Haptic Learning

Towards Neural-Network-based adaptive Cobot Path-Planning for unstructured spaces

Gabriella Rossi[1], Paul Nicholas[2]
[1]Odico [2]CITA / KADK
[1]gr@gabriellarossi.tech [2]paul.nicholas@kadk.dk

Collaborative Robots, or Cobots, bring new possibilities for human-machine interaction within the fabrication process, allowing each actor to contribute with their specific capabilities. However creative interaction brings unexpected changes, obstacles, complexities and non-linearities which are encountered in real time and cannot be predicted in advance. This paper presents an experimental methodology for robotic path planning using Machine Learning. The focus of this methodology is obstacle avoidance. A neural network is deployed, providing a relationship between the robot's pose and its surroundings, thus allowing for motion planning and obstacle avoidance, directly integrated within the design environment. The method is demonstrated through a series of case-studies. The method combines haptic teaching with machine learning to create a task specific dataset, giving the robot the ability to adapt to obstacles without being explicitly programmed at every instruction. This opens the door to shifting to robotic applications for construction in unstructured environments, where adapting to the singularities of the workspace, its occupants and activities presents an important computational hurdle today.

Keywords: *Architectural Robotics, Neural Networks, Path Planning, Digital Fabrication, Artificial Intelligence, Data*

INTRODUCTION

Within the realm of automated manufacturing, industrial robots are programmed to run at full speed within fenced and clearly structured production environments, performing repetitive tasks. However, their introduction within the creative architectural realm over the last decade has shifted what we consider the role of these machines to be (Scheinman et al. 2016). Architects and designers are developing novel fabrication methods using robots, and experimenting with construction methods in which humans interact in and influence the same environment as robots. This motivates a need, not only for smart machines that can adapt to a changing surrounding, and react to the materiality of their production, but also for greater flexibility and ease in robotic programming (Braumann et al. 2016). The creative community has developed design-environment-based

robot control tools such as KUKA|prc [1], HAL [2] and Robots [3] which require the designer to manually draw and adjust toolpaths curves until the robot can follow without any collision. This requires knowledge and custom calculation of obstacle position and avoidance parameters in advance. Thus, complex path planning and environmental sensing remain difficult to integrate. Haptic teaching, through direct human-robot interaction, offers the possibility to explore alternatives to complex robot programming that can be applied in a safe non-fenced research lab environment.

COBOTS

This paper focuses on table-top Collaborative Robots (Cobots) and the potential they can offer coupled with Haptic Learning. First conceived in the late 1990s as productivity tools for small-scale assembling in the automotive industry (Peshkin et al. 2001), Cobots have quickly become a symbol of Industry 4.0, marking the rise of smart auxiliary machines that can work alongside humans in a safe close environment. Established industrial robotic manufacturers have entered the Cobot market, ABB with the double handed YuMi-IRB 14000, and Kuka with LBR iiwa, competing with younger producers such as Universal Robots. Cobots are mechanically different from their industrial counterparts: They present slender and light bodies, and no sharp edges to minimize the possibility of user harm. They have a relatively long reach due to their wide rotational range, but smaller payload capacity. Most importantly, every axis is equipped with force-torque sensors enabling the robot to feel and react to contact and pressure at every joint. This enables motion to be halted in case certain thresholds are exceeded, preventing injuries, and self-damage. Another important difference is the introduction of Manual Compliant Mode (MCM). This mode allows to release the brakes of the motor, so each axis can be easily manipulated by the user, while a mechanism compensates for eventual weights of tool and workpiece in order to avoid collision with the base. For the scope of this research we use a Universal Robot UR5-e (figure 1), with input via MCM a key part of the workflow.

