then simulate the results within a single environment. However, at the end of such an iterative optimization process the data is condensed into a robot control data file, which is then copied over to the robot and thus loses its parametric relationship with the code that generated it. In this research we present a newly developed system that allows a dynamic link between the robot and the controlling PC for parametrically adjusting robotic toolpaths and collecting feedback data from the robot itself - enabling entirely new approaches towards robotic fabrication by even more closely linking design and fabrication.

Keywords: *Adaptive robot control, Real-time interfacing, PLC, Visual programming*

INTRODUCTION

Within a decade, the role of robotic arms in the creative industry has greatly changed: Once only used for high-end research in cooperation with mathematicians and mechanical engineers, robots have now become a much more common sight [1]. A large part of this development is due to new interfaces that are developed within the creative industry and build upon visual programming environments to intuitively define the robot's toolpath (Braumann and Brell-Cokcan, 2014).

Today, industry has taken notice of these new developments and is beginning to utilize software such as KUKA|prc to quickly define and prototype robotic processes. In parallel the creative industry is looking into automating its robotic processes so that robotic arms can become *creative factories* of their own - towards rapidly fabricating parametrically defined products.

In this research we present a new and reliable interface that allows the user to *stream* robot control data directly from any PC to the robot. While most workflows result in a robot control data file that has to be copied to the robot (Brell-Cokcan and Braumann, 2010) - thus losing its parametric relationship with the code that generated it - we can now bidirectionally couple the robotic arm with the visual programming environment, streaming code to the machine and receiving data back, which can in turn modify and inform the fabrication process.

EXISTING SYSTEMS

The idea of interactive toolpath planning is well researched and one of the core research areas in the field of service robotics (Kunz et al., 2010). The creative industry has also put significant effort into interacting with robots in a more direct fashion, for example Bot&Dolly (Byrne et al., 2014) and SciArc (Kruysman and Proto, 2012) are controlling their robots via custom network infrastructures, while ETH Zurich interfaced their Universal Robots with Grasshopper (Lim et al, 2013).

Research oriented robots such as the URs, KUKA's iiwa or Schunk's LWA come with interfaces that make external control from the outside relatively accessible, as they are often used as parts of larger robotic setups such as DLR's Justin robot (Ott et al., 2006) or the Care-o-bot (Graf et al., 2009). For heavy-payload industrial robots - which are currently more relevant in the field of architecture - we often have to repurpose existing industrial interfaces with their individual strengths and weaknesses.

In the case of KUKA industrial robots, until recently there have been three common ways of streaming data from a PC to the robot: Via custom fieldbus systems, by generic communication methods, and via the Real-time Sensor Interface (RSI). While custom fieldbus systems offer great flexibility, they are complex to set up and require additional costly hardware. On the other side, generic communication systems (e.g. using the RS232 serial port (Figure 1) or as XML over TCP/IP via the Ethernet.XML plugin) are primarily made for exchanging state data, but not optimized for streaming motion data. In order to achieve a fluent, interpolated movement, the robot has to know at least one position following the current movement target - i.e. a custom buffer has to be implemented, which introduces a significant amount of complexity.

The Real-time Sensor Interface's main purpose is to offset toolpaths in real-time, e.g. by reacting to data from a force/moment sensor in order to ensure that neither workpiece nor robot are damaged. By enlarging the maximum allowed positional offset and replacing the sensor with a custom server application, position information can be streamed at the

