



A Five-axis Robotic Motion Controller for Designers

Andrew Payne
Harvard University

ABSTRACT

This paper proposes the use of a new set of software tools, called Firefly, paired with a low-cost five-axis robotic motion controller. This serves as a new means for customized tool path creation, realtime evaluation of parametric designs using forward kinematic robotic simulations, and direct output of the programming language (RAPID code) used to control ABB industrial robots. Firefly bridges the gap between Grasshopper, a visual programming editor that runs within the Rhinoceros 3D CAD application, and physical programmable microcontrollers like the Arduino; enabling realtime data flow between the digital and physical worlds. The custom-made robotic motion controller is a portable digitizing arm designed to have the same joint and axis configuration as the ABB-IRB 140 industrial robot, enabling direct conversion of the digitized information into robotic movements. Using this tangible controller and the underlying parametric interface, this paper presents an improved workflow which directly addresses the shortfalls of multifunctional robots and enables wider adoption of the tools by architects and designers.

Keywords: robotics, CAD/CAM, firefly, direct fabrication, digitizing arm.

1 Introduction

There are literally thousands of different applications currently performed by industrial robots. With more than one million multifunctional robots in use worldwide, they have become a standard in automation (Gramazio 2008). The reason for their widespread use lies in their versatility; they have not been optimized for one single task, but can perform a multiplicity of functions. Unlike other computer numerically controlled (CNC) machines - which are task-specific - robots can execute both subtractive and additive routines. Among other operations, they can load, unload, place, cut, bend, stack, spray, weld, and mill.

However, industrial robots were not designed to be 'user friendly' – their size, support infrastructure, and programming demands make them ill-suited for untrained operators. They can also be dangerous. Industrial robots are often kept in isolated areas in order to protect human workers. In fact, many robots are painted 'security orange' (RAL color reference 2003) to "remind us that this complex piece of heavy machinery is not particularly observant of misplaced hands or feet and requires caution when operating." (Edgar 2008). Lastly, the design-to-fabrication workflow for industrial robots has traditionally been a slow and cumbersome process (see Section 3). Machine tooling, kinematic simulations, and robotic movement programming often require intimate knowledge of scripting and manufacturing processes, all of which limit the utilization of such tools by the architect/designer.

Despite considerable advances in digital software used to control robots, there often remains a detachment between the designer and the final output. Bringing physical input and output closer together through purpose-built tools for fabrication can enable many new creative opportunities for designers. Working from observations about the way architects design, this paper presents a prototype for a 3D drawing tool that takes realtime input and translates that into machine code for robotic fabrication. The purpose of this paper is not to suggest that the proposed workflow is a ready-made solution to replace existing fabrication process; rather the work should be seen as a proof of concept that could enable wider use of digital fabrication tools by architects and designers. Finally, the paper concludes with a look at the results and limitations of the proposed system and outlines a number of considerations for future development.

2 Related Work

The fields of digital fabrication and Tangible User Interface (TUI) design have seen a dramatic increase in activity in recent years. However, despite advancements made in both fields, there have only been a handful of projects which bridge the gap to embody direct physical input and output. John Frazer's *Flexible Intelligent Modeling System* is a notable early example which was developed in the early 1980s as a response to the existing CAD systems which at that time were clunky and cumbersome (Shaer 2010). Frazer and his team developed a '3D modeling system' where users built objects by stacking up sensor embedded cubes or blocks into various configurations. The computer then deduced location, orientation, and type of each component in the system and output a 2D representation of the pattern to a plotter.

More recently, there have been a number of projects which have explored various gestural interfaces as a form generator for digital fabrication. One notable example is the *Sketch Furniture* system developed by Front Design which uses a professional motion-capture system to create a new way to materialize free hand sketches (Front 2006). Designers can create 3D digital geometry by drawing lines in physical space. This data is then processed in the computer and converted into a mesh which can be fabricated at full scale using a Selective Laser Sintering (SLS) process. In another example, the *Spatial Sketch* application uses a stereo-vision 3D input system to capture gestural movements which can be translated into a series of 2D profile curves to be output to a laser cutter for final fabrication (Willis 2010). Finally, the *Shaper* prototype developed at Carnegie Mellon University in 2010 uses touch-screen technology to interactively control a three-axis CNC machine which deposits expanding polyurethane foam material according to the user's finger placement (Willis 2011). While these works offer unprecedented fluidity between the design interface and final output, they often require a significant investment