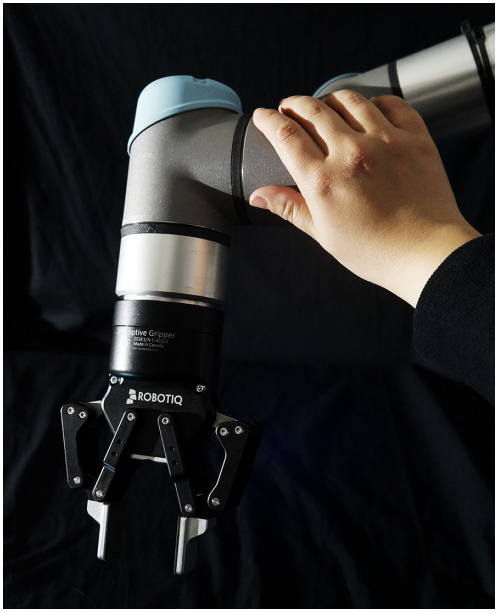


Figure 1
UR5-e equipped with a Robotiq gripper, haptically guided by the User

CURRENT PATH PLANNING APPROACHES

Parametric Instructions - Digital Programming

Robotic fabrication in architecture exploits their potential as constantly re-programmable tools for customised, non-standard fabrication (Bonwetsch 2015). Recent research has demonstrated how the inherent precision and programmability of industrial robots opens up new opportunities for customised design, in particular around materials, forms and construction methods that are geometrically complex, high-precision or high-tolerance (Bloss 2014)(Brell-Cokcan and Braumann 2010), as well as the requirement to match robotic fabrication with 'fabrication-aware' design approaches that incorporate material and fabrication limits into process of specifying robotic commands (Nicholas 2018)(Menges 2012)(Svilans 2019). Creative usage of robots is facil-

itated through geometry-based visual programming tools (for example GH on Rhino) which allow to immediately visualise fabrication feedback, and update the instructions if any change to the geometry is made without incurring a significant overhead compared to common off-line programming methods, since the relationships between the elements are defined parametrically. It also allows for mass customisation of components deriving from the same parametric workflow. This process is especially useful for repetitive applications such as additive manufacturing, milling, or assemblies (Stumm et al. 2016).

However, in some cases the complexity of the path planning and collision detection and solving goes beyond what is straight-forwardly programmable in the parametric design environment. This is the case encountered during the development of multi-robot geometrically-irregular (Gandia et al. 2018). This research utilised existing tools for path planning developed for autonomous mobile robots. For example, the Open Motion Planning Library (OMPL) [4], that consists of many state-of-the-art sampling-based motion planning algorithms, that use different path optimizers, as well as collision detection. It is implemented through the Robotic Operating System (ROS) [5] which runs on Linux and requires a considerable amount of robot programming experience. Research were successful at making use of OMPL to program multi-robot assemblies, by developing a pipeline pushing data from the design environment, to VRep (a windows-based wrapper for OMPL) [6] through a container platform as part of the Compas framework [7]. However, they report the need for a faster, simpler and more intuitive control directly from the design environment, without the need to learn how to use the other softwares involved in this process.

In this case, even though the tools used originated from the field of mobile robotics, the path planning strategy was a global one: To find an optimal path avoiding any collisions given known start and end points and obstacle positions. Optimal paths are created within the programming environment with the element, manipulator and obstacles mapped. However the time cost of these algorithms means that they are not suitable for implementation in real-time, as they sometimes cannot find a solution and the assembly sequence has to be restarted from scratch. Conversely, in the world of mobile robotics, path planning strategies are local, and they leverage localized sensor feedback technology to avoid collisions with dynamic obstacles. They can thus be implemented without complete knowledge of the environment and require far less computation.

Repetitive Instructions- Haptic Programming

Robotic Fabrication in industrial contexts, using cobots, exploits their potential as very easily programmable tools through Kinesthetic Teaching -or learning by demonstration, which is defined as the intuitive nature of teaching robotic task through direct physical interaction (Welschehold 2017). This method allows to overcome difficulties of more advanced robotic methods (Iturrate 2017), and guarantees that the taught motion is both efficient, safe, and most convenient to the robot joint configuration. The robot operator puts the robot in Manual Compliant Mode (MCM), and moves the robot to the desired position, records the target using the electronic pendant, and then on to the next one. While easy and fast, the goal of programming through Kinesthetic Teaching is for the robot to just repeat the task as taught. The teaching is thus "one-time-use", since the recorded motion is very specific to the task and the workspace calibration.

The inability to reuse recorded teachings is regarded as a major shortcoming of kinesthetic teaching, and critically limits its potential in the field of construction and assembly, especially in unstructured environments. One approach aimed at addressing this problem is to move away from teaching for replication, to teaching for adaptation (Stumm and Brell-Çokcan 2018). Using a simplified CAD environment, a predefined robot path is digitally programmed, and defined as a sequence of motion primitives. Then,

when physically running, this ideal path is adapted to the tolerances of the physical world through haptic programming by the robot operator at the start of every motion primitive. This approach uses human-robot interaction as short circuit to complex environment aware programming and sensing feedback, however, the user is not allowed to move the robot beyond the preprogrammed spatial interval of operation. Further efforts are currently directed at direct usage of recorded kinesthetic teaching data to derive haptic motion primitives. Cobots do allow for motions to be easily recorded and streamed, however, a considerable amount of work needs to go into cleaning of recordings, and relating them to goals the robot is set to complete (Stumm et al. 2019).

A HYBIRD ROBOTIC PROGRAMMING APPROACH

Neural-Networks meet Human-Robot Collaboration

While Neural Networks are widely established in the field of mobile robotics to establish vision-based on-the-fly navigation decisions and motion within their environments (Zou et al. 2006), their usage to control the motion of robotic arms is limited. However, we believe that the use of Neural Networks can aid the generation of robotic motion from within the design environment. By combining Neural Networks with Robot-Human collaboration, we are able to bridge the gap between the generic dataset with and the physical environment the robot needs to operate in, through haptic inputs.

We develop an experimental approach that allows for the robot to generalize and learn both from a digital dataset and from previously recorded motions, rather than replicating any of the two per se. The NN-powered robot develops an intuition on how to autonomously handle motion, and obstacles within its environment. Our approach capitalizes on open source plugins available for Grasshopper visual programming environment for Rhino, and provides control in an easy and fast way without leaving the geometry software, where the robot's task is parametrically defined, in the case of architectural applications. We believe this makes advanced robotic control available for creative users who are familiar with geometry software. We call this method *"Haptic Learning"*.

Our approach learns from precedent experiments utilizing Neural Networks with robotic tasks. For instance, (Wu and Kilian 2018) use a convolutional neural network trained with a collecteddatabase of stochastic assembly experiment images, to suggest assembly positions of desired topological connections. However, relying on external sensor feedback via images of the robot's pose and position through space, or scans of the assembled logs to compile the dataset, significantly increases possibilities for data loss and inaccuracies. Our approach instead utilises clean and controlled digitally generated data, coupled with live joint position streaming from the robot controller when the robot is guided in MCM by the user. Using a data handling, prepping and wrapping approach similar to that developed by (Rossi and Nicholas 2018), we are able to control the workflow directly from within the design environment.