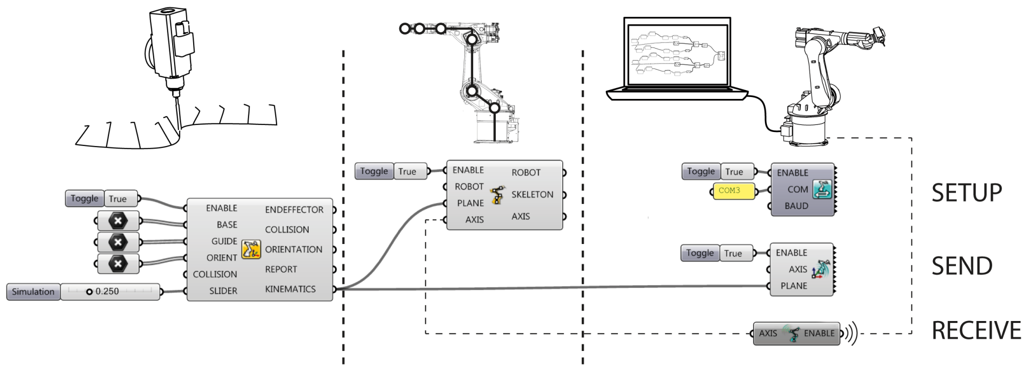
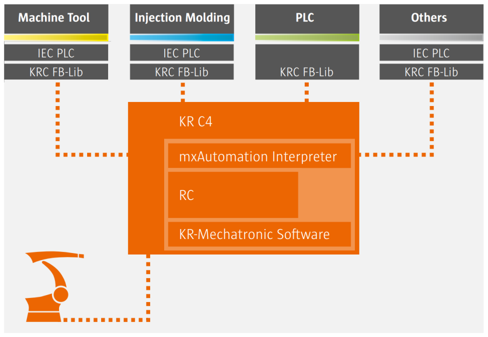


Figure 1
Robot programming through Grasshopper via the robot's serial port. Early approaches towards direct robot control (Braumann and Brell-Cokcan, 2012)

robot's cycle time, i.e. 4ms (250Hz) or 12ms (80Hz). Unlike other systems that basically transfer only the basic geometric data and toolpath parameters (e.g. target position as XYZABC, speed in m/s, acceleration in percentage, and interpolation strategies) and leave the interpolation and interpretation of these values to the robot, RSI requires the user to take care of all these parameters. Therefore, at every cycle a new offset value is sent that has to have these parameters "baked in" - i.e. a higher speed is represented by a larger distance over the same cycle time. This allows fine-grained control and extremely quick reaction times, but by doing that makes the user responsible for even minute details that can possibly damage the mechanics and endanger the user. Initial experiments with non-optimized code led to oscillations around the robot's target and significant issues with latency spikes exceeding the robot's cycle time on non-real-time systems such as Windows (and by extension Grasshopper), causing vibrations in the robot's movements.

A middle-ground between the aforementioned interfaces can be found in KUKA mxAutomation, which makes it possible for external controllers with an embedded PLC (programmable logic device) to command KUKA robots on the basis of regular motion instructions. So rather than having to define a custom interface between the robot and a controller, mxAutomation provides a link to several brands of controllers such as Siemens Sinumerik, Rockwell, and CODESYS-based systems.

On the robot side, mxAutomation acts as an interpreter that accepts, buffers, and executes commands and then returns data on the success of that operation. Interestingly, for example CODESYS uses a visual programming approach that is somehow similar to Grasshopper, where components provide certain inputs and outputs, with editable code working in the background. However, where Grasshopper code is generally run once, providing a certain result at the end(s) of the directed, acyclic graph, the CODESYS graph is called at every cycle and movement sequences are defined by e.g. linking the DONE output of a component with the EXECUTECMD input of the next component, so that one movement is called once the previous movement has finished (Figure 2).



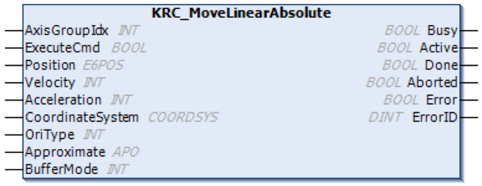


Figure 2
mxAutomation structure (above), PLC function block for a linear movement in accordance with IEC 61131 (below). (KUKA, 2014)

APPLYING INDUSTRIAL INTERFACES IN THE CREATIVE INDUSTRY

As part of a research project we are currently building upon an as of yet unnamed interface from KUKA that utilizes generic UDP packets to communicate with and control KUKA robots. So instead of having to build up an expensive fieldbus infrastructure, it enables us to basically use every network-capable device to stream information to the robot and process the returned data, from regular Windows PCs and laptops to smartphones and tablets, and even tiny ARM-based microcontrollers such as the Raspberry Pi.