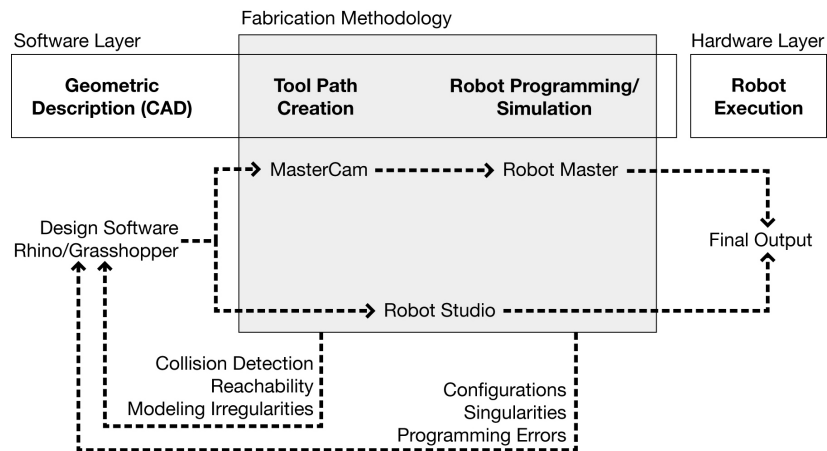


Fig. 1

in time and equipment in order to be used effectively. Additionally, none of the projects mentioned above have been designed specifically for use with multi-functional robots. Making an inexpensive and intuitive gestural interface for robotic control became the primary driver for many of the design decisions made during the development of this project.

3 Existing Design-to-Robotic Fabrication Workflow

Despite significant progress made in CAM software over the last decade, the existing design-to-fabrication workflow can be difficult to traverse for architects and designers. In the traditional sense, design conception typically occurs as a 2D sketch which is then converted into a 3D CAD model following one of two trajectories. The architect or designer can build a 3D CAD model directly from the sketches; refining certain design parameters as the idea becomes more fully developed. Or, an alternative is to create a physical model from which a 3D model can be captured through some form of 3D scanning equipment such as a digitizing arm. A digitizing arm is a device which has several degrees of freedom and uses sensors to measure the angle of each joint. From this data, the location (or coordinate) of the tool tip can be calculated. A designer can trace physical objects by moving the tool tip along the surface of an object to return a string of point coordinates which can be processed by a computer.

Once the digital model has been made, the architect or industrial designer must choose a fabrication methodology from which they will execute their design (**Figure 1**). For working with ABB robots, this means exporting the CAD model into a format (.iges, .stl, .3ds) that can be understood by the various ABB supported software applications such as MasterCAM with the Robot Master plug-in or the stand alone program Robot Studio.

While these applications offer sophisticated algorithms for tool path creation, collision detection, and singularity analysis; they are often only employed by trained operators as they require a level of manufacturing education and scripting knowledge that architects and designers typically do not have. This type of workflow introduces a level of detachment in the design process. The CAM operator, an intermediary between the designed input and final physical output, must make decisions based on prior experience in order to make the design a reality. However, if errors are found in the fabrication methodology – either from inaccuracies discovered in the CAD model or from simulation feedback – the design must be modified and the entire process begun anew. This can significantly increase production times and consequently the overall costs of the project.

4 The Tangible Controller

This paper presents a prototype for a tangible controller and a new parametric interface specifically designed to streamline the robotic fabrication process for architects and designers. The low-cost custom-made digitizing arm was constructed using the same

Figure 1. Existing design-to-fabrication workflow for ABB industrial robots

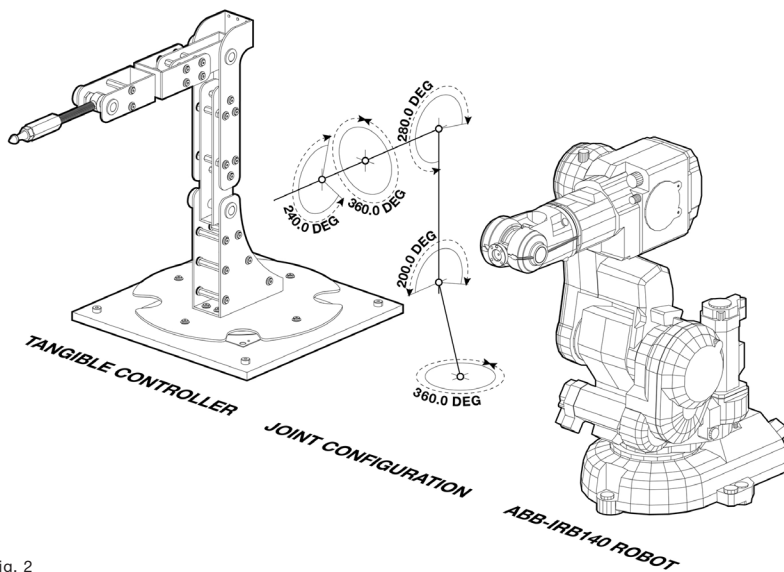


Fig. 2

joint and axis configuration as the ABB-IRB 140 industrial robot; enabling direct conversion of the digitized information into robotic movements (Figure 2).