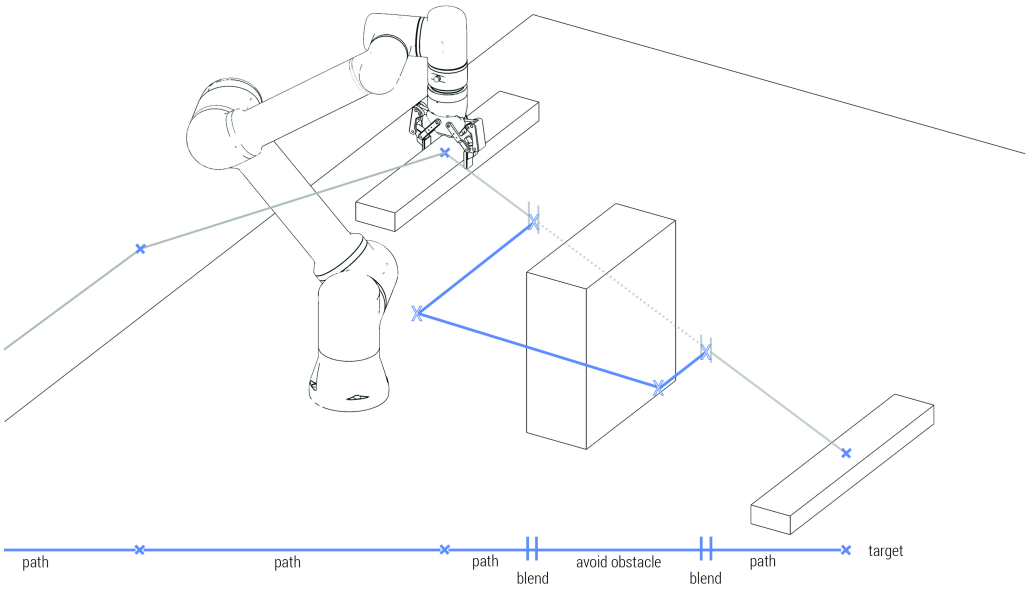
METHOD

The method is fully developed on Grasshopper for Rhino, and is tested to control a UR5-e. It is developed in three stages described below.

Data Generation & Network Training

The initial step is to generate data to train the neural network. Using a combinatorial algorithm, we are able to generate a great amount of digital data via simulation in a short period of time. To run the simulations, we use Robots [3] plugin for Grasshopper, and create a dataset of input geometrical information and output robotic joint configurations. This data is then normalized and wrapped into tensors. Using Owl [8] plugin for Grasshopper, we train a Deep Neural Network, using Accord.Net backend. The output of the network is then unpacked and evaluated. The great advantage of using Grasshopper as an environment, is that it allows to visualize not only the output

Figure 2
Localized obstacle avoidance on a toolpath. The colliding path is split and an obstacle avoidance routine output by the network is grafted and blended.



of the network, but also the training phase, making it really easy to understand the process, debug any problems, and work with the learning parameters in an immediate and intuitive way. It also avoids unnecessary hassle in dealing with different file formats for handling the data or the network in other environments.

Spatializing avoidance strategies

Another advantage of working within the geometric environment is that it makes it very easy to integrate the output of the neural network, in terms of obstacle avoidance, to the robot path that is parametrically being programmed with workflows described in section "Parametric Instructions - Digital Programming". It then becomes possible to generate the path, detect obstacles, push the geometric parameters of the obstacles to the neural network, and then blend in the obstacle avoidance routine output from the neural network back into the original path (figure 2). Therefore it becomes really important to go beyond a generic response to obstacles, but rather to develop a relationship between the relative position of the obstacle and the robot pose, and the reaction the network suggests to use in order to avoid clashing. This conditioning can be easily integrated in the training dataset, since we are using a geometry based digital robot simulation.

Haptic Learning

Although all of our approach so far has been digitally based, Haptic inputs are crucial for real-life scenarios and applications. When dealing with complex, task specific applications, it becomes far more sensed to start from a safe and efficient human input, rather than trying to digitally generate all possible combinations of all possible motions. In fact, this would also act to the detriment of the Neural Net-

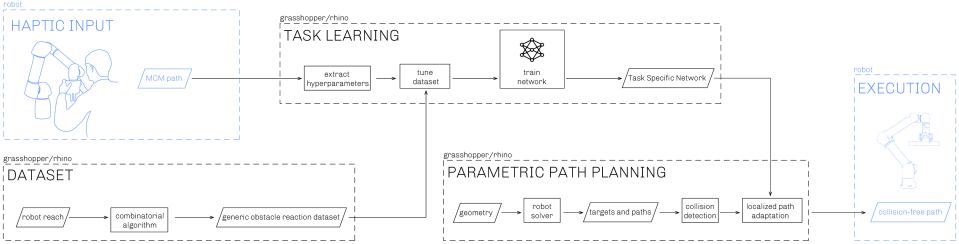


Figure 3
Haptic Learning Workflow. Physical inputs tune the digital dataset.

work's accuracy, since multiple inputs would have the same output. The method we developed allows to use physical training to make the digital dataset respond to the specificity of the physical situation. It allows to explicitly tune the dataset using implicitly embedded physical hyperparameters, such as preferred avoidance strategies, offset between the obstacle and the robot to account for the tool and potential objects it is carrying, and the zone of the motion. This becomes very important when the robot is holding a tool which is not orientation agnostic, for instance. By using a custom GH python component, we are able to have this happen again within the design environment, with a few passes needed to be performed by the robot operator prior to the start of task execution.

EXPERIMENTS AND RESULTS

Linear motion

The first experiment served as a proof of concept of the methodology. The network was fed paths composed of single segment going from left to right. The segments followed a sampling grid (figure 4a) of the robot's reach range, and had different lengths. The network outputted a series of joint targets the robot would have to follow to go from start to end of the test segment. The validation data consisted of segments that were not in the sampling grid, whether by position or by length. The network scored an error on the cost function of 0.008 after 2000 iterations, training on 800 tensors. When observing the actual angle values of the joints, the variations fall within an acceptable tolerance for obstacle avoidance (figure 5a).

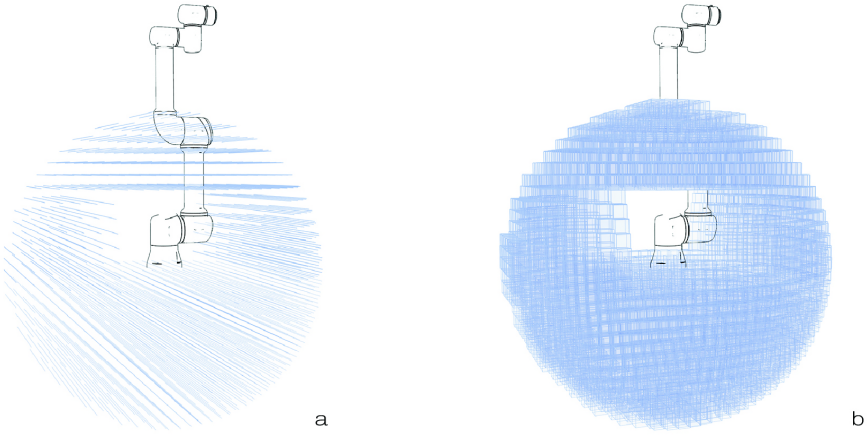
Obstacle along line

The second experiment introduced the obstacles to the above experiment. Again a sampling grid was generated with an obstructing cube at the middle point of the path. The geometric coordinates were fed to the network, which again outputted the joint targets for the robot. The robot scored an error of 0.003 after 1000 training iterations on 800 tensors. The validation tests showed that the network was able to adapt not only to curves in unknown positions, but also to different positions of the obstacle along the line, which it had not previously trained on. This meant that it was able to generalize the relationship between curve, obstacle, and pose.

Obstacle with respect to robot pose

Rather than populating the robot's reach range with path curves, in this third experiment we populate it directly with obstacles (figure 4b), and defining how the robot should react to them using geometric conditioning, for instance whether the robot should go around it from the top, or from the bottom. These geometric coordinates are fed to the neural network, which again outputs joint targets for the robot. The robot scored an error of 0.006 after 2000 training iterations on 2500 tensors. Validation tests show that the robot was able to generalize to obstacles that the network had not trained on, while maintaining the preferred avoidance strategy depending on which zone the obstacle was located in (figure 5b).

Figure 4
digital dataset generation for linear motion experiment (a) and obstacle with respect to robot pose experiment (b)



Haptic Tuning

The final set of experiments uses physical inputs to tune hyperparameters relevant for the task. Two key parameters are focussed upon: offset distance from the target and preferred tool orientation. Variations in these hyperparameters are then demonstrated through the three cases shown in figure 6. Determining the values for hyperparameters is achieved entirely through MCM input. A small number of passes moving around a given obstacle is sufficient, although this is dependent on the complexity of the element carried in the manipulator. In our experiments an average of 4 passes are made on 3 different obstacle locations. Reaction zones are defined on the basis of these passes and their associated data. This tunes the geometric digital dataset input, and allows for the network to train for this specific task.

CONCLUSIONS

In this research we propose a new experimental motion planning method for cobots. Haptic Learning utilises Neural Networks coupled with a combination of digital data generation as well as haptic recording for the compiling of the training dataset. This combination allows for the generalisation of data gained via haptic training to different conditions. We report the parameters necessary for the development of the training within Grasshopper programming environment, and the results of initial experiments.

The method we introduce reduces the burden of programming, and enables faster adaptation as the NN is able to predict new paths faster than they could be modelled. This is important for environments where configurations or deployments of obstacles are not fully known a-priori. It represents a low barrier of entry to generate new task specific avoidance routines - only the manual guiding of the robot around an obstacle is required. The method is fully integrated within the design environment and parametric programming workflows, and operates with a precision suitable for many architectural assembly or creative collaboration tasks.

Further development will improve the method for autonomous generation of robotic complex motion path directly from the design environment, which can be either modeled, or sensed through vision feedback. Beyond obstacle avoidance, we see a larger role for NN in architectural robotics, particularly because they give a means to respond to customization and content variation of manufacturing information. It allows to shift from complex explicit calculations to approximations and predictions that

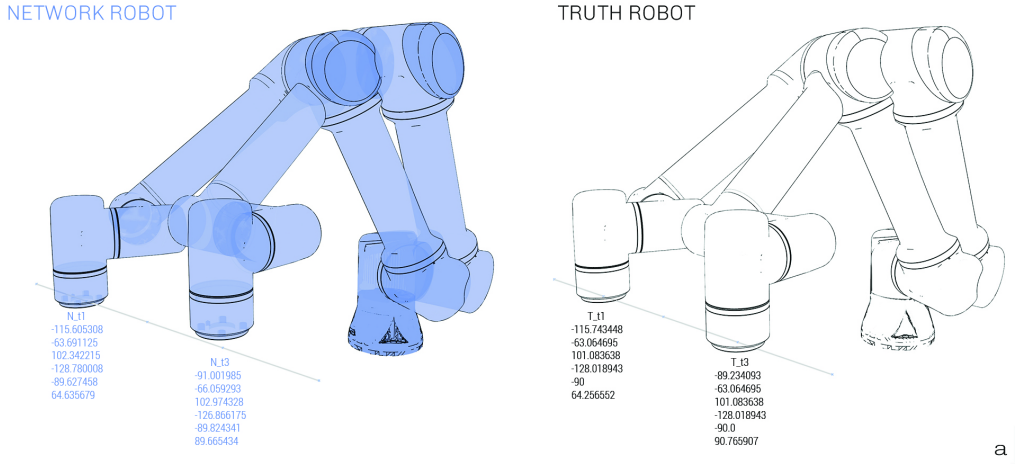


Figure 5
Joint configuration output of the Neural Network compared to the output of the simulation by Robots plugin for the linear motion experiment (a) and obstacle with respect to robot pose experiment (b)

Figure 6
Robot adaptation to different hyperparameters input through Haptic Learning: a different clearance offset between left and middle, and a different avoidance orientation on the right.

are fast and easy to achieve. They directly reduce the problem of lag, and therefore enable creative robot users to experiment with complex fabrication.

ACKNOWLEDGMENTS

We would like to thank Ayoub LHarchi and CITAstudio students for the exchange that sparked the discussion of this paper, during the "Introduction to Robotics" workshop.

Image credits. Figures 1-6: Gabriella Rossi, 2019

REFERENCES

Bloss, R 2014, 'Robots have come to architecture to model, construct, fabricate and offer new approaches to create innovative designs, elements and structures', *Industrial Robot: An International Journal*, 41(5), pp. 403-407

Bonwetsch, T 2015, *Robotically assembled brickwork: Manipulating assembly processes of discrete elements*, Ph.D. Thesis, ETH Zurich

Braumann, J, Stumm, S and Brell-cokcan, S 2016 'Towards New Robotic Design Tools: Using Collaborative Robots within the Creative Industry', *ACADIA // 2016: POSTHUMAN FRONTIERS: Data, Designers, and Cognitive Machines*, Ann Arbor, pp. 164-173

Brell-Cokcan, S and Braumann, J 2010 'A New Parametric Design Tool for Robot Milling', *Proceedings of ACADIA 2010*, pp. 357-363

Gandia, A, Parascho, S, Rust, R, Casas, G, Gramazio, F and Kohler, M 2018 'Towards Automatic Path Planning for Robotically Assembled Spatial Structures', *Robotic Fabrication in Architecture, Art and Design 2018*, Zurich, pp. 59-73

Iturrate, I, Hallundbæk, E, Østergaard, M and Savarimuthu, TR 2017 'Learning and correcting robot trajectory keypoints from a single demonstration', *Control, Automation and Robotics (ICCAR), 2017 3rd International Conference*, pp. 52-59

Menges, A 2012 'Morphospaces of robotic fabrication', *Robotic Fabrication in Architecture, Art and Design 2012*, pp. 28-47

Nicholas, P 2018, 'Fabrication for Differentiation: Towards an Adaptive Material Practice', in Daas, M and Wit, AJ (eds) 2018, *Towards a Robotic Architecture*, ORO Editions

Peshkin, MA, Colgate, JE, Wannasuphoprasit, W, Moore, CA, Gillespie, RB and Akella, P 2001 'Cobot Architecture', *IEEE Transactions on Robotics and Automation*, pp. 377-390

Rossi, G and Nicholas, P 2018 'Re/Learning the Wheel: Methods to Utilize Neural Networks as Design Tools for Doubly Curved Metal Surfaces', *ACADIA // 2018: Recalibration. On imprecisionand infidelity.*, Mexico City, pp. 146-155

Scheinman, V, McCarthy, J and Song, J 2016, 'Mechanism and Actuation', in Siciliano, B and Khatib, O (eds) 2016, *Springer Handbook of Robotics*, Springer

Stumm, S, Braumann, J and Brell-Cokcan, S 2016, 'Human-Machine Interaction for Intuitive Programming of Assembly Tasks in Construction', *Procedia CIRP*, 44, pp. 269-274

Stumm, S and Brell-Cokcan, S 2018 'Haptic Programming', *Robotic Fabrication in Architecture, Art and Design 2018*, pp. 44-58

Stumm, S, Brell-Cokcan, S and Feldmann, M 2019, 'Robotik im Stahlbau 4.0: Von der digitalen Planung zu Produktion und Bau', in Kuhlmann, U (eds) 2019, *Stahlbau Kalender 2019: Verbindungen Digitales Planen und Bauen*, Wiley

Svilans, T, Tamke, M, Ramsgaard Thomsen, M, Runberger, J, Strehlke, K and Antemann, M 2019, 'New Workflows for Digital Timber: Innovative Techniques of Representation in Architectural Design', in Bianconi, F and Filippucci, M (eds) 2019, *Digital Wood Design*, Springer

Welschehold, T, Dornhege, C and Burgard, W 2017 'Learning mobile manipulation actions from human demonstrations', *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, pp. 3196-3201

Wu, K and Kilian, A 2018 'Designing Natural Wood Log Structures with Stochastic Assembly and Deep Learning', *Robotic Fabrication in Architecture, Art and Design 2018*

Zou, AM, Hou, ZG, Fu, SY and Tan, M 2006 'Neural networks for mobile robot navigation: a survey', *International Symposium on Neural Networks*, Berlin, pp. 1218-1226

[1] https://www.robotsinarchitecture.org/kuka-prc

[2] https://www.food4rhino.com/app/hal-robot-programming-control

[3] https://github.com/visose/Robots

[4] https://ompl.kavrakilab.org/

[5] https://www.ros.org/

[6] http://www.coppeliarobotics.com/

[7] https://block.arch.ethz.ch/brg/tools/compas-computational-framework-for-collaboration-and-research-in-architecture-structures-and-digital-fabrication

[8] https://www.food4rhino.com/app/owl