Using an early version of this communication library, we created a custom "soft-PLC" that runs as a separate high-performance thread on a regular Windows-PC and directly communicates with

Grasshopper. The interpreter on the robot side accepts commands and queues them inside a buffer to execute them in sequence. At every cycle, these commands are refreshed and values from the robot, such as all axis positions, velocity, etc. are returned to the server program. Internally, all robot commands are executed exactly as if they were written into a KRL file, with all safety and interpolation options intact.

This gives us a great number of advantages over custom-developed generic interfaces: First of all, as an interface marketed to industry it is highly stable and well tested. It can be deployed without requiring any additional hardware on either robot or PC side and works bi-directionally so that e.g. sensor values or robot parameters can be returned to the controller. While the problem of Windows (by default) not being a real-time capable operating system persists, the integrated buffer gives us a much larger leeway so that brief communication issues do not have an impact on the robot's toolpath or the stability of the communication. However, the buffer also prohibits any hard-real-time applications, i.e. processes where millisecond reaction times are needed.

We believe one of the main reasons for the popularity of robots in the creative industry to be their robustness: Instead of having to focus on the mechanics of the machine, the creative user can focus entirely on the application and rely on the robot to follow the instructions as well as possible. Similarly, this new interface opens up new possibilities without requiring us to control every finest robot detail in real time (Figure 3).

Having a capable interface is only one step towards enabling adaptive and dynamic robotic processes. The main question remains on how dynamic processes can be controlled by the user and how the flows of data are laid out between the robot and the external controller.

KUKA|PRC: A LIBRARY FOR ROBOT SIMULATION

Currently the most common workflow for interacting with a KUKA robot in the creative industry is split into two parts - *programming* and *execution*: First, the robot's movements are visually programmed within Grasshopper and then simulated via the according robot control plugins. Once the simulation is working, a control data file is written and then copied via Ethernet to the robot where it has to be manually executed. Similar to Grasshopper's process of "baking" geometry, which turns parametric objects into static geometry that can be exported and rendered, the parametric robot control data is condensed into a static text file.

This process is also the default workflow of KUKA|prc, which we developed as an interface for

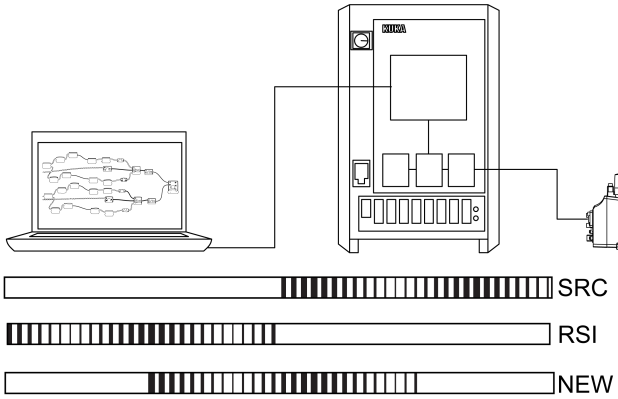


Figure 3
Location of command logic: Entirely on the robot controller for regular control data files (SRC), entirely on the external controller for real-time controllers (RSI), distributed command logic (NEW).

controlling KUKA robots through the visual programming environment Grasshopper. In the 6 years since the first approaches were shown at eCAADe 2009 (Brell-Cokcan et al., 2009), the software has evolved from simple code generation to a full robotic simulation environment, building upon a custom kinematic solver that can simulate 6 and 7 axis robots, supports up to 4 external axes and plots the entire axis movements of a robotic program in advance so that it can be immediately - either visually or automatically - checked for unreachable points or singularities. As such, KUKA|prc is now being used not only at many universities worldwide, but also at global leaders in the aeronautical industry, in the film industry, at high-end fabrication companies, architectural and industrial design offices as well as FabLabs and enthusiasts.

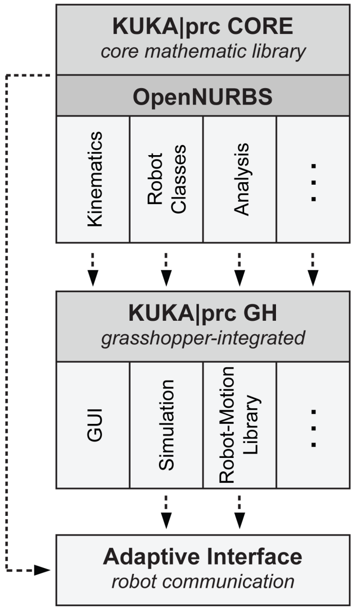


Figure 4
KUKA|prc structure: Underlying custom CORE library providing all mathematical and geometric calculations building upon OpenNURBS. The Grasshopper component exposes these functions and provides an asset library. ARC utilizes the GH interface as well as core functions.