A degree of freedom (DOF) is a geometric definition of freedom of movement either along an axis in space or rotation about an axis in space (Pottman 2007). The digitizing arm has five joints or pivot points – each corresponding to one DOF on the ABB robot. Each joint on the digitizing arm uses a high precision potentiometer to measure the angular rotation about each axis. Rotary potentiometers work by changing their resistance as the central shaft is turned to the left or right. This changes the voltage which is read by the analog-to-digital converter on the microcontroller – returning a sensor value between 0 and 1023. These particular potentiometers were able to rotate up to 340°, so the angular position (in radians) was determined using the following equation:

$$\text{Angle} = (\text{PotVal} * (1023/340)) * (\pi/180)$$

In addition to the five analog sensors (potentiometers), the digitizing arm is equipped with a tool tip circuit board with two push button controls. These allow the user to easily record or reset the digitized information on the fly (Figure 3). All of these inputs are connected to a custom designed circuit board which processes the information and sends a formatted string of information over the serial port to the virtual interface.

5 The Parametric Interface

The development of a new parametric interface was vital to enable wider adoption of robotic fabrication techniques by architects and designers. The following sections highlight some of the features of the proposed system.

5.1 GRASSHOPPER AND FIREFLY

Over the last 15 years, there has been a migration in design practices toward the utilization of parametric modeling. The term refers to a method of digitally modeling a series of design variants whose relationships to each other are defined through one or several parameters which then form a parametric space which may comprise dozens or thousands of related but distinct forms (Lagios 2010). Even though parametric modeling began as a means to develop new animation techniques in the gaming/film industries in the mid 1990s, there has been a confluence of disciplines in recent years who have embraced this type of design methodology primarily due to its ability to create designs that can quickly be adapted or modified.

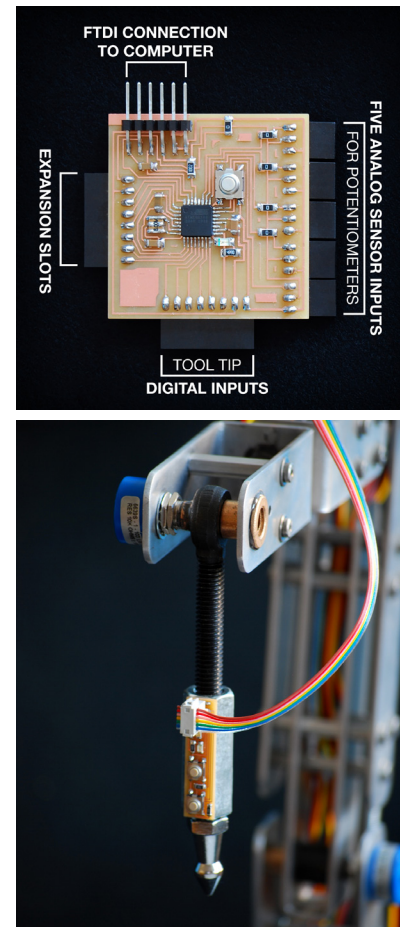


Fig. 3

Figure 2. The digitizing arm was designed to have the same joint and axis configuration as the ABB-IRB 140 robot

Figure 3. Two custom circuit boards were designed to stream the sensor information from the digitizing arm into the 3D parametric environment

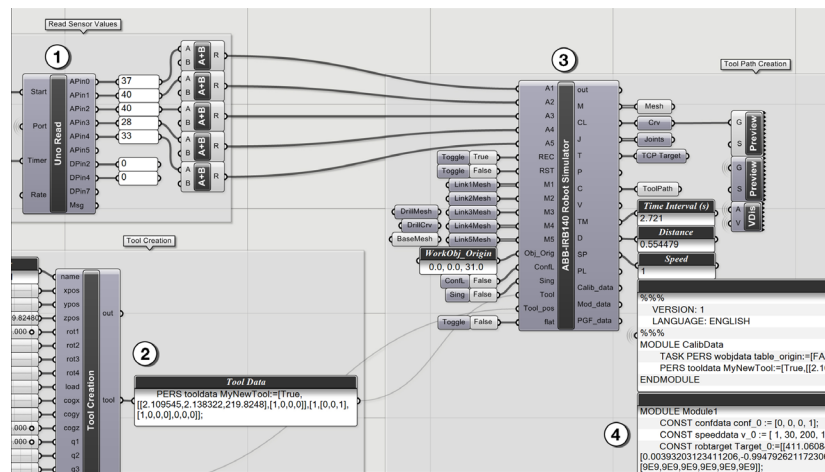


Fig. 4

One recent example is Grasshopper™ - a visual programming language developed by David Rutten at Robert McNeel & Associates which provides an intuitive parametric interface to the Rhino CAD application. In Grasshopper, programs are created by dragging components onto the editor (called the canvas). The outputs to these components are then connected to the inputs of subsequent components – creating an acyclic graph of information flow (Figure 4). Grasshopper enables a way for designers to look at design problems as a set of sophisticated relationships and to map those relationships graphically and programmatically into a system that allows them to interactively play with alternatives (Day 2009).

As a generative modeling tool Grasshopper offers unprecedented capabilities, but by default, it lacks the ability to communicate with hardware devices such as programmable microcontrollers or other haptic interfaces. Firefly is a set of tools which bridge the gap between digital and physical worlds by enabling direct serial communication between hardware devices and the Grasshopper plug-in. Firefly allows realtime access to each of the sensors mounted on the digitizing arm; forming the parametric basis for the kinematic robotic simulation.

5.2 ROBOTIC SIMULATION

Kinematic simulation of industrial robots has become an important means for the increased efficacy of robotic fabrication. Kinematics pertains to the motion of bodies in a robotic mechanism without regard to the forces/torques that cause that motion (Siciliano 2008). Since most CAM simulation tools start with the input of one or more target points (or goal objects), they often employ an *inverse kinematic* solution – or the process of determining all of the joint angles and configurations in order to reach a desired position. The reverse procedure is called *forward kinematics*. If given all of the relative angles of each joint and the lengths of each leg; the tool tip (also known as the end effector) can be found by performing a series of matrix transformations on each body in the robotic mechanism. A custom-made VB.NET script inside Grasshopper uses each of the potentiometer values from the digitizing arm to create a forward kinematic simulation of the ABB robot. As the user moves the digitizing space – tracing physical objects or creating their own custom tool paths – they can see an immediate simulation of the robot performing the same actions (Figure 5). One of the push buttons located on the tip of the digitizing arm allows users to record the tool path information which is directly translated into RAPID code – the programming language used by all ABB robots. The other push button resets or clears the recorded data.

5.3 ROBOTIC PROGRAMMING

A critical aspect of the proposed workflow was to be able to output the robotic programming codes in order to quickly test the recorded movement patterns on the larger industrial robot. RAPID is the high-level programming language used to

Figure 4. A portion of the Grasshopper definition:
1) Read sensor values via Firefly 2) Tool definition
3) Custom VB.NET component 4) Final RAPID code output

control all ABB robots and entire books have been written about how to generate your own custom programs. This section does not attempt to replicate the existing literature; rather serves to outline a few of the features which were implemented in this project.

Like other computer controlled machines, robots require a set of instructions in order to make them run properly. While there are many different data types, operators, and functions that can be included in a set of codes; there exists two main declarations to every program: 1) a statement that defines one or more target points (called robtargets) to accurately locate the end effector in space and 2) a set of movement commands that tell the robot how to move to those positions. Each composite data type requires their own set of parameters, or arguments - as seen in **Figures 6 and 7**. All of these definitions were created on the fly as the digitizing arm was controlled by the user.

A robtarget is a composite data type composed from a predefined number of values. Each robtarget is defined by its name, absolute position as XYZ coordinates, rotation or orientation of the robot as four quaternion values, joint configurations for axis# 1, 4, 6, and cfx, and finally any extra external axis configurations (**Figure 6**).

Each robtarget is given a unique identification each time the solution is recomputed. The position information can be calculated by performing the matrix transformations on all of the joints to determine the location of the tool tip in space. Quaternions provide a convenient mathematical notation for representing the orientation and rotation of objects in three dimensions. Space would not permit the full derivation of the quaternion math used in this project, but if given the roll = ϕ , pitch = θ , and yaw = ψ of the tool tip; then the four quaternion values can be found using the following equations.

$$q0 = \cos(\phi/2) * \cos(\theta/2) * \cos(\psi/2) + \sin(\phi/2) * \sin(\theta/2) * \sin(\psi/2)$$

$$q1 = \sin(\phi/2) * \cos(\theta/2) * \cos(\psi/2) - \cos(\phi/2) * \sin(\theta/2) * \sin(\psi/2)$$

$$q2 = \cos(\phi/2) * \sin(\theta/2) * \cos(\psi/2) + \sin(\phi/2) * \cos(\theta/2) * \sin(\psi/2)$$

$$q3 = \cos(\phi/2) * \cos(\theta/2) * \sin(\psi/2) - \sin(\phi/2) * \sin(\theta/2) * \cos(\psi/2)$$

The configuration data is used to identify the current quadrant of the robot axis for joints 1, 4, and 6. The last configuration number is used to select one of eight predefined robot configurations. Lastly, any external axis configurations can be supplied to locate the position of any external axis. The value 9E9 is defined for axes which are not connected.

Movement instructions tell the robot how to move from its current location to a specified robtarget (**Figure 7**). Provided with six or more DOF, the robot can move in any number of ways and the first argument in the move command specifies the type of desired movement. In this example, the robot will attempt to move linearly to the next robtarget. The velocity of the end effector is measured in mm/s and can be determined by taking the distance from the current tool location to its previous position and dividing it by the time interval between recordings. The zone data describes the position accuracy, while the tool and work objects to be used are defined elsewhere in the program.

6 Results

As of this writing, there have only been a limited number of tests that have been conducted using the five-axis robotic motion controller. However, the initial results suggest that the proposed direct-to-fabrication process could prove to be a viable alternative to existing robotic workflows. Because the digitizing arm was designed to have the same proportions and axis configuration as the larger industrial robot, it provides an intuitive interface for working with the robot. Designers immediately understand that the movements they record using the digitizing arm will be immediately translated into robotic movements, while the realtime visual feedback

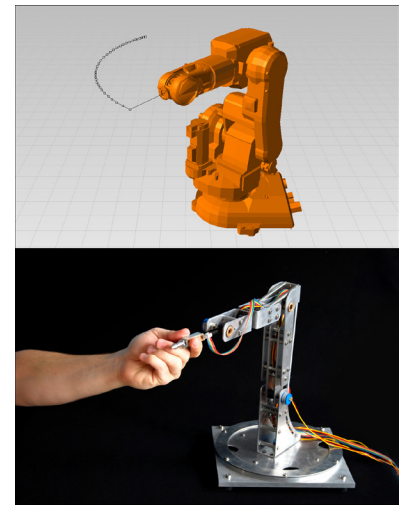


Fig. 5

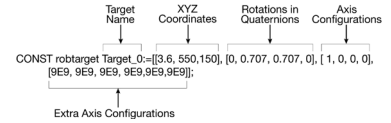


Fig. 6

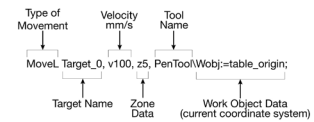


Fig. 7

Figure 5. The interface provides a fast and intuitive way to create custom tool paths or to simulate robotic manufacturing processes

Figure 6. Definition of a robot target

Figure 7. Definition of a movement command

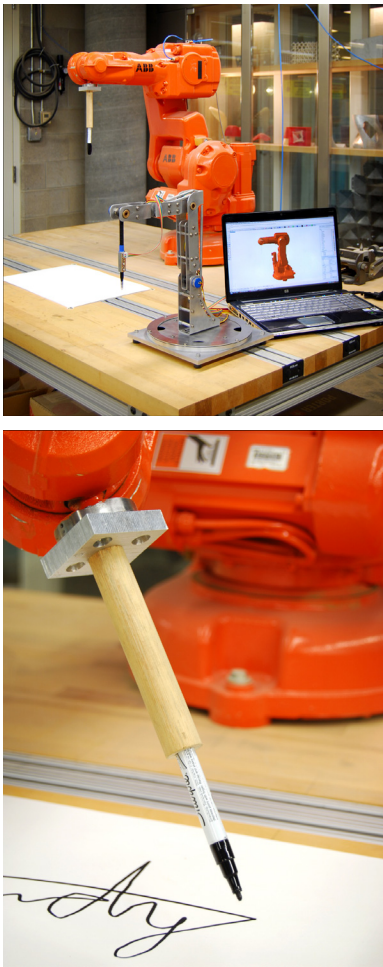


Fig. 8

Figure 8. Various end effectors can be purchased from robotic suppliers or you can make your own such as this spring loaded pen attachment

from the simulation further reinforces this fact. Given that robots can support any number of off-the-shelf or custom-made end effectors, the potential applications are limitless (**Figure 8**).

There were a number of factors in the design of the controller which contributed to the fidelity of the final robotic output and the results from these tests helped to identify areas of improvement. The decision to use high precision potentiometers, as opposed to the more accurate method of optical encoding, proved to be the primary limiting factor in the resolution of the final output. The choice was made based on costs, as potentiometers are considerably cheaper than optical encoders and the premise of the project was to provide a reliable low-cost tangible solution to robotic fabrication. However, if costs were not a factor, the fidelity of the project could be greatly enhanced by using more accurate sensors for the joint angles.

Secondly, the design of the physical prototype was a first pass at the design of a sturdy five-axis digitizing arm. During design development, some mechanical aspects of the design were discovered which could be improved upon in future iterations. Oiled bronze bearings were used for each joint on the digitizing arm, but the connection detail for joint 4, or the wrist, deteriorated to a small degree over time and became a potential source of error in the system. Likewise, the stabilization springs located on the top and back of the digitizing arm added too much tension to the system; making it somewhat difficult to move the arm into various positions. While they did provide the appropriate amount of force to leave the arm in a 90° resting position without falling over, perhaps an offset counterweight would be a more elegant solution.

7 Conclusion

This paper presents a new tangible controller and optimized workflow for robotic fabrication. Although there has been considerable progress made in the digital tools used to control robots, there is an identifiable problem in the existing design-to-fabrication process. The physical articulation of embodied input and output through purpose-built tools for fabrication can allow for wider adoption and new creative opportunities by architects and designers. In turn, this will help re-establish the relationship between designers and the physical fabrication process. The proposed system allows for a fast and intuitive way to create custom tool paths, simulate robotic kinematics, and production of RAPID code for controlling an ABB-IRB 140 industrial robot.

Acknowledgements

This project was developed as part of two graduate level courses taught at the Graduate School of Design at Harvard University and the Media Lab at MIT and could not have been realized without the tremendous support from the faculty and teaching assistants at both universities. I would like to thank Martin Bechthold, Director of the Design Robotics Group at the GSD and Neil Gershenfeld, Director of the Center for Bits and Atoms at the Media Lab for their guidance and support during the development of this project. Lastly, I would like to thank Prof. Panagiotis Michalatos for his mathematical and programming assistance in the development of the parametric interface.

References

- Day, M. (2009). Rhino Grasshopper, AEC Magazine, X3DMedia, <http://aecmag.com>.
- Edgar, B. L. (2008). A Short Biography of KR150 L110. In *Digital Materiality in Architecture* eds. F. Gramazio and M. Kohler, 49-56. Baden: Lars Müller Publishers.
- FRONT Sketch Furniture. (2006.), <http://www.designfront.org>.
- Gramazio, F., and M. Kohler. (2008). *Digital Materiality in Architecture*. Lars Müller Publishers.
- Lagios, K., J. Niemasz, and C. F. Reinhart. (2010). *Animated Building Performance*

Simulation (ABPS) - Linking Rhinoceros/Grasshopper with Radiance/Daysim, Accepted for Publication in the Proceedings of SimBuild 2010, New York City, August 2010.

Pottman, H., A. Asperl, M. Hofer, and A. Kilian. (2007). *Architectural Geometry*. Exton, PA: Bentley Institute Press.

Shaer, O., and E. Hornecker. (2010). Tangible User Interfaces: Past, Present and Future Directions. In *Foundations and Trends in Human Computer Interaction* 3 (1-2): 1–137.

Siciliano, B., and O. Khatib. (2008). *Springer Handbook of Robotics*. Springer-Verlag New York Inc.

Willis, K. D. D., J. Lin, J. Mitani, and T. Igarashi. (2010). Spatial Sketch: Bridging Between Movement & Fabrication. Paper presented at *Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied interaction*, 2010: Cambridge, MA.

Willis, K. D. D., C. Xu, K. J. Wu., G. Levin, and M. D. Gross. (2011). Interactive Fabrication: New Interfaces for Digital Fabrication. Paper presented at *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction*, 2011: Funchal, Portugal.