A FRAMEWORK USING TANGIBLE INTERACTION FOR AUTOMATICALLY CAPTURING AND EMBEDDING DESIGN INTENT IN PARAMETRIC MODELS

A Dissertation

by

EMAD KHALED AL-QATTAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Wei Yan
Committee Members,	Mark J. Clayton
	Negar Kalantar
	Philip Galanter
Head of Department,	Robert Warden

May 2019

Major Subject: Architecture

ABSTRACT

The objective of this research is to address some of the challenges of parametric design associated with defining a model's frameworks using mathematics and computer programming. This work proposes a tactile-based approach to automate the generation of such information. A design-based research method is implemented for this work, which involves developing research prototypes consisting of Tangible User-Interfaces (TUIs) to demonstrate and test the digital-physical workflow. Five prototypes were created each generating a type of information for setting up parametric models, including; linear and polynomial mathematical equations, algorithmic rules and seed configurations for a Cellular Automata (CA) component, geometric transformations (single and compound), and Non-Uniform Rational Basis Spline (NURBS) objects. During the progress of the work, prototypes were improved to include a higher level of automation by performing multiple and more complex modeling tasks.

This research includes two levels of evaluation. The first is system correctness, which tests the prototypes for translating tangible interaction with design objects into modeling information. The second is a qualitative comparison between the developed method and the conventional parametric modeling approach using graph-based and/or text-based programming applications. The results of the research have shown the plausibility of the workflow and its potential benefits for parametric modeling practice and education. This work provides a proof-of-concept for a novel approach that translates design intents into mathematical and algorithmic modeling information for

establishing parametric frameworks. The outcomes of this research include; detailed workflows describing algorithmic procedures for interpreting analog data, TUI specifications, and an overall theoretical framework of the method.

DEDICATION

To my mother Samia Alqattan and my father Khaled Alqattan.

ACKNOWLEDGMENTS

I would like to take this opportunity to thank my committee chair, Dr. Wei Yan for his knowledge, patience, and constant encouragement and support throughout this research. It is an honor and a privilege to be his student, and he will always be regarded as my mentor. I would like to thank my committee members, Dr. Mark J. Clayton, Dr. Negar Kalantar, and Prof. Philip Galanter, for their feedback and support.

Thanks go to my colleagues and faculty members at the college of architecture at Texas A&M University and to my fellow doctoral students and friends; Bara Safarova, Chengde Wu, Fatemeh Shahsavari, Hyoungsub Kim, Jawad Altabtabai, Kyeong Rok Ryu, Mehdi Farahbakhsh, Mohammad Al-Awadhi, Mohammad Rahmani Asl, Nesrine Mansour, Saied Zarrinmehr, Shermeen Yousef, Nancy Al-Assaf, and Zohreh Shaghaghian, Thanks to Dr. Stephen Caffey, Prof. Geoffrey Booth, and Dr. Valerian Miranda for their help, feedback, and support throughout my research.

I would like to also thank my family for their support and encouragement throughout my years in graduate school. My thanks go to my brother Bashar, and sisters Esraa and Rawan. I would like to take this opportunity to especially thank my beloved wife, Amal Alturkmani, for her support and patience throughout this journey. I am truly blessed and very grateful for having such a partner in life. Thanks to my daughter Layan and son Khaled for being my source of motivation and inspiration.

I would like to thank my mentors and faculty members at Kuwait University for their help, support, encouragement; Dr. Jawaher Al-Bader, Dr. Abd al-Muttalib Al-

Ballam, Dr. Muhannad Al-Baqshi, Dr. Mohammad Al-Jassar, Dr. Talal Al-Kanderi, Dr. Yasser Mahgoub, Dr. Abdullah Al-Mohaisen, Dr. Adil Al-Mumin, Dr. Omar Khattab, Dr. Mohammad Sadeq, and Arch. Abdullah Al-Awadhi.

I also would like to extend my gratitude to my mentors at The University of Michigan. I would like to thank Dr. Amy Kulper, Prof. Karl Daubmann, and Prof. Malcolm McCullough at Taubman College; and Dr. Diann Brei at the College of Engineering. A special thanks go to my dear friends whom I had the pleasure of knowing and forming a lasting relationship with; Christopher Byerly, Han-Yuan Tsao, Jawwad Naki, and Rizkallah Chaaraoui.

I also would like to thank my friends and colleagues at the Ministry of Public Works of the State of Kuwait; Ali Alharbi, Bader Bosakher, and Mishari Al-Tulihi.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Dr. Wei Yan (Chair), Dr. Mark J. Clayton, and Dr. Negar Kalantar of the Department of Architecture and Professor Philip Galanter of the Department of Visualization.

All of the work for the dissertation was completed by the student independently.

Funding Sources

Graduate study was financially supported by Kuwait University from Fall 2014 to Spring 2019.

NOMENCLATURE

AEC	Architecture, Engineering, and Construction
API	Application Programming Interface
BIM	Building Information Modeling
CA	Cellular Automata
CAD	Computer Aided Design
CPU	Central Processing Unit
CS	C-Sharp
CSV	Comma-Separated Value
DoF	Degree of Freedom
EA	Evolutionary Algorithm
FDM	Fused Disposition Modeling
GUI	Graphical User Interface
HCI	Human-Computer Interaction
IDE	Integrated Development Environment
ML	Machine Learning
NURBS	Non-Uniform Rational Basis Spline
PLA	Polylactic Acid
RAM	Random-Access Memory
ROM	Read-Only Memory
TUI	Tangible User Interface

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

Parametric design is broadly defined as a method of applying algorithmic thinking for establishing geometric dependencies (Woodbury, 2010; Jabi 2013). Geometric relationships can be established using analog and digital techniques. An early example of analog parametric modeling is seen in Antoni Gaudí's upside down model of the Colònia Güell chapel. The design is composed of a vault structure made with strings and birdshots. The geometric composition of this chapel can be altered by changing the length of the strings and position of the weights. Later, Ivan Sutherland introduced *Sketchpad*, a software tool based on propagation mechanisms and a relaxation method of a simultaneous equation solver (Sutherland, 1963). Sketchpad was considered as a breakthrough in constraint modeling for creating technical and artistic drawings (Woodbury, 2010). The two examples demonstrate two distinct parametric design approaches; i.e., Gaudí's analog approach using visual and haptic senses, and Sutherland's digital constraint solver.

Current development in digital technologies has transformed the architectural practice (Salim et al., 2010) making it possible for designers to experiment with generating architectural form expressively. The common practice of constructing parametric models is done through text-based (imperative) and graph-based (declarative) programming applications (Appleby & VandeKopple, 1997; Davis, 2013). Nevertheless, designers find it challenging to conceptualize forms algorithmically (Woodbury, 2010) as they are required to adapt to a new way of thinking that involves utilizing

mathematics and software programming to translate design knowledge into explicit algorithmic procedures. Instead, designers tend to operate digital tools in such a way that imitates traditional analog techniques (Garber, 2014). For designers, analog approaches provide a more natural way to creatively explore, communicate, and represent design ideas (Kępczyńska-Walczak, 2014; Smith, 2004).

1.1. Research Problem

Research has shown that defining parametric frameworks is problematic for designers (Davis, 2013; Weisberg, 2008; Gerber 2007) as it requires them to possess a level of knowledge in mathematics and computer programming. Although, algorithmic editors have enabled designers and novice-programmers to create non-standard geometric forms and interactively modify them (Stavric & Marina, 2011; Issa, 2013), such tools do not "alleviate the lack of understanding about more non-visual fundamentals of programming and mathematics" (Austin & Qattan, 2016, p. 832). Designers' ability to utilize such tools does not demonstrate their level of knowledge and understanding of the fundamentals of mathematics (Ozcan & Akarum, 2001). Algorithmic education is generally challenging for students, both in theory and practice, because students have not been properly trained in the fields of computer programming and mathematics (Eckerdal, 2009; Austin & Qattan, 2016), which is also the case for architecture students (Beesley, Williamson, & Woodbury, 2006). This lack of essential knowledge creates a gap between designers' intent and action, i.e., their inability to effectively translate their design ideas into digital models.

Furthermore, research has shown that the creative design process is limited when designers lack the required skills for operating digital tools (Kępczyńska-Walczak, 2014). However, analog techniques have shown to support the digital design process as they provide designers with an intuitive approach for conceptualizing architectural form. Research claims computer input components, such as mice and keyboards, do not take advantage of designers' haptic skills in the modeling process (Ishii, 2008; Eng, Camarata, Do, & Gross, 2006), because of their limited interactive capabilities.

1.2. Research Questions

This research addresses the previously mentioned challenges of parametric design by answering the following questions.

- What are the types of user interfaces that can assist in capturing and translating design intent into parametric models?

Text-based and graph-based programming applications are commonly used for embedding design intents in digital models. Yet, this research explores a method using tangible interaction by linking Tangible User Interfaces (TUIs) with digital models to improve the modeling process by automating the generation of modeling information required for establishing parametric frameworks. Research has shown that TUIs provide an intuitive approach for designers to work with digital models, which will be further discussed in the Literature Review chapter.

- Can TUIs automate the generation of mathematical equations for establishing relationships in digital models?

This question focuses on utilizing TUIs for automating the generation of mathematical information. The equations generated by the TUIs are used for setting up modeling constraints. This work focuses on algebraic constraints, which require user-defined inputs such as variables, functions, formulas, and logical arguments for establishing more complex object relations than the standard and commonly used geometric constraints.

- How can a TUI interpret the different types of tangible interaction as algorithmic rules for creating a parametric model?

Expressing design intent in digital models requires designers to demonstrate a level of mastery in both mathematics and computer programming. Mathematical functions as previously described are used to setup constraints to establish object relations for representing a design intent in a digital model. In a constraint modeling process, the designer has an initial idea of the model's outcomes of (e.g., rotating a panel array to a specific angle by changing parameter inputs). However, algorithms are used for embedding more complex design intents and for form finding. This question investigates using TUIs for generating algorithmic rules and initial cell states (i.e., seeds) for setting up a Cellular Automata (CA) algorithm.

1.3. Research Overview

This research demonstrates a method using tangible interaction for automatically capturing and embedding design intents in parametric models. The workflows developed

in this research link TUIs with virtual modeling environments to assist designers in generating modeling information for defining parametric frameworks.

1.4. Research Objective

The objective of this study is to provide an approach for generating mathematical and algorithmic information through interacting with physical design objects instead of the common practice of only using generic computer input devices with text-based and/or graph-based programming methods. The work focuses on developing and testing a novel approach that combines tangible interaction and analog data interpretation procedures to automate the translation of physical design intents into parametric models. This includes workflows for generating mathematical equations (linear and polynomial) and algorithmic rules, detecting and applying compound and non-compound types of geometric transformations, and constructing Non-Uniform Rational Basis Spline (NURBS) objects. The prototypes developed for this work consist of a custom-made TUI that is connected to a virtual modeling environment. The prototypes will monitor and translate physical design object states into parametric models.

The outcomes of this research include:

1. Flowcharts describing the programming logic and system workflow
2. Documentation of the prototypes including photographic demonstrations of user interaction and digital responses
3. Criteria for developing TUIs for modeling applications

4. An approach to interpreting analog data for automating modeling procedures
5. An overall system framework

1.5. Research Method

This research includes three main phases: 1) literature review, for identifying the challenges of parametric modeling and locate some of the related works to this research and formulating the research questions; 2) prototyping and implementation, 3) Evaluating the outcomes of the work.

This work adopts a design-based research method, which includes four main steps as described by Reeves (2000): 1) identifying a problem, 2) proposing a solution, 3) testing and evaluating the solution, and 4) documenting the results.

Haptic-based interactive prototypes are developed and used for demonstrating and testing the workflows. This work uses similar criteria to those established and discussed by Shaer, Leland, Calvillo-Gamez, and Jacob (2004) for developing TUIs. A typical workflow consists of four main parts (Figure 1.1) as described below:

1. *User*: designer(s) interacting with the system.
2. *TUI*: an apparatus consisting of:

- An *artifact*, which is composed of design objects (a physical representation/counterpart of the digital model) and a *workbench*, a workspace defining the limits of the artifact.