It is built upon a fully original core library that defines the robot-specific classes and performs all core operations such as inverse and forward kinematics, collision checking, automated calculation of external axes, etc. and can be built to work either with RhinoCommon for applications within Rhinoceros 3D or openNURBS for external usage (Figure 4).

For our new interactive workflows we can therefore use the stable and tested infrastructure of KUKA|prc, with the library of robot commands, robot models, tool models and robot specific functions and expand it towards adaptive workflows.

INTERACTIVE ROBOT PROGRAMMING

Interactive toolpath planning is a highly complex topic, with topics such as bin picking having been researched for more than 30 years (Siciliano and Khatib, 2008) without arriving at some kind of universal formula. The main challenge of bin picking is that a system has to identify the countless objects in a box, grab the right one with the right strategy, and at the same time avoid collisions with e.g. the sides of the bin. To do so, the system has to evaluate countless possible approach-strategies towards different objects and even simulate their physical behavior (Schyja and Kuhlenkötter, 2015).

However, we believe that most possible application within the creative industry do not pose this level of complexity, but instead can be tackled with existing tools and workflows, such as the visual programming environment Grasshopper.

As such we built our range of tools for adaptive robot programming, referred to as ARC, on top of the existing KUKA|prc library and components, allowing us to re-use the different assets as well as entire parametric definitions. Therefore, the user can first simulate and evaluate the entire process "offline" using KUKA|prc and then connect to the robot and go "online" to stream the commands. ARC therefore does not have to offer any simulation capabilities by itself, but only accepts data, interfaces with the robot, and visualizes the position data that gets sent back from the robot - keeping the software slim and highly performant while allowing the user to simulate other projects in KUKA|prc without disturbing the dataflow.

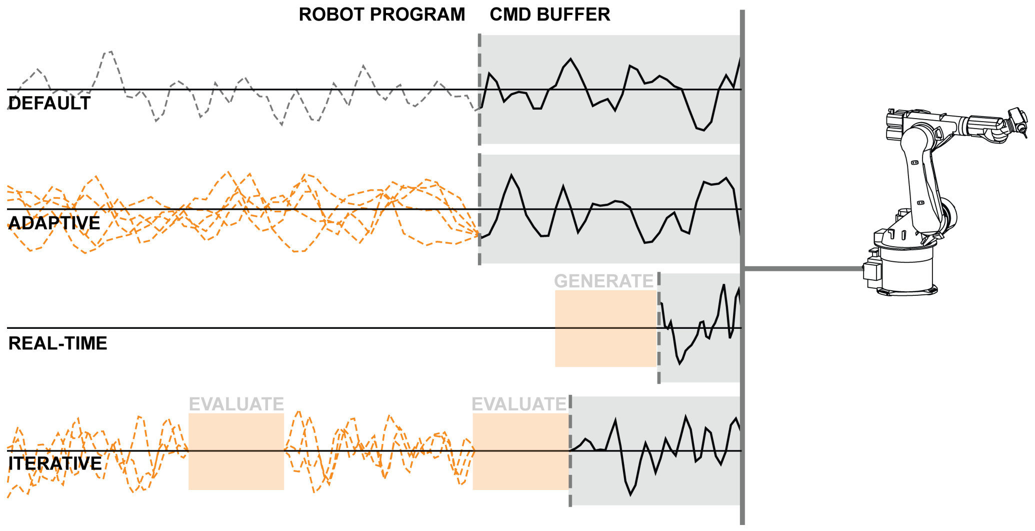


Figure 5
The four developed Interaction modes for ARC: Default, adaptive, real-time, and iterative.

Through analyzing our past robot projects, we have identified four distinctive interaction strategies that should cover most current requirements for interactive toolpaths within the creative industry (Figure 5). These strategies can be switched within the Grasshopper component depending on the preference of the user.

Default Mode: This is the most basic mode whose purpose is simply to transfer commands to the robot. Once commands are connected to the ARC component they are immediately processed and sent to the robot. While the robot is already moving, additional commands are streamed until the interpreter's buffer has reached its defined capacity. From then on, executed commands are culled from the buffer and replaced with new commands. If the input data from the ARC component is changed, it immediately wipes the robot's buffer and starts streaming new commands.

Adaptive Mode: In comparison with default mode, adaptive mode users a much shallower buffer that contains only the bare minimum of commands to cover brief lags in the Ethernet communication. While usually robot programs only allow the user to set a speed override, adaptive mode attempts to keep the entire robot job completely parametric for as long as possible. Only when a command is committed to the robot's buffer it cannot be changed anymore. This allows us to incorporate sensoric feedback to continuously inform the fabrication process and all its parameters

A current application of the adaptive mode is AROSU, an EU-funded research project that explores the structuring of natural stone (Figure 6). The new "adaptive" toolpath capabilities are used to adjust toolpath parameters and geometry in real-time, compensating e.g. for the non-homogeneous structure of natural stone.

Iterative Mode: In interactive mode, the ARC behaves mostly like default mode with a similarly large buffer, but does not accept any new commands until the robot signals that it is ready for the next iteration, i.e. when the previous commands have been processed and an optional timeout has passed. The main applications is therefore to switch between fabrication and evaluation. Essentially, the user only has

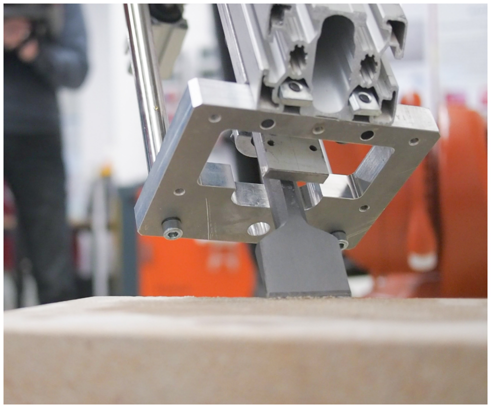
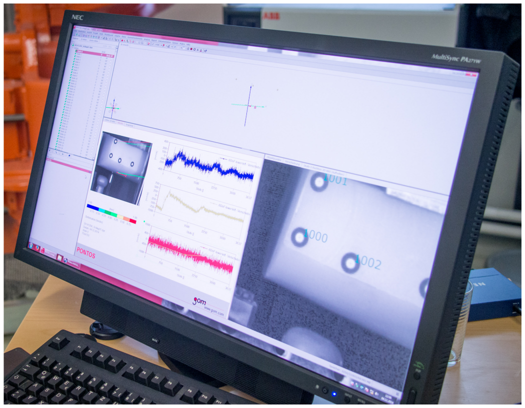



Figure 6
Early, mechanical prototype for the structuring of natural stone (left), sensor analysis and output (right).

to define a single operation such as "pick up here and drop off there" and a set of global rules regarding the placement. Then the robot performs an operation, sensor data - e.g. from a 3D camera - is processed, and the next iteration adjusted according to the captured data. Refer e.g. to Dörfler et al., 2012.

Real-time Mode: Similar to the adaptive mode, real-time mode uses a shallow buffer to reduce the reaction time to a minimum within ARC's framework. However, while adaptive mode works with the entire parametric toolpath, real-time mode generates each command on the fly by taking the current robot-position, calculating the difference to the target position and then adjusting it according to the pre-set step-values. So if the target, e.g. based on live motion capture data, were to move, with a step size set to 5mm and a buffer of 5 positions, the robot would continue for 25mm before reacting to the change.

CHALLENGES AND OUTLOOK

This research marks our initial steps towards even more integrated design and production workflows that allow us to directly apply the flexibility of visual programming in the physical world using robotic arms.

With its large user-base, powerful geometric functions and accessible layout, Grasshopper marks a very suitable platform for defining such flexible robotic processes. However, in more in-depth analysis we observed some issues when e.g. frequent redraws are issued. Finally we implemented our own display pipeline to Rhino and spun out all interfacing operations into highest-priority threads that run in parallel to Grasshopper for a minimum of interference. While this still does not make Windows a real-time operating system, most multi-core processors have enough processing power to stream even dense position data to the robot with a minimum of lag. The combination of a simulation framework with a new data-streaming interface is highly synergetic, as e.g. jobs can be automatically simulated without user-interaction and are only queued for fabrication once all performance criteria such as avoiding collisions and ensuring reachability are met.

While similar robot control solutions exist, this new interface offers the advantage that the robot reacts in exactly the same way as if it was processing a regular robot control data file. Furthermore, it is extremely easy to install and maintain, requiring only a regular Ethernet cable, the range of Grasshopper components, and the relevant KUKA software on the robot. Once the IP of the robot has been set, commands can be immediately streamed to the machine.

We believe that adaptive robot control marks a very important step for the creative industry as it can be used for dynamic processes as well as automation. Rather than having to set up complex communication infrastructures, a single robot connected to a single PC can become a powerful *creative factory*, interfacing e.g. with a web-server to receive data from users around the world. Through this ease of use we hope to enable customized and reactive fabrication processes for highly individualized products on all scales, from industrial design to architecture and beyond.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n°606453.

REFERENCES

Braumann, J and Brell-Cokcan, S 2011 'Parametric Robot Control: Integrated CAD/CAM for Architectural Design', *Proceedings of the 31st Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*

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

Braumann, J and Brell-Cokcan, S 2014 'Visual Robot Programming – Linking Design, Simulation, and Fabrication', *Proceedings of the 5th annual Symposium on Simulation for Architecture and Urban Design (SimAUD)*

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

Brell-Cokcan, S and Braumann, J 2014 'Robotic Production Immanent Design: Creative toolpath design in micro and macro scale', *Proceedings of the 34th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*

Brell-Cokcan, S, Reis, M, Schmiedhofer, H and Braumann, J 2009 'Digital Design to Digital Production: Flank Milling with a 7-Axis CNC-Milling Robot and Parametric Design', *Proceedings of eCAADe 2009*

Byrne, K, Proto, J, Kruysman, B and Bittermann, M 2014, 'The Power of Engineering, the Invention of Artists', in McGee, W and Ponce de Leon, M (eds) 2014, *Robotic Fabrication in Architecture, Art, and Design 2014*, Springer

Dörfler, K, Rist, F and Rust, R 2012, 'Interlacing - An experimental approach to integrating digital and physical design methods', in Brell-Cokcan, S and Braumann, J (eds) 2012, *Robotic Fabrication in Architecture, Art, and Design*, Springer

Graf, B, Reiser, U, Hägele, M, Mauz, K and Klein, P 2009 'Robotic Home Assistant Care-O-bot® 3 - Product Vision and Innovation Platform', *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*

Kruysman, B and Proto, J 2012, 'Augmented Fabrications', in Brell-Cokcan, S and Braumann, J (eds) 2012, *Robotic Fabrication in Architecture, Art, and Design*, Springer

KUKA, Robotics 2014, *CODESYS Library for KUKA.PLC mxAutomation 2.0*, KUKA

Kunz, T, Reiser, U, Stilman, M and Verl, A 2010 'Real-Time Path Planning for a Robot Arm in Changing Environments', *Proceedings of the International Conference on Intelligent Robots and Systems (IROS'10)*

Lim, J, Gramazio, F and Kohler, M 2013 'A Software Environment for Designing through Robotic Fabrication', *Proceedings of the 18th International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA 2013)*

Ott, C, Eiberger, O, Friedl, W, Bäuml, B, Hillenbrand, U, Borst, C, Albu-Schäffer, A, Brunner, B, Hirschmüller, H, Kielhöfer, S, Konietschke, R, Suppa, M, Wimböck, T, Zacharias, F and Hirzinger, G 2006 'A humanoid two-arm system for dexterous manipulation', *IEEE-RAS International Conference on Humanoid Robots*

Schyja, A and Kuhlenkötter, B 2015 'Realistic simulation of industrial bin-picking systems', *6th International Conference on Automation, Robotics and Applications (ICARA)*

Siciliano, B and Khatib, O (eds) 2008, *Springer Handbook of Robotics*, Springer

[1] http://www.robotsinarchitecture.org/map-of-creative-robots