- A *physical computing system*; an electrical circuit composed of microcontrollers, sensors, and actuators.

3. *Data interpretation scheme*: a set of algorithmic procedures for analyzing and processing analog data for generating modeling information.
4. *Digital model*: a geometric model representing a design problem for testing the workflow.

Physical computing is a term that refers to any form of communication between the digital and physical environments, and most physical computing systems include three main parts: 1) input (sensor), 2) transducer (microcontroller), and 3) output (actuator) (O'Sullivan & Igoe, 2004).

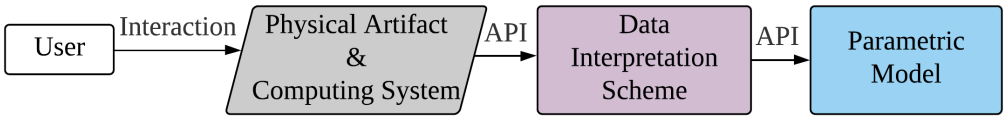

Figure 1.1 Graph showing the four main parts of the workflows developed and tested in this research. The workflow allows designers to translate design intent into parametric models through physically interacting with design objects.

1.6. Prototyping

The prototyping process includes four main steps as described below:

1.6.1. Hardware

Each prototype has a custom-made TUI, including an artifact and a physical computing system. The physical computing system provides the inputs for the digital

model through sensors that monitor the designer's interaction with the artifact. The physical computing system is composed of an Arduino microcontroller (2005), which is an open-source reconfigurable device used for a wide range of applications. The microcontroller has a CPU with limited ROM and RAM and is programmed using the Arduino IDE. The microcontroller can also be set up using other programming languages such as; Python, C-sharp, and C++. Electronics and circuit for each prototype are set up according to the inputs required for the computer algorithm. For example, Prototype 1 uses a rotary potentiometer for providing angles of rotation as data samples for regression analysis, while Prototype 3 uses pressure sensitive sensors for detecting design object location to generate the seeds for a CA algorithm. Sensors are consistent with the type of geometric transformation the designer wants to apply to a physical object. For a rotation transformation as an example, a rotary potentiometer is used, because it is operated by manually rotating the sensor's handle.

1.6.2. Software

Software applications used for this research include 1) 3D modelers (CAD and BIM authoring tools), 2) algorithmic editors, 3) data transfer and linkage, and 4) data management. Each of the five workflows includes a combination of these tools and is different from one prototype to the other. A sample workflow, as developed for Prototype 1, includes: *Revit*, a BIM authoring tool published by Autodesk; *Dynamo*, a visual programming add-on for Revit; *IronPython*, text-based programming language;

Firefly, a set of tools for data communication between Dynamo and the Arduino microcontroller (Payne & Johnson, 2012); and Microsoft Excel.

1.6.3. Digital Fabrication & Assembly

For each prototype, a design object is modeled using Rhino 3D or Revit. These objects and the overall artifacts are prepared for digital fabrication to construct the TUIs. Material type and properties, and digital fabrication machinery are taken into consideration during the 3D modeling process of the artifacts. Visual programs were written in Grasshopper and Dynamo for rapidly modifying and customizing the TUIs for each prototype.

1.6.4. Interoperability

Data communication between the TUI and the digital model is established using the software package Firefly. Other methods of data transfer and linkage were created for this research and further discussed in the Prior Work chapter.

1.7. Prototype Implementation

The prototypes are categorized into three groups: Algebraic constraints, 2) Algorithmic rules, and 3) TUI Improvements. Each of these categories tests a workflow for automatically setting up a parametric model. The independent variable in these prototypes is the physical design object's state when manipulated by the designer. The dependent variables are the generated information and digital geometric responses.

1.7.1. Algebraic Constraints

This category includes two prototypes, which focus on generating mathematical equations representing design intent (i.e., object relationships) for setting up algebraic constraints. Prototype 1 utilizes the TUI for generating linear equations for setting up the constraints in the parametric model. Prototype 2 focuses on a more complex type of object relationships based on polynomial equations. Both prototypes utilize regression models for generating the equations.

1.7.2. Algorithmic Rules

This category includes one prototype for setting up a CA component and is tested for two workflows. The first workflow is for generating the seeds, and the second for generating the rules. The workflows include a set of conditional statements to process analog data. For generating the seeds, the designer places blocks representing CA cells on a grid to create and modify a neighborhood configuration. This configuration is used as a custom seed for initiating the evolutionary process. For generating the rules, the designer defines three cell states using the blocks; alive, dead and surviving. The cell states composing the rule are generated by having the TUI counting the number of blocks placed on the grid. Changing the number of blocks modifies the rules of the algorithm.

1.7.3. TUI Improvements

This group includes two prototypes: Prototype 4, which automatically detects physical object transformations; and Prototype 5, which is used to create a NURBS object. The algorithm for Prototype 4 detects two types of physical object transformations, rotation, and translation. It identifies these transformations using incoming sensor values received from the artifact (angles of rotation or distance). These values are sent to their corresponding cells in a transformation matrix to transform the digital model. Prototype 5 uses a similar method to detect transformations, in addition to the number of design objects (representing control points) used in the artifact. This information is provided to construct a NURBS curve. The artifact and algorithm are developed to generate multiple types of algorithmic information during the process of interaction such as; 1) the number of control points for the NURBS curve and 2) boundaries for the NURBS curve. The boundaries are used afterward for generating curve configurations (design options).

1.8. Prototype Evaluation

The prototypes are tested internally by having the researcher evaluating the work for *system correctness* and by conducting a *qualitative comparison* between the developed workflow and the common practice of using text-based and/or graph-based programming methods in a parametric modeling process. *Correctness* in the context of this work refers to the correlation between data input and output, i.e., types of interaction with the generated modeling information and geometric responses.

The qualitative assessment of the work provides insight into the system in terms of its benefits and drawbacks in the parametric modeling process. Techniques used for this evaluation include using visualizations of mathematical graphs (for comparing geometric profiles generated using the mathematical equations), and examples of parametric frameworks describing the conventional process of translating design intent into mathematical and algorithmic procedures.

1.9. Research Significance

Physical models have shown to be beneficial in a digital design process. For example, Shelden (2002) emphasizes the importance of such artifacts in Frank Gehry's digital practice as they provide insight into a design's physical and material properties that would aid in further design development. Currently, development in digital technologies and the emergence of notions such *Mixed Reality* system (Milgram & Kishino, 1994), the physical *embodiment* of computation (Dourish, 2001), *ubiquitous computing* (Weiser, 1991) and *tangible interaction* (Hornecker & Buur, 2006) have promoted research in context-aware computing for architectural applications (Salim et al., 2010).

Furthermore, since the launch of Sutherland's Sketchpad (1963), there has been a growing interest in enhancing the interactive capabilities of CAD systems (Davis, 2013). Monedero (2000) mentions that "A fundamental problem in CAD is how to make explicit some intuitive knowledge we have about something in such a way that a machine can interpret and treat it in an automatic way" (p. 371).

TUIs and other forms of digital-physical workflows provide a unique platform for working with parametric models as they enable real-time interaction with digital models and instant feedback to the designer. Existing TUI examples demonstrate some of the opportunities for improving digital processes (Ishii, 2008; Salim et al., 2010) and designers' cognitive abilities (Kim & Maher, 2008a, 2008b).

This research finds that there is a need to improve digital workflows to capture design intents and embed them in digital models automatically. This research is expected to contribute to the body of knowledge by suggesting that tangible interaction can provide an intuitive approach for generating mathematical information and computer programming procedures required for defining parametric frameworks. To my knowledge, there is hardly any research investigating the prospect of utilizing TUIs for assisting designers in the process of generating such information in a parametric modeling process.

1.10. Dissertation Outline

This dissertation consists of eight chapters as described below:

- Chapter 1 – Introduction: describes the outline of the dissertation.
- Chapter 2 – Literature Review: provides the literature for the two main topics of this work, parametric *modeling,* and *tangible interaction*. This chapter focuses on describing the current methods used for digital modeling and providing examples of related work; defining the research problem; and formulating the research questions.

- Chapter 3 – Research Method: provides a detailed description of the research outline, process, and phases of implementation.
- Chapter 4 – Prior Work: describes the process of developing the TUIs and data transfer methods.
- Chapter 5 – Prototype Implementation: describes the process of testing the developed workflows and TUIs.
- Chapter 6 – Evaluation: presents a comparative analysis between the developed workflow and the current practices for embedding design intents in digital models.
- Chapter 7 – Discussion: provides a description of how the method addresses the challenges of parametric modeling, limitations of the study, and its applications in both practice and academia.
- Chapter 8 – Conclusion: discusses the contribution of the work to the body of knowledge and future work.

2. LITERATURE REVIEW

Current developments in digital tools have enabled designers to explore new ways to conceptualize form (Schnabel, 2007), which is a fundamental shift from the 1990s where such tools were mainly used for design representation (Stavric & Marina, 2011) The emergence of algorithmic editors and software packages (e.g., Grasshopper, Dynamo, etc.) have provided designers and novice-programmers with the means to create and modify "non-standard" geometric forms (Stavric & Marina, 2011, p. 9). For example, Grasshopper, an algorithmic editor and plug-in for Rhino 3D, is commonly used for architectural design (Payne & Issa, 2014), because it offers a wide range of mathematical tools, such as "operators, conditional statements, functions, and trigonometric curves" (Stavric & Marina, 2011, p. 12) for creating algorithms. Such algorithms have provided designers with generative power for digital modeling (Stavric & Marina, 2011), which goes beyond the conventional and limited use of 3D modelers (Terzidis, 2006).

Parametric design is based on algorithmic thinking, a process of expressing designs through a set of procedures. It provides designers with a novel approach to digital modeling. Woodbury (2010) states that "*Parametric modeling*…introduces fundamental change: 'marks', that is, *parts of a design*, relate and change together in a coordinated way" (p. 11). The designer assigns different values to an object's parameters and modifies them to rapidly generate design variations for the same model (Woodbury, 2010; Maher, 2011; Hernandez, 2006). An advantage of parametric design is that

changes can be implemented in digital models without having the need to reconstruct them (Burry, 2011). The flexibility of a model to adapt to changes is a key feature of parametric modeling, as Davis (2013) explains, "Flexibility makes-up the central tenet of parametric modeling. By maintaining a flexible model the designer can afford to make changes, which is important given the inevitability of change on an architecture project" (p. 36).

Davis (2013) further explains, "a parametric model is unique not for what it does but rather how it was created" (p. 31). The designer explicitly states how a set of outcomes can be derived from a set of parameters (Davis, 2013). Parametric modeling suggests that parameters are used to generate form; however "what is actually in play is the use of relations" (Monedero, 2000, p. 371). Novak (1998) explains that parametric modeling is more with the manipulation of relations and less with the manipulation of geometric objects. The construction of a parametric model and the way it behaves reflects on the purpose of the model, its intent. Therefore, it is essential to discuss the relationship between parametric modeling and design intent as it provides insight into the way a model is constructed to serve its purpose.

2.1. Design Intent: Parametric Relationships

Design intent is a concept widely referred to in design-related fields; however, it is challenging to formally define (Otey, Company, Contero, & Camba, 2018; Chen & Hoffman, 1995). Several authors have provided a description of the term with respect to digital modeling. For example, Martin (2017) explains that a design intent at the most

basic level is, “What I intend my designs to do” (para. 1.2). This refers to the model’s expected behavior for achieving a required function. An extended description of design intent is provided by Otey et al. (2018) in the context of CAD modeling for mechanical engineering. Although Davis (2013) mentions that the uniqueness of parametric models is in the way they are constructed. i.e., relating functions to outcomes, a design intent mostly focuses on the model’s behavior and flexibility to adapt to change when modified.

Martin (2017) mentions that the objectives of design intent in CAD modeling are creating “parametric, flexible, and robust models that update in ways we plan and expect when we implement changes” (para. 1.3). Software tools such as CATIA and SolidWorks (Dassault Systèmes, 1995); Digital Project (Gehry Technologies, n.d.) Cero (PTC, 2011), etc. follow a similar process for constructing parametric models that include creating a 2D sketch and adding constraints and parameters to the geometric objects composing the model to establish relationships. A design intent in this process is captured by embedding mathematical equations using the constraints. Another approach is by expressing design intent through explicit algorithmic procedures such as the pattern examples provided by Woodbury (2010) in his book *Elements of Parametric Design*.

In a parametric modeling process, it is essential to properly prepare a framework describing the relationships between the different parts of the model to achieve the expected behavior. As Rynne and Gaughran (2007) mention that, the extent to which a design intent is captured in a parametric model relates directly to the way the model is planned and built.

A parametric model consists of constraints and parameters (Hernandez, 2006), both of which are used to establish the design space and maintain the relationships between the geometric objects (Maher, 2011). Object relationships and their behavior are defined mathematically and geometrically in digital models (Stavric & Marina, 2011) using formulas, equations, and functions (Burry, 1999).

Davis (2013, p. 21) mentions that a parametric equation must meet two conditions: it should (1) express a set of quantities (geometric objects) through a number of parameters (e.g., *x* and *y* in terms of a free parameter *t*), and (2) relate the outcomes to the parameters through explicit functions. For example, a two-dimensional circle equation can be represented mathematically in Eq. 2.1, and its parametrization as shown in Eq. 2.2 and 2.3.

$$r^2 = x^2 + y^2 \qquad \text{Eq. 2.1}$$

$$x = r\cos(t)\,, y = r\sin(t) \qquad \text{Eq. 2.2, 2.3}$$

The parametric equations of the circle show *x* and *y* as the quantities, which are made explicit in the functions in terms of an independent variable *t*, referred to as the *free parameter*. The free parameter generates a point on the circle. If Eq. 2.2, 2.3, for example, are to be used in a design context for changing the size of the circle using its radius, then both *r* and *t* are used as free parameters. The geometric entity in this example (the circle) and its behavior are expressed through functions, and the outcomes

of these functions are related to the parameters. These characteristics of Eq. 2.2, 2.3. meet the criteria that define a parametric equation.

It is essential to explain the notion of a parameter and the method of implementing equations and establishing relations between objects in digital models. A parameter in a broader sense can be described as a "boundary" that defines a "design space" (Maher, 2011, p. 10). In a parametric model, it is important that the designer carefully establishes this boundary, as "The combinations of sets of values for the parameters in each of the parameterizations are 'design space' and determine the space's flexibility" (Maher, 2011, p. 10).

Designers utilize constraints in digital models to establish object relationships. A *constraint* is a relation that limits design possibilities (Cuff, 1991) by restricting the "behavior of an entity or group of entities" (Monedero 2000, p. 372). Hoffmann and Joan-Arinyo (2002) categorize constraint types into four groups:

- Geometrical constraints
- Equational constraints
- Semantic constraints
- Topological constraints

This research focuses on *equational constraints*, also referred to as *algebraic constraints*, which utilize mathematical functions and conditional statements to link parameter values together to define object relationships. Such constraints include distance (or length), radius, and angle.

Constraints offer a way to establish a relationship in parametric models, yet other methods of constructing similar or even more complex relationships do require alternative programming methods. The following section will provide a description of some of the conventional programming methods used for parametric modeling.

2.2. Programming Methods

Weisberg (2008) reports that designers using Pro/ENGINEER have found that parametric modeling is more like programming than the conventional practice of design. Pro/ENGINEER is the first commercial parametric modeling software published in 1988 by Parametric Technology Corporation. Currently, a wider range of parametric modeling tools exists with each having distinct programming and interactive features. Davis (2013) describes these software tools in his *Taxonomy of Programming Languages* graph (p. 62), which was initially developed by Appleby and VandeKopple (1997). As Davis explains (2013), in this classification, software tools fall under two programming paradigms *Imperative* and *Declarative*. For Imperative, programming applications are text-based and are subcategorized under more specific programming paradigms, such as Object Oriented (such as Maxscript, AutoCAD.NET, MEL, Processing, Revit Python, and Rhino Python) and Procedural (such as GDL). Programming applications such as Rhino VB and Digital Project VB overlap both subcategories. For Declarative programming, applications are graph-based and are subcategorized under Functional/Dataflow programming paradigm, and it includes Grasshopper, GC, Houdini,

and MaxMsp (Davis, 2013). Dynamo, an algorithmic editor for Revit, can also be included in functional and dataflow programming.

The difference between both types of paradigms is the way the designer constructs the algorithm. For Imperative, it is mostly about the designer stating 'how' the algorithm works, the designer describes "a sequence of actions for the computer to perform" (Davis, 2013, p. 99). As for Declarative, the designer describes "what" the algorithm is going to achieve, the designer defines the results without having to explain the process (Van Roy & Haridi, 2004; Davis, 2013).

In both programming paradigms, the designer must develop a set of skill and acquire knowledge in software programming and mathematics to utilize the tools (Woodbury, 2010). This process is uncommon for designers as they are required to approach design using a different mindset than the conventional approach of using analog techniques to conceptualize form. This problem is further discussed in the following section.

2.3. Research Problem

Designers are trained in model making and drafting (Eng et al., 2006). Such techniques enable them to produce artifacts that are considered essential in the design process (Sass, 2009). These analog skills and artifacts enable designers to naturally communicate their ideas and explore design solutions (Kępczyńska-Walczak, 2014). However, parametric modeling requires designers to adapt to a new way of thinking for translating design knowledge into algorithmic procedures, which is unconventional for

them. Designers must carefully plan the sequence and precise algorithmic procedures for establishing a parametric model. Davis (2013) mentions that:

> *Planning is a necessary component of parametric modeling because the logical rigidity of a model's explicit functions requires that the designer anticipate, to some degree, the parameters of the model and the hierarchy of dependencies between functions. (p.39)*

The process of defining parametric frameworks is an essential step in the modeling process, yet it requires "a significant amount of upfront cognitive investment" (Gerber 2007, p. 205).

The process of defining parametric frameworks requires specialized knowledge in mathematics and geometry, both of which are considered as the core of the design process from the initial stages of design to manufacturing (Stavric & Marina, 2011). Therefore, developing designers' skills in software programming is essential (Aish, 2005), because it allows them to embed their design intents in digital models. Issa (2013) mentions that algorithmic editors (e.g., Grasshopper) have enabled designers with no programming background to design expressive geometric forms. Nevertheless, for designers to translate design knowledge into mathematical functions and algorithmic procedures is problematic. Lacking computing skills often results in limiting or even blocking the creative design process (Kępczyńska-Walczak, 2014), and simply operating digital tools does not necessarily reflect designers' understanding of the process. Kępczyńska-Walczak (2014) reveals in a study that, some students were not always

successful in writing computer codes for creating and controlling geometric forms. Their skillset in computing did not necessarily follow their imagination as a level of algorithmic thinking, and programming proficiency was required (Kępczyńska-Walczak, 2008). Beesley, et al. (2006) also mention that:

> *Effective use of a parametric modeler requires a practical understanding of such concepts as a vector, the cross-product, projection, parametric functions and Frenet frames. As anyone who has studied linear algebra knows, these concepts require some sophistication to master, that is to use effectively and with control…Most designers have not had much formal education in mathematics, computing or software engineering. (p. 3)*

Monedero (2000) states that the problem designers encounter with CAD systems is "How to make explicit some intuitive knowledge we have about something in such a way that a machine can interpret and treat in an automatic way" (p. 337). This research finds that there is a need to investigate and provide an approach for automating the process of capturing and embedding design intents in parametric models. The following research questions will assist in exploring alternative possibilities to current parametric modeling practices to support designers in the digital design process by generating mathematical and algorithmic information that is required for establishing parametric frameworks.

2.4. Research Questions

Research claims that operating computers using keyboards and mice do not reflect the way designers interact with objects in the physical world (Dourish, 2001). Fischer (2005) explains that with GUIs, the physical attributes of a model are lost, such as "The tactile and material qualities as well as spatial realism and the required tectonic and construction skills that used to play a much more important role in design education" (p. 59). The following question aims to investigate a physical and a natural approach for interacting with parametric models, suggesting a method that is similar to using analog techniques in the design process.

- What are the types of user interfaces that can assist in capturing and translating design intents into parametric models?

Since the launch of Sketchpad in 1963, advances in digital technology have shifted interest from "the physical to the digital" in both the practice and the education of architecture (Fischer, 2005, p. 59). Although, computing technology has enabled designers to create elaborate forms (Stavric & Marina, 2011) and functional models using simulation tools and engines (Fischer, 2005), research claims that "GUIs fall short of embracing the richness of human senses and skills people have developed through a lifetime of interaction with the physical world" (Ishii & Ullmer, 1997, p. 7). Eng et al. (2006) mention that manipulating digital objects lacks the tactile feedback that traditional models provide, which is "counter-intuitive to the designer's education" (p. 1). Research suggests that TUIs can provide an intuitive approach for digital modeling

and are being developed as an alternative to using generic computer input devices like keyboards and mice (Maher & Kim, 2005).

In addition, to the TUI's intuitive interactive capabilities, it also does support design cognition and decision making. According to Kim and Maher (2008b), TUIs have shown to have a positive effect on designers' "perception and reasoning of visuo-spatial information" (p. 248). Kim and Maher's extensive research on TUIs and design cognition supports this research's objectives, specifically for using TUIs to develop designers' level of algorithmic thinking and knowledge in computer programming.

- Can TUIs automate the generation of mathematical equations for establishing relations in digital models?

As described earlier, the common practice of establishing relations in parametric models is often achieved through text-based and/or graph-based programming applications. A challenge of parametric modeling is that it requires specialized knowledge in mathematics, geometry, and computer programming (Woodbury, 2010). Aşut & Meijer (2016) state that a challenge of teaching CAD is that, "CAD mostly requires to communicate explicit information which do [*sic*] not mostly overlap with the implicit realms of design knowledge" (p. 322).

The aim of this research is to explore an approach to translate design intent into mathematical form to be used in a digital model. This question aims at investigating a workflow using tangible interaction to automate the generation of mathematical equations depicting object relationships. TUIs do provide an intuitive approach for

parametric modeling; designers can manually manipulate physical objects to represent a relationship instead of explicitly stating it through mathematical functions.

Existing works utilizing tangible interaction and user-interfaces do demonstrate the uniqueness and opportunities for improving the design process. These works mainly focus on gathering analog data from users and the surrounding environment as parameter inputs to control and manipulate digital objects. Such examples link TUIs with defined parametric frameworks, i.e., the designer establishes an object relationship using programming tools. Salim et al. (2010) state that, in their prototype demonstrations, an existing parametric framework is used with the TUIs. They suggest that, in their future work, physical computing systems can be used to define object relationship in digital environments.

Furthermore, software packages and mathematical modules such as Math.Net Numerics (Ruegg, Cuda, & Gael, 2002) can assist in this workflow. It includes regression models for generating mathematical equations that depict correlation in data sets. This module can be used with software programming applications such as C-Sharp and Python. Another example is the *Trendline tool* in Microsoft Excel, which uses a curve-fitting function to evaluate a set of data points. Trendline generates best-fit curves using linear, polynomial, logarithmic, and other types of equations. For digital modelers such as Rhino 3D, designers can find best-fit curves for a set of data points using the Grasshopper plug-in Mantis (Zaghloul, 2010).

Utilizing the previously mentioned software tools do offer the possibility of discovering the mathematical definition of geometric elements and patterns within data

sets. This research finds that incorporating these tools in the digital-physical workflow provides an innovative approach for automatically deducing physical object relationships and representing them as mathematical equations.

- How can a TUI interpret the different types of tangible interaction as algorithmic rules for creating a parametric model?

Research has shown that computer programming can be challenging for designers, because they are expected to learn, create, and utilize their algorithms in a short period of time (Austin & Qattan, 2016). As explained previously, most designers have not had any formal training in computer programming or developed a basic understanding of programming principles. Lahtinen, Ala-Mutka, and Järvinen (2005) further explain, software programming, in general, can be challenging because of users' difficulty to 1) understand some of the abstract notions of programming, 2) construct algorithms, and 3) envision algorithms' real-world applications.

This research tests a workflow integrating tangible interaction and a CA algorithm for generating complex geometric patterns. CA rules, similar to object relationship, can be easily described using natural language, because of its clear algorithmic grammar (i.e., cell state being alive, or dead based on the number of its surrounding neighbors) and its simple rules, which can be used as parameter inputs. These characteristics of CA makes it straightforward to use and to generate outputs using the rules rapidly. However, the process of "formulating and elaborating the rules are more difficult" (Araghi & Stouffs, 2015, p. 154). CA provides a good example for

testing the method of generating rules using tangible interaction for reasons described below.

CA is a generative algorithm represented by an infinite lattice with each of its cells having one possible state; a cell can be alive or dead in a two-dimensional grid or have more possible states if it is generated in three-dimensions (Frazer, 1995). In a 2D CA, a cell's livelihood in later generations of the evolutionary process is determined by its eight surrounding neighbors (referred to as a neighborhood). In 1963, von Neumann introduced CA, and a popular example of it is shown in John Horton Conway's *Game of Life*. The game only requires the player to input an initial cell state (seed) to start the evolutionary process. The game produces emergent patterns that resemble the behavior of living organisms (Gardner, 1970; Krawczyk, 2002). This complex behavior (i.e., patterns) is generated by using simple "local" rules, while the overall pattern is affected by the seed (Frazer, 1995, p. 54).

CA has been extensively researched as an architectural design method (Cruz, Karakiewicz, & Kirley, 2016) because the characteristics of the generated patterns can be interpreted as spatial configurations used in conceptualizing architectural form (O'Sullivan & Torrens, 2001; Herr, 2008). Herr (2008) mentions that:

> *Two – or three-dimensional CA-generated patterns seem related to architectural design at several levels… they can be a reminiscent of urban or architectural plans or building form. CA further depend on procedural rule-based logic, which can be related to rules governing architectural composition or functionality used in architectural design*

> *processes. CA are based on spatial relationships between cells, which usually refer to some form of cell "neighborhood." Finally, relationships between CA cells have a temporal and procedural dimension, which can be linked to the gradual development of design proposals during the architectural design process. (p. 5)*

An early example of the application of CA rules and physical computing systems can be traced back to work developed by John and Peter Frazer. In 1979, they created a Self-replicating Cellular Automata model, which is composed of cells integrated with electronics. The electronics enable each cell to "know the rules of self-replication, to be aware of its neighbors, and to display with LEDs the addition point of the next cell" (Frazer, 1995, p. 56). The rules control the growth of the model, and the lights used in the model indicate the location of the added cells by "human intervention or by a robotic arm controlled by the system" (Frazer, 1995, p. 56).

This research question utilizes CA in the workflow as a sample algorithm. The work focuses on setting up CA by automatically generating the seeds and rules using tangible interaction. This approach like the previous question focuses on providing a natural and tactile-based approach for generating algorithmic information required for establishing a parametric model. A widely used software package for implementing evolutionary algorithms in digital models is the Grasshopper plug-in *Rabbit* (Morphocode, n.d.). It is used in this workflow, because it offers a set of comprehensive tools to utilize CA, and other Evolutionary Algorithms (EA) such as L-systems, in a digital modeling process.

2.5. Tangible Interaction

Prior to digital modeling, analog techniques were used for form finding, such as the example of the hanging chain model of the Colònia Güell chapel by Antoni Gaudí's (Burry, 2011). The chapel's structure is composed of vaults made of strings and birdshots (Burry, 2007; Woodbury, 2010). The chapel represents all the attributes of a parametric model including independent variables, which can be manipulated for generating different form configurations. Davis (2013) states that "A hanging chain has at least four parameters: its length, its weight, and the two points it is attached to" (p. 22). Each chain in the model creates a curve when hung, and this curve is an explicit function of gravity. Gaudí was able to generate different design configurations for the chapel all of which are under compression by changing the parameters of the model. These results are automatically generated without having the need to calculate them manually (Davis, 2013).

Parametric modeling using analog techniques provided a tool for form finding, which was further explored in Otto's work (Otto & Rasch, 1996). These examples demonstrate an approach using tangible interaction and a computing method using physical models. However, such models can only generate design options for a specific set of functions (Davis, 2013). In 1963, Ivan Sutherland launched Sketchpad, which was considered the first parametric software tool (Davis, 2013). Sketchpad was a breakthrough in constraint modeling, which was used for creating technical and artistic drawings (Woodbury, 2010). It had a unique interactive system using a light pen to draw

on the TX-2 computer's console. Sketchpad and its method of interaction were revolutionary as they allowed for "a man and a computer to converse." (Sutherland 1963, p. 8).

An essential part of any parametric model is its ability to adapt to change. Sutherland created Sketchpad to accommodate change, e.g., "A designer using Sketchpad could change their [sic] mind about the relationship between objects (the critical part) and Sketchpad would automatically update the objects (the related parts to satisfy this relationship)" (Davis, 2013, p. 4). The two constraint solving methods used in Sketchpad are *Relaxation* and the *One-pass*, the first uses numeric optimization and the second analytical solving of explicit functions (Sutherland, 1963). However, only the second method is considered as a parametric feature of Sketchpad, because it deals with explicit functions (Davis, 2013).

Both examples of work, by Gaudí and Sutherland, demonstrate a significant approach to parametric modeling using distinct interactive methods, one being analog and the second being digital. Each takes advantages of its medium to support designers in the modeling process. In Gaudí's example, the designer can use a physical model that is derived from physical laws, and in Sutherland's software, it takes advantage of the computer's computing power.

Physical models, however, do have their limitations in a digital design process. For example, it is difficult in most cases to demonstrate or to deduce the underlying parametric framework that leads to a specific design solution using physical models; Fischer (2005) states that:

> *Traditional physical models, while possessing some advantages over digital models in allowing assessment of physical interaction-related criteria, are understandably not always very useful in answering questions that address issues of process such as interaction performance and sequential logic. (p. 59)*

Another example of physical models' limitation, when compared to digital methods in a parametric modeling process, can also be seen in Gaudí's inverted model, the chapel is designed to solve a single parametric equation, unlike Sketchpad which can solve any parametric equation (Davis, 2013).

Garber (2014) mentions that designers use digital tools in such a way that imitates conventional analog techniques. In Sutherland's example, it is evident that the user is interacting with Sketchpad in a similar way to using a pen and a piece of paper to draw geometric objects, yet unlike CAD tools, it takes advantage of computing in the digital process. Sutherland (1963) further explains:

> *The major feature which distinguishes a Sketchpad drawing from a paper and pencil drawing is the user's ability to specify to Sketchpad mathematical conditions on already drawn parts of his drawing which will be automatically satisfied by the computer to make the drawing take the exact shape desired. (p. 110)*

Both analog and digital tools offer unique benefits for parametric modeling. Research suggests that there is a need to "integrate the real world and the digital

information space for designing parametric models" (Salim, Mulder, & Burry, 2011, p. 133). Embedding computing in physical objects can take advantage of both mediums, the digital and physical in the parametric modeling process.

2.5.1. Digital-Physical Workflows and TUIs

Sears and Jacko (2007) mention that haptic-based user interfaces are a new frontier of media technology, which has great potential in contributing to human life. The implementation of such technology can be of great value for supporting designers creative process. Kim and Maher (2008a) conducted a comparative study between the effects of TUIs and GUIs on designers' spatial cognition, concluding that "TUIs change designers' spatial cognition and these changes are associated with the creative process" (p. 26). Kim and Maher's (2008a) study focused on the *epistemic* actions and their effects on designer's cognitions by reducing cognitive loads using physical objects. Epistemic refers to the motor active exploration of information that is challenging to compute manually (Fitzmaurice, 1996). Hornecker and Buur (2006) state that, "Tangible User Interfaces (TUIs) and Tangible Interaction are terms increasingly gaining currency within HCI. This field of research relies on tangibility and full-body interaction and gives computational resources material from" (p. 437). Tangible interaction is defined by Hornecker and Buur (2006) in an inclusive manner around a broad range of systems and user interfaces, which "encompasses approaches from HCI, computer science, product design, and interactive arts" (pp. 437-438). User interfaces can be organized into three categories; "text-based, graphical, and emerging user interfaces" (Razzaq, Qureshi,

Memon, & Ullah, 2017, p. 466). Shaer and Hornecker (2010) provide an extensive study on TUIs and a comprehensive accumulation of TUI models and application. Hornecker (2005) explains that TUIs are categorized in literature into three groups:

- Data-centered view
- Perceptual-moto-centered view
- Space-centered view

The *data-centered view*, which is pursued in the fields of computer science and HCI (Hornecker, 2005), is a type of TUI where "Tangible interaction is about physical representation of digital functions and data, or of rather physical objects" (Hornecker & Buur, 2006, p. 441). The suitability of the representational is significant in such systems as they provide users with comprehensible objects that would facilitate the interaction (Ullmer & Ishii, 2000). In haptic-based systems, physical objects are useful to users and intuitive if they are expressive and are considered as an essential part of the TUI (Eden, Scharff, & Honecker, 2002). More precisely, Hornecker and Buur (2006) state that, the representation must be:

> *A salient part of the representation (e.g. emphasizing 3Dness or material qualities) or in effecting the style of interaction. Thus, these interactions should not be peripheral, but need to be salient to the overall use process…legibility of system reactions and experience of the system as being hybrid are enhanced by perceived coupling between physical objects and digital representations and between user actions and effects. (p.442)*

McNerney (2004) provides an overview of the development of TUIs at MIT which are mainly used for educating students in computing. The list of works McNerney discussed was included from the mid of the 1960s with the groundbreaking work of Seymour Papert and his students at MIT and elsewhere, which has "provided a rich body of research into the field of educational programming and tangible user interfaces" (McNerney, 2004, p. 336).

The conceptual framework provided by Fitzmaurice, Ishii, and Buxton (1995) described in their *graspable user interfaces* provided a methodological approach and the foundation for developing TUIs. A graspable user interface uses physical objects as "input devices that can be tightly coupled or 'attached' to virtual objects for manipulation or for expressing action" (Fitzmaurice et al. 1995, p 442). The graspable user interface is evolutionary in the sense that it complements the GUI, taking advantage of both computing and human skills (Fitzmaurice et al., 1995). Ullmer and Ishii (2000) further expand on this framework in their study by exploring the characteristics of TUIs and different methods of interaction.

2.6. Related Work

Ishii and Ullmer (1997) have demonstrated three TUI examples (metaDESK, transBOARD, and ambientROOM). The first two examples utilize foreground objects on interactive surfaces. The metaDESK example is related to this research as it shows a system for users to interact visually and haptically with digital environments: Ishii and Ullmer's physically represented GUI elements in the TUI, e.g., using a flat-panel for

physically representing a *window* on a computer screen, or a *phicon* (physical icon) for representing a computer *icon*. In this example, GUI elements are given a symbolic physical form for users to interact with. The aim of their work is to free the user form GUIs' limited control system, providing a more intuitive approach of interaction. The objective of developing and using TUIs is to bridge the gap between digital and physical environments (Ishii & Ullmer, 1997). Such an approach has been further developed and implemented in the context of design and geometric modeling. For example, the work by Eng et al. (2006), provides a different approach for developing TUIs using a *hub* and *strut* construction kit, referred to as FlexM. Enhancing the kit using computation allows the user to explore digital geometry interactively by manually creating the different geometric configuration. These configurations are directly displayed on the computer screen using VRML (Virtual Reality Modeling Language), and later using FormWriter.

In the previous examples, the TUI was used for geometric modeling; the artifact assisted the designer in exploring and creating 3D graphical representations. Salim et al. (2010) however, provide a novel approach utilizing TUIs for parametric modeling, which focuses on analyzing physical data to "capture relations and interactions that exist in the physical world as parameters" (p. 380). The interactive experiments demonstrated by Salim et al. (2010) include *Ur-moeba*, a tangible user-interface for collaborative parametric modeling; and Rapid Design Coordination (RDC), which is for parametric design and construction coordination. Both examples have a similar setup using video cameras, Processing IDE, and reacTIVision TUIO for detecting fiducial markers. The table top setup is linked to GC (Generative Components) using *UbiMash* (a generic

interoperability software tool for connecting hardware with CAD systems). Their work shows dynamic and interactive systems for parametric modeling. Ur-moeba demonstrates the potential of a TUI to facilitate "direct feedback on a complex simulation process," and RDC is an intuitive approach using a TUI to deal with "coordination and change management in various stages of building design to construction" (Salim et al., 2010, p. 396). Further exploration of this approach is shown in the work developed by Salim et al. (2011) where they connected parametric models in Rhino 3D to a Wiimote controller. The controller in this example is intended for exploring the "potential of mapping of the user's embodied space onto a 3D model" (Salim et al., 2011, p. 138). The same controller was used in a later experiment with GC to reform a parametric surface, to draw BsplineCurves, and to act as a camera view controller. The experiments with the controller demonstrate a range of possibilities using the device as an alternative technology to interactively control and manipulate parametric models (Salim et al., 2011).

Later experiments such as the work conducted by Kensek (2014), links physical computing systems and artifacts with Revit models. Her demonstrations do not include a TUI for designers to manipulate. The work focuses on simulating kinetic responses in the physical artifact coupled with the environmental analysis in the digital environment. The sensors are used to record ambient data to transform building components in both the digital and physical models. The link in Kensek's work utilizes *Firefly* (Payne & Johnson, 2012), software tools for Grasshopper and Dynamo, to establish data communication between 3D models and physical computing systems using Arduino

microcontrollers. More experiments using physical computing systems and Firefly with digital modelers are posted on Firefly's website, showing the wide range of possibilities and application for using interactive and responsive systems with parametric models.

2.7. Summary

The previous examples have shown some of the workflows utilizing TUIs for parametric modeling. These applications range from design collaboration, environmental analysis, to construction management. Furthermore, the examples also show some of the studies developed for assessing the technology in terms of intuitiveness of interaction, supporting design cognition, and computing education.

This research takes advantage of the intuitiveness of interaction with the TUIs, which is established in the literature, to develop a workflow for generating mathematical and algorithmic information to address the problem of defining parametric frameworks. The research finds that tangible interaction and methods for data interpretation can automate the process of establishing parametric models. To the author's knowledge, there is hardly any research on this topic of utilizing tangible interaction for automating the generation of mathematical and algorithmic information for establishing parametric models. This information is representative of physical design intents that is captured by the system and embedded in a parametric model.

3. RESEARCH METHOD

This work assumes a design-based research method that focuses on developing, testing, and evaluating haptic-based interactive prototypes. These systems utilize tangible interaction as an approach for addressing some of the challenges associated with parametric modeling, defining frameworks using mathematics and computer programming for embedding design intents. This research is conducted in three phases. Phase 1 involves 1) identifying the challenges of parametric modeling, 2) formulating the research questions and proposing a solution, and 3) providing a comprehensive overview of works and locating related examples of work. Phase 2 describes (1) the prototyping process and implementation, and 2) testing of the workflow. Phase 3 involves 1) internally evaluating and documenting the work, and 2) developing a framework for the method. Each prototype developed for this work reveals new findings in this experimental research approach, which contributes to the progress and the constant refinement of the method.

3.1. Design-Based Research

The act of design as a research approach as explained by Ma and Harmon (2009), is the process that:

> *Usually starts with a complex real-world problem. It involves iteratively generating a problem solution based on existing theories and practice, gathering empirical data to evaluate the solution, and*

reflecting on the design experience to refine the solution and to construct theoretical knowledge. It is usually a long-term research engagement requiring close collaboration among researchers and practitioners. (p.75)

The Design-based research development process as explained by Reeves (2000) and Ma and Harmon (2009) includes four main steps as illustrated in Figure 3.1 below.

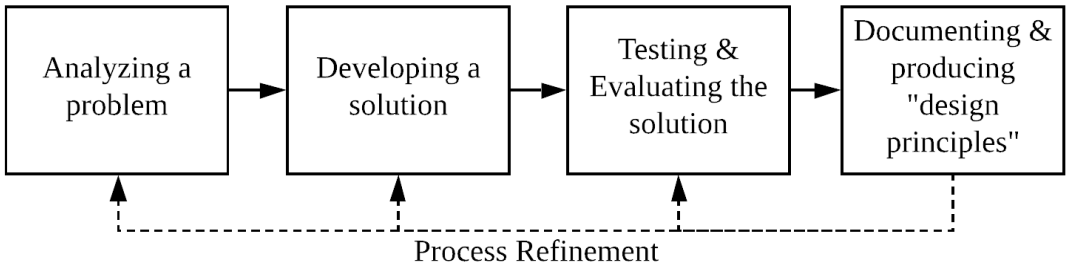

Figure 3.1 Research development process. Adapted from Ma and Harmon (2009), and Reeves's (2000).

3.2. Phase 1: Literature Review

This research intersects the disciplines of HCI and computational design. Examples of work in digital modeling that are influenced by these fields of study are examined in this research. Research articles on these topics are gathered from online repositories such as Cumincad (an open source Cumulative Index about research in Computer Aided Architectural Design), ACM digital library, ProQuest, WorldCat, and other scholarly platforms. The literature review process continues throughout the study to identify the most recent developments in tangible interaction for digital modeling.

While this research is primarily focused on parametric modeling, the work will also expand the investigations to include CAD tools and other virtual modeling platforms that are operated through tangible interaction. Research preparation involves acquiring skills in computer programming languages such as Python and Processing, computational modeling tools using algorithmic editors, and electronics.

3.2.1. Theoretical Background

This section focuses on providing a historical description of the development of parametric modeling and tangible interaction. Its core content provides an overview of fundamental concepts, terminology, definitions, and examples of significant works related to these topics in the context of this research.

3.2.2. Research Problem & Questions

Literature review assists in identifying some of the main challenges of parametric design. This research focuses on the problem of *defining* parametric frameworks using mathematics and computer programming for embedding design intents in digital models. Defining a parametric framework requires the designer to translate design knowledge in an explicit way into programming procedures.

The research questions, which are discussed in the Literature Review chapter, are intended to address the previously stated challenges of parametric modeling. This research includes three questions that investigate an approach for solving the issues of establishing parametric frameworks.

3.2.3. Theoretical Workflow

This research builds on top of earlier research that claims that tangible interaction can be considered as an intuitive method for interacting with digital models. A general workflow for the tactile-based system is illustrated in Figure 3.2. The workflow utilizes TUIs for generating the mathematical and algorithmic information necessary for embedding design intents in parametric models.

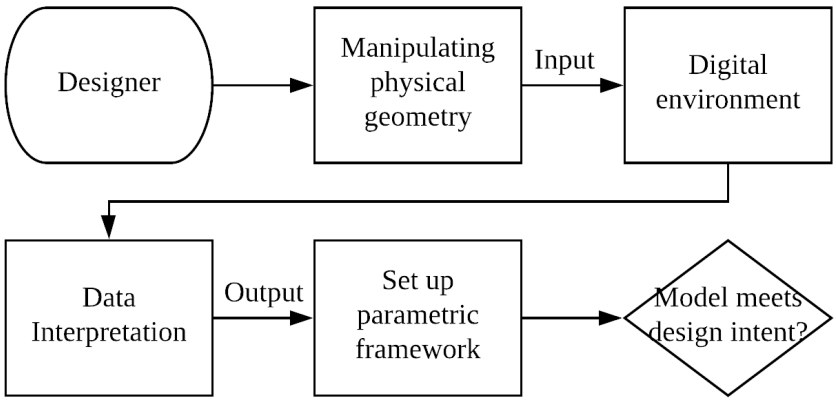

Figure 3.2 Prototyping theoretical workflow.

A list of criteria is developed for each segment of the workflow. The list is created by answering the following two questions:

- What is the type of physical interaction used for operating the prototype?

This question focuses on *interaction design* and the role of the designer in the digital modeling process. The objective of each prototype defines the specific type of interaction required to generate modeling information; e.g., if the designer wants to set

up a geometric relationship based on a rotation parameter, then physical design objects must be manually rotated to generate angles of rotations for the parameter inputs.

- What are the components of a TUI?

The TUI models developed for this work include two main components:

1. An *artifact*, which is composed of:

- *Design objects*: a physical representation of digital geometry or information and used for tactile manipulation.
- *Workbench*: a workspace that defines the physical boundaries of the TUI and holds the design objects.

2. A *physical computing* system: a circuit composed of a microcontroller, sensors, and actuators. The computing system is embedded in the artifact and monitors the changes in the objects' state.

The physical representation of a design object is essential in the design of a TUI; it assists the designer in making sense of the task at hand. Ishii (2008) mentions that there are *general purpose* and *special purpose* TUIs, and both have advantages and disadvantages in a digital-physical workflow. For the general purpose TUIs, they do lose the legibility of the physical representation because the objects in the artifact have an abstract form. However, they are suitable for a wide range of applications. For the special purpose TUIs, they can be limited in their application because of their specific design.

The representation of design objects in the artifact is determined in this research based on the modeling task and type of information, which the TUI will support in

generating. Design objects for this research are mainly geometric representations of architectural elements (e.g., louvers and panels) or digital information (e.g., control points for NURBS curves). Each type of design objects is used in an example design scenario for solving a parametric modeling problem.

3.3. Phase 2: Prototyping

The prototyping process includes three main steps: defining the objective of the prototype (problem), interactive procedure (input), and the type of parametric modeling information it will generate (output). A detailed process is provided in the diagram in Figure 3.3.

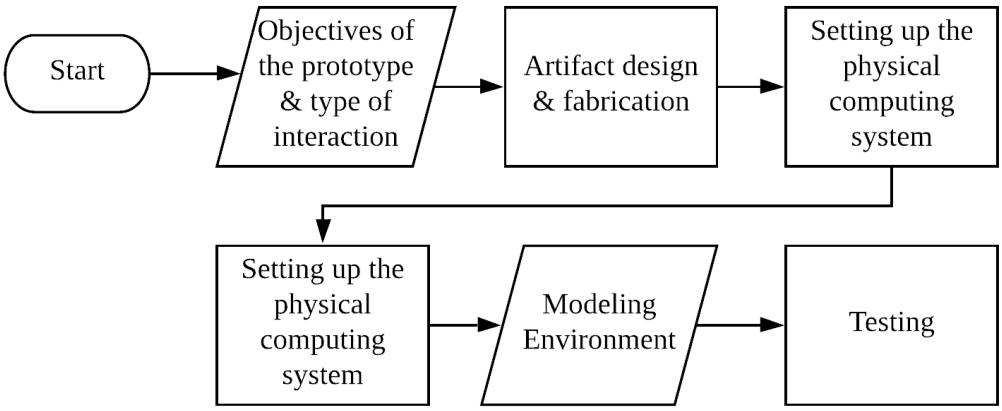

Figure 3.3 Graph showing the prototyping process.

3.3.1. Objective and Scope of Work

Each prototype investigates a specific problem related to the general inquiry of this research as described in the Literature Review chapter. The objective of each prototype determines the type of interaction for collecting analog data. In general, the prototypes are focused on direct manipulation by applying geometric transformations to physical objects.

3.3.2. TUI Specifications

A list of specifications is developed for each prototype, and is determined by the following points:

- Type of user interaction
- Transformation/behavior (physical object manipulation)
- Artifact components and data collection
- Type of data input
- Type of data output/digital model feedback

An example set of TUI specifications is shown in Figure 3.4.

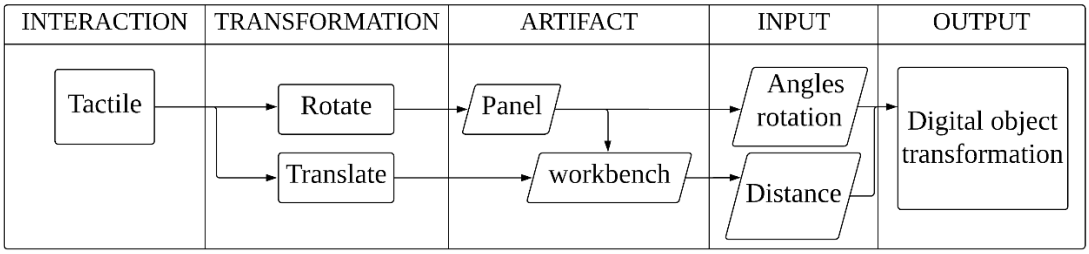

Figure 3.4 TUI specifications developed for Prototype 5.

The following description will focus on the data components of the specification list.

- Data collection

Data collection method refers to the type of sensors used in the physical computing system. Sensors are embedded in the artifact, and they are manually operated by manipulating the design objects. Sensor choice reflects the way the interaction will take place in the physical environment, e.g., a rotary potentiometer is used for objects that will be rotated, and a ribbon sensor for objects that will be moved.

- Data input

Sensors monitor and record physical interactions with their corresponding objects. Data gathered by these sensors reflect the physical state of the design objects, which is then sent to the digital model as raw values.

- Data output

Each prototype generates specific information, which includes a combination of geometric and non-geometric data. Non-geometric data in the context of this work refers

to equations, algorithmic rules, and numerical values that are generated after data interpretation for establishing the parametric model.

3.3.3. Control System Setup

The physical computing system is a circuit that is composed of a microcontroller, and a set of sensors and actuators. The purpose of this system is to translate a design object's physical state into the digital environment, and vice versa.

The criteria for choosing and setting up the physical computing system includes:

- Resources and support

Arduino and Raspberry Pi are two of the most common types of microcontrollers used for prototyping. The Microcontroller manages data exchange between the different components of the physical computing system. The type of microcontroller chosen for this research is Arduino, models MEGA 2560 and UNO. Both Arduino models are programmed using personal computers and the Arduino IDE software, which is written in Java and based on the Processing programming language. Arduino can be used with a verity of hardware components (sensors and actuators) to create a customized system for a wide range of applications. Information and documentation for building these circuits, including component specifications and schematics, IDE software updates and modules, training tutorials, and other technical support are available online for users through the official Arduino website and other online databases.

- Customization and reproduction

Arduino microcontrollers allow for creating specialized instruments for operating and interacting with digital models, because 1) they are reconfigurable opensource devices, with very minimal software and hardware restrictions compared to smartphones (or tablets), 3D scanners, and other digitization equipment; and 2) they can be easily linked to several computer application programs, which are accessible by the modeling platforms used for this research.

This research focuses on limiting data inputs to types of numerical values, angles of rotation and distance, using linear and rotary sensors. The objective is to maintain a consistent structure for the physical computing system assists in its rapid reproduction for other prototypes.

3.3.4. Design, Fabrication, & Assembly

Following is modeling the components of the artifact and preparing them for fabrication and assembly. This process uses a conventional CAD/CAM procedure. The artifact is designed using Rhino 3D. The completed components are stored in two separate file formats .3ds (for PLA 3D printing), or .dwg (for Laser cutting). The modeling process takes into consideration the following:

- Assembly details
- Design objects' mechanical movement
- Circuit integration
- Material properties

- Fabrication machinery specifications

These aspects of the artifact can be time-consuming to define for each prototype. Therefore, assembly details (such as joint configuration, type of nuts and bolts, and circuit casing), and materials (acrylic boards) were unchanged for the ease of reproduction. Furthermore, the artifact's parts (objects and workbench) were designed using visual programming. The algorithm included the assembly details, material dimensions, and fabrication machinery bed size. Having this information ready in the visual program allowed for the rapid reproduction of the different artifacts and the customization of individual parts as needed.

3.3.5. Interoperability

Several methods were developed and used for data communication between artifacts and digital models. Data flow for the prototypes can either be unidirectional or bidirectional. In addition, calibration and remapping are required: calibration, for removing errors in sensor reading and value fluctuation, and remapping, for maintaining consistency between sensor values and a model's measurements.

The methods for data exchange developed was primely achieved through the software package Firefly. Other methods that were tested for this work are discussed in the Prior Work chapter. Firefly offers a convenient approach for data communication and interaction with digital models compared to other methods of data transfer.

The modeling and visual programming environments used for this work are Rhino 3D (NURBS modeler) and Grasshopper (visual programming plug-in for Rhino),

and Revit (a BIM authoring tool) and Dynamo (visual programming add-on for Revit). Both sets of digital tools are commonly used for architectural design and are chosen for this work for their parametric modeling features.

3.3.6. Testing

Testing in this phase is for determining if the prototypes are fit for use, and it is not to be confused with *testing for validation* that is explained in Phase 3 (Prototype Implementation chapter). Testing helps in determining if a part of a system is working correctly (e.g., data flow, code, and operating procedures), and it is done separately for the artifact and the visual programming workflow in the digital model.

3.4. Phase 3: Prototype Implementation

Each prototype is explained in the Prototype Implementation chapter using the following outline:

- *Introduction*: describing the specific objective of the workflow and scope of work.
- *Prototyping*: Tools and TUI specifications used for developing the prototype.
- *Testing for validation*: demonstrating the workflow and documenting the results.

- *Results*: reflecting on the study to determine if the workflow meets the objectives of the research and proposing further developments to the workflow.

3.4.1. Testing Scenarios

Five prototypes are developed for this work and categorized into three groups: Algebraic Constraints, Algorithmic Rules, and TUI Improvements.

- Algebraic Constraints

This work focuses on setting up algebraic constraints for establishing parametric relationships in digital models. Two types of relationships are tested for this work which is based on linear and polynomial equations. This work involves using a regression analysis model in the visual programming algorithm for automatically deducing physical object relationships and representing them in mathematical equations. The objective of the work is to find the coefficients of the equations, which is challenging to calculate and implement manually for setting up algebraic constraints in digital models.

- Algorithmic Rules

The prototype is tested for setting up the initial cell state and rules for a CA algorithm to produce three-dimensional geometric patterns. CA rules define the dynamic relationship between cells on a lattice. Unlike the previous examples of constraints, which require equations for setting up a parametric framework, CA requires a set of rules for defining the cell states (alive, dead, or surviving). The objective of the work is

to provide a tool that promotes a higher level of understanding of abstract algorithmic concepts by associating physical configurations with programming logic.

- TUI Improvements

This section explores approaches for improving the workflow and functionality of the prototypes. Two prototypes are developed for this work and are described below:

The first involves developing an algorithm that detects the types of physical interactions to produce single and compound transformations using transformation matrices. The objective is to test a higher level of automation that allows for defining geometric operations in the digital workflow manually through the TUI. The artifact can be easily modified by adding and removing design objects to the workbench. This feature provides the flexibility needed to increase or decrease analog inputs

The second prototype is developed for modeling NURBS curves. This work involves developing an algorithm using visual programming to achieve three steps, modeling NURBS objects, set up its boundaries, and generate design options. The NURBS curve is constructed by proving the number and the location of its control points. The boundaries are established for the manipulation of the NURBS curve. The design options (curve interpolations) are generated using the boundaries.

3.5. Evaluation

The evaluation process includes two phases:

- Phase 1: correctness of the system

The observation will enable the researcher to evaluate the correctness of the system, by juxtaposing digital and physical results.

- Phase 2: qualitative comparison

The evaluation process involves providing a comparison between the developed workflow using TUIs and the conventional practice of establishing parametric models using text-based and graph-based programming methods.

3.6. Reflection

Reflection is stated in the Results section of the Prototype Implementation chapter, and it highlights the advantage and disadvantages of each prototype to inform the progress of the work. The objective of the work is to develop workflows that progress from performing simple to more complex modeling tasks with a higher level of automation for establishing parametric models, as shown in Table 3.1.

Table 3.1 Workflow progress.

Simple Modeling Tasks	Complex Modeling Tasks
- Basic object relationships - Programmed geometric transformations - Single geometric transformations	- Pattern generation - Automatic detection of transformations - Compound geometric transformations

4. PRIOR WORK

This section explains the process of developing the prototypes for this research. The work presented in this chapter was published in the Proceedings of the *34th eCAADe conference*, "Developing a Tangible User Interface for Parametric and BIM Applications Using Physical Computing Systems" by Al-Qattan, Galanter, and Yan, 2016. This section includes updated figures in addition to the published material.

A series of tangible and interactive systems were created and tested for setting up the TUIs. The two main phases of this process include:

1. Prototyping
    - Prototype components
    - Hardware and software tools
2. Types of workflows
    - The direction of data transfer and linkage methods
    - Interaction design and TUI specifications
    - Overall system framework

4.1. Prototyping

The physical computing system is designed and seamlessly integrated within the TUI's artifact. The *artifact* is composed of physical design representations of either architectural design objects or digital information and a workbench, which is the physical workspace of the TUI and casing for the physical computing system. A link is

developed for enabling direct manipulation of the digital model through the TUI. Several linkage methods have been tested for this work and are demonstrated in the examples of this chapter. The integration between the computing system and artifact allows for 1) real-time data processing during user interaction with the TUI and 2) providing the user with familiar objects to naturally interact with for manipulating digital models. Figure 4.1 shows the different parts of a TUI in a fully developed prototype for parametric modeling.

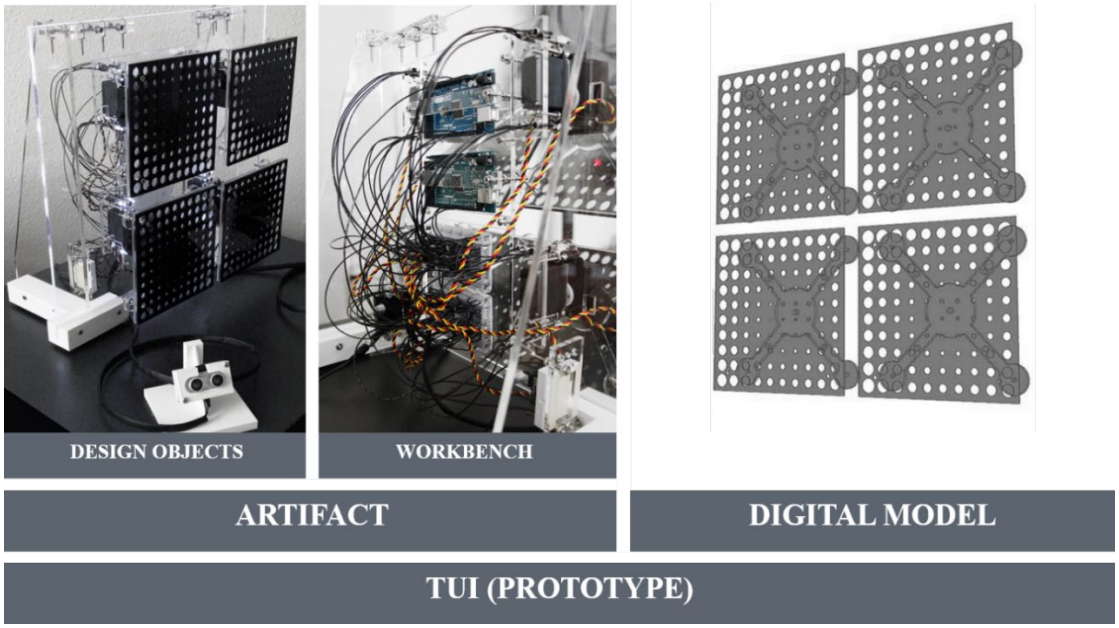

Figure 4.1 Prototype main parts; left and middle images show the TUI, which is composed of an artifact (panels), and workbench (supporting frame for the panels and physical computing system). The right image shows the digital model of the panels created in Rhino. Figure Adapted from Al-Qattan et al. (2016).

4.2. Workflows

Sensors embedded in the design objects monitor physical object changes and pass on this information to the microcontroller. Each linkage method uses a different set of computer applications for processing analog data as will be shown in the following tests.

Three tests were conducted showing the different approaches using a combination of different sensors and actuators, and data flow methods for setting up TUIs. These examples are organized based on the direction of data flow, and they include:

- Unidirectional data flow: Physical to digital

A workflow for collecting and processing analog data provided by the designer to transform digital geometry. The set of tools for this example includes Revit, Dynamo, Arduino, a proximity sensor, a rotary potentiometer, Excel, and PLX-DAQ (Parallax Inc., n.d.).

- Unidirectional data flow: Digital to physical

A workflow for exporting digital data from the model to actuate physical design objects. The set of tools used for this system includes Revit, Dynamo, Arduino, two servomotors, Excel, and Processing.

- Bi-directional data flow

A workflow that enables data exchanges to occur between both the digital and physical environments. This workflow enables the designer to take advantage of both digital tools and haptic skills in the modeling process (Figure 4.2). The set of tools for

this example includes Rhino, Grasshopper, Firefly, Arduino, two proximity sensors, and eight servomotors.

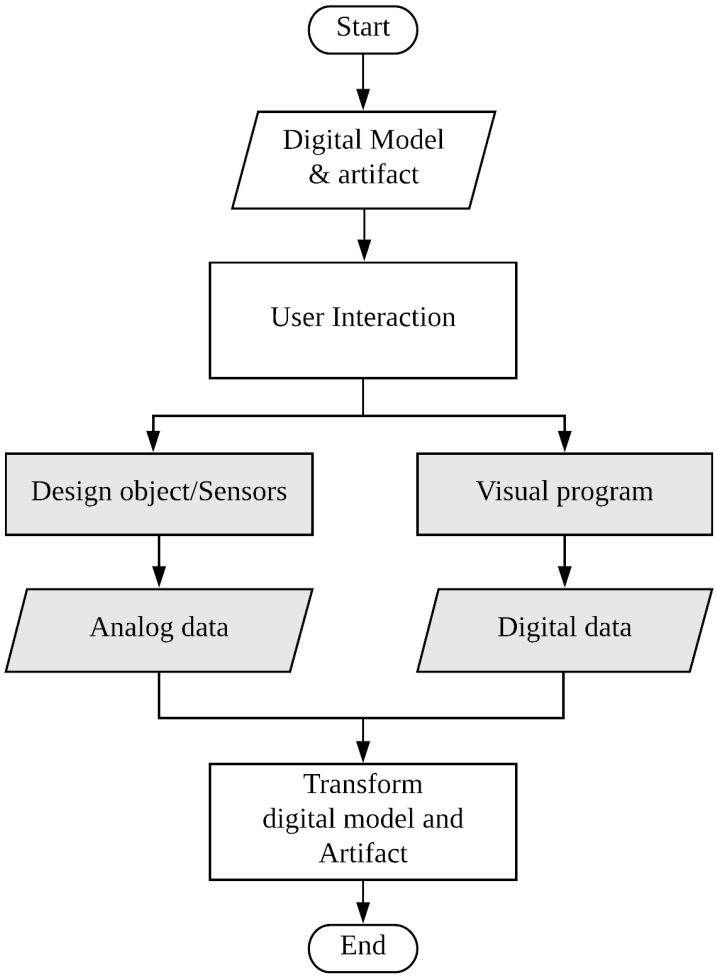

Figure 4.2 Workflow is showing a bi-directional link between the TUI and the digital model.

4.2.1. Unidirectional Data Flow: Physical to Digital

The TUI for this test explores a simple unidirectional data flow for providing parameter inputs for a Revit model. The physical computing system provides two types

of inputs, which are angles of rotation by using a rotary potentiometer and distance measurements using a proximity sensor. This set up does not include design objects or a workbench. It only includes the sensors for the designer to operate as shown in Figure 4.3. This set up is simplified to test the operability of the system.

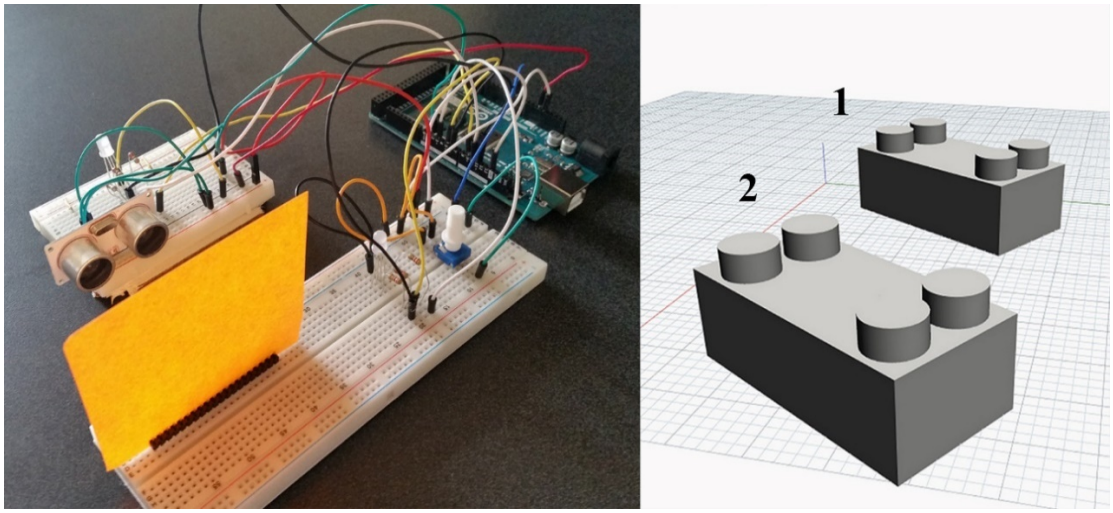

Figure 4.3 Testing a physical computing system for transforming design objects in Revit. The left image shows the circuit having a proximity sensor, which is linked to Mass 1 in the Revit model and a rotary potentiometer linked to Mass 2. Sensor data will provide numerical inputs to transform the objects in the digital model (Al-Qattan et al., 2016).

The specifications for the TUI are illustrated in Figure 4.4. The artifact does not include a design object or workbench as the purpose of this test is to explore the set up for the physical computing for collecting, processing, and transferring analog data to the virtual environment. The designer interacts with the device by rotating the potentiometer's handle or by moving the proximity sensor closer or away from the piece of paper attached to the breadboard. The designer is required to make these actions to

operate the control system and provide the inputs for transforming the objects in the Revit model.

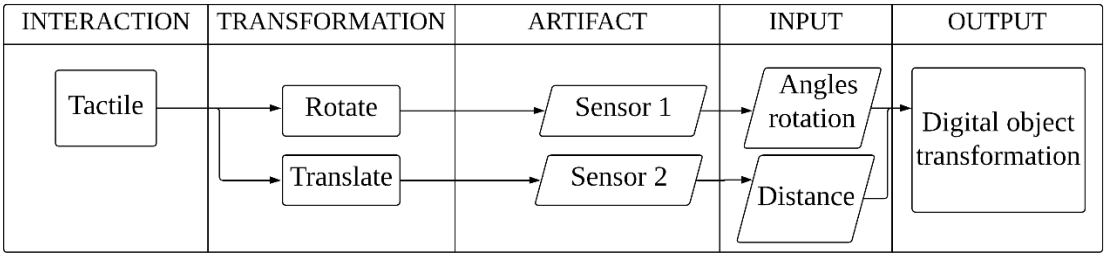

Figure 4.4 TUI specification for testing a digital to physical workflow.

The Revit model consists of two geometric masses that resemble Lego blocks and includes two transforming parameters: rotation and translation. Each sensor is connected to its corresponding parameter in Revit: rotary potentiometer with rotation and proximity sensor with translation. Each transforming parameter is assigned to one of the Revit blocks using Dynamo. The aim of associating parameters to sensors in this manner is to maintain consistency between the designer's interaction with the device and geometric behavior, i.e., if the designer rotates the sensor's handle, it provides Dynamo with angles of rotation for transforming. The workflow for this test is shown in Figure 4.5.

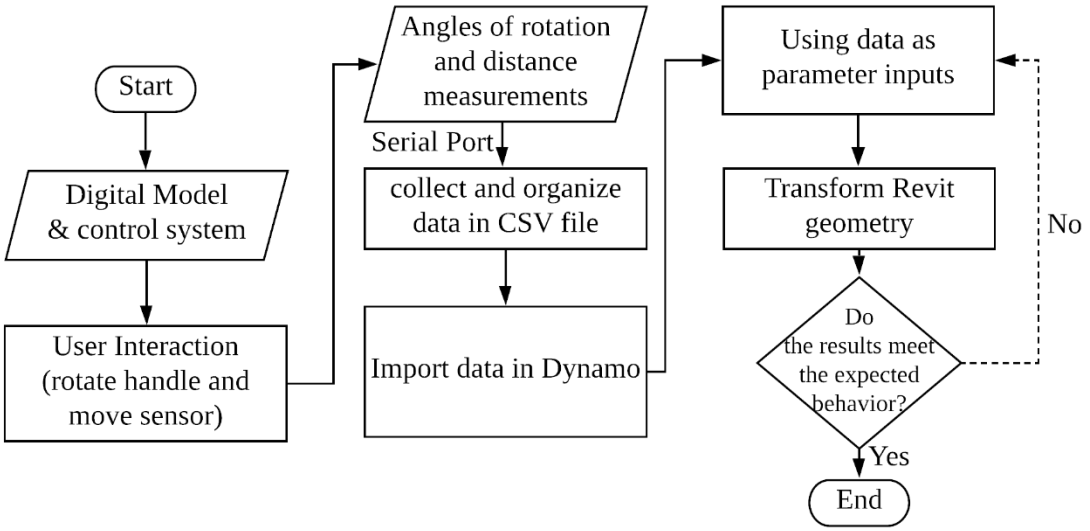

Figure 4.5 The digital to physical workflow setup adapted from Al-Qattan et al. (2016).

1. Testing

The physical computing system for this test is linked to Revit and Dynamo using Excel and PLX-DAQ. Sensors send data continually to Arduino during the user's operation of the device. This data is stored in a Comma-Separated Values (CSV) file for compatibility with Excel. Values are arranged in two separate columns in Excel, one for angles of rotation and the other for distance values. The latest values received from the sensors are inserted in a new row in their corresponding columns in the spreadsheet. As seen in Figure 4.6, there are two columns of numerical values in both the Arduino serial port and the Excel spreadsheet. The left column is for distance in inches, and it is received from the proximity sensor, and the right column is for the angles of rotation in degrees, and it is received from the rotary potentiometer. The Excel file afterward is imported in Dynamo where the values form each sensor are extracted and sent to their

assigned parameters. The link between Excel and Dynamo enables data exchange in almost real-time by using a MACRO in Excel, which automates the file Save function.

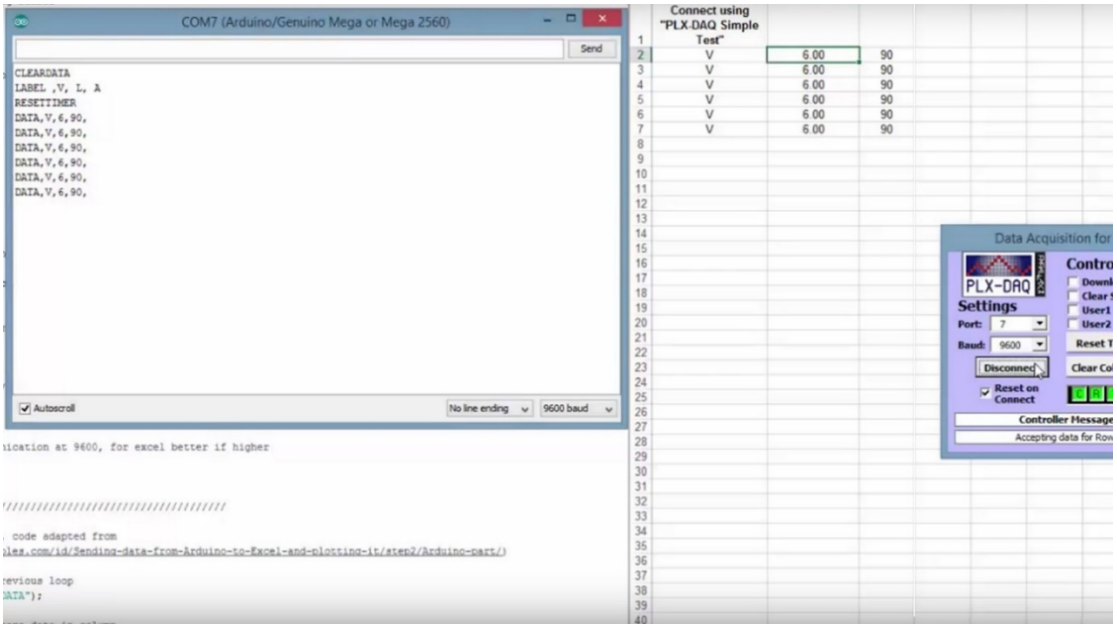

Figure 4.6 A screenshot showing the Arduino serial port (left), and Excel spreadsheet and the dialog box for the PLX-DAQ plug-in (right) (Al-Qattan et al., 2016).

Figure 4.7 (left image) shows the masses in Revit. The mass on the left (Mass 1) is located at the origin of the coordinate system and set up with a rotation parameter. The rotary potentiometer provides the rotation angles for this mass to rotate it around the Z-axis. The mass on the right (Mass 2) is set up with a distance parameter, which controls the distance between it and Mass 1. The distance parameter establishes a simple type of parametric relationship between the two masses, which can be manipulated by operating the proximity sensor. Distance values translate Mass 2 closer to or away from the Mass 1. The proximity sensor measures the distance between it and the piece of paper attached

to the breadboard. If the sensor is moved closer or away from the piece of paper, the Revit model responds in a similar way. Figure 4.7 shows the transformation of both masses in Revit using the two sensors. The left image shows the Starting Position for both masses, and the middle and right images show the process of transforming both masses gradually to their target position (image on the right).

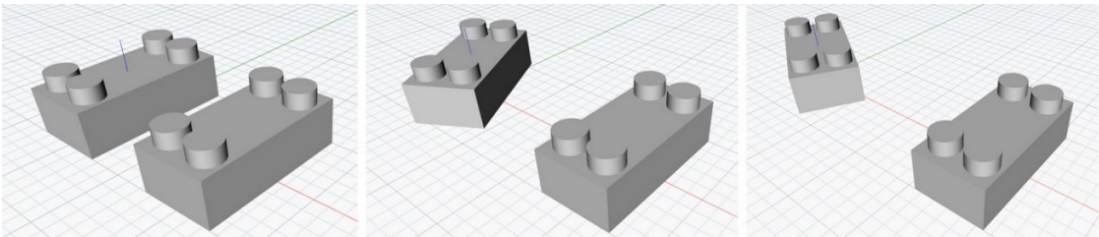

Figure 4.7 The Revit masses transformed using the control system (Al-Qattan et al., 2016).

2. Results

This test shows a simple workflow for setting up a physical computing system for creating TUIs. The unidirectional data flow provides designers with an interactive device using sensors to manipulate digital models. The test also shows consistent results between the behavior of both masses in the digital model and the designer's interaction with the sensors. However, the link established between the digital model and control system using several computer program applications made it difficult to monitor data flow and navigate the computer programs during operation.

4.2.2. Unidirectional Data Flow: Digital to Physical

This test explores a workflow for sending digital data to TUIs. The physical computing system includes two servo motors to display rotation transformations. A piece of paper is attached to each of the servo motors' arms as shown in Figure 4.8 (image on the left). The pieces of paper assist in monitoring the servo motors' performance (rotation) when they receive data from the digital model. The digital model from the previous workflow is reused for this example except for the translation parameter (image on the right). The physical computing system and digital model are linked together using the same set of tools as in the previous workflow, with the exception of PLX-DAQ being replaced with Processing.

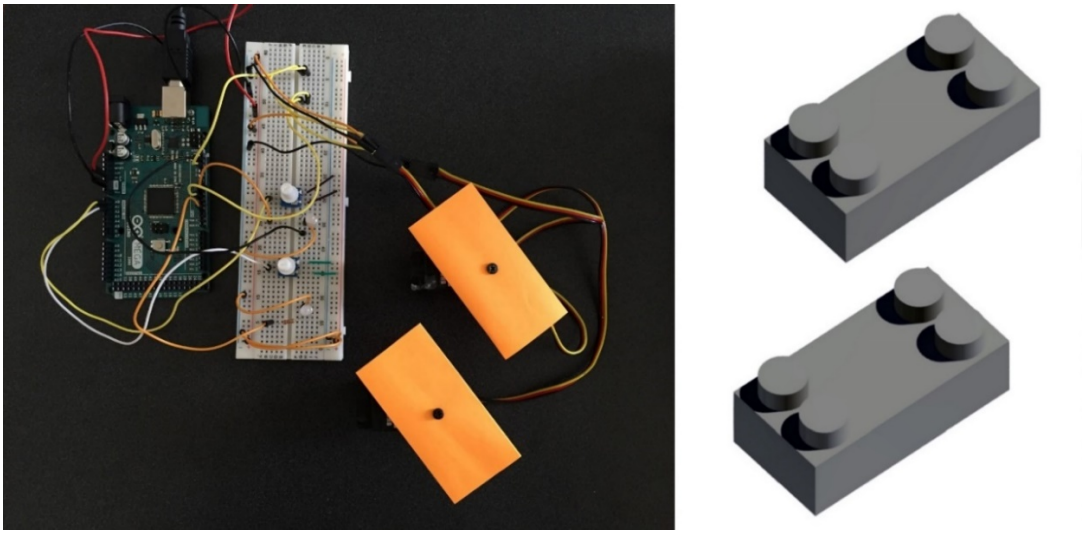

Figure 4.8 The physical computing system (left image) having two servomotors with a piece of paper attached to each of them. Each mass in the Revit model (right image) is connected to one of the servo motors in the physical computing system (Al-Qattan et al., 2016).

The specifications for designing the user's interaction with the system are shown in Figure 4.9. The designer provides angles of rotation in Revit for each of the two masses using Dynamo. The angles of rotation are then passed on to the Arduino microcontroller then to the servo motors. The user interacts with the system using the keyboard and mouse.

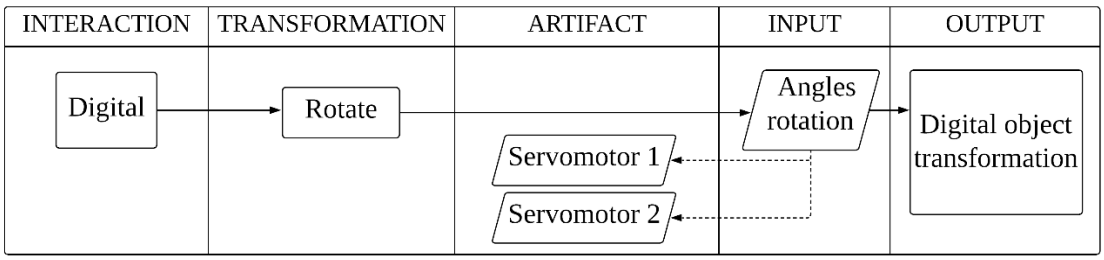

Figure 4.9 Specification for developing the physical computing system and user interaction with the system. The dashed lines indicate that there is an indirect relationship between user input and servo motor's motion, as the angles of rotation values are not set directly by the user manually rotating the servo motors' arms.

The two rotation parameters are linked to the servo motors in the physical computing system and are provided using a number slider in Dynamo. The angles of rotation are transferred to the servo motors by having them recorded and stored in a CSV file using the Excel tools in Dynamo. The data file is accessed and sent to the microcontroller using Processing. Figure 4.10 shows the workflow for this test.

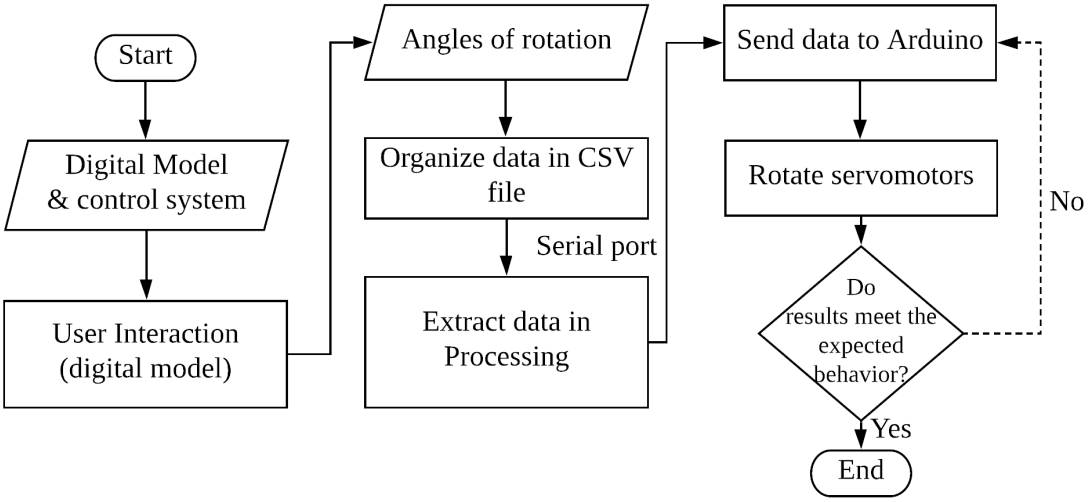

Figure 4.10 The workflow for the digital to physical data flow test adapted from Al-Qattan et al. (2016).

1. Testing

Figure 4.8 previously demonstrates an example of how both physical and digital objects respond accordingly showing consistent results when their parameter values are changed. This test is a simple demonstration of data flow from the digital model to the physical computing system.

2. Results

This workflow, using Processing and Dynamo has helped in reducing the number of computer applications that were running at the same time, which made navigating the system much more efficient. It is important to note that actuators can be limited in their response, unlike digital models which are more flexible in constructing and manipulating. Thus, the physical behavior of physical objects must be considered early in the prototyping phase of the TUI to ensure that both digital and physical geometry demonstrate consistent results.

4.2.3. Bi-directional Data Flow

The previous tests helped in understanding the essentials of building interactive systems for digital modeling. This example expands on previous tests to include a fully functional prototype using bi-directional data flow. The artifact includes a physical computing system that is composed of two Arduino microcontrollers, eight servomotors, and two proximity sensors. The artifact also includes design objects and workbench, a full-scale section of a cladding system with its panels capable of three-dimensional rotation (Figure 4.11).

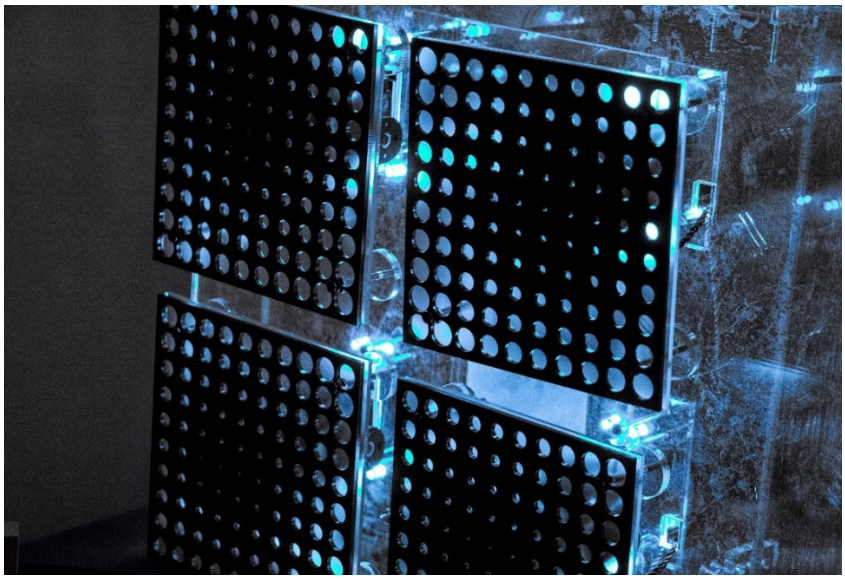

Figure 4.11 The artifact showing the four panels in front and the workbench in the back (Al-Qattan et al., 2016).

1. Prototyping

The designer interacts with the TUI through the proximity sensor or through the digital environment as an alternative. Each panel includes two axes of rotation, and each

axis is controlled by one of the two proximity sensors. The proximity sensors are operated by measuring the designer's distance from them. Distance values are converted to angles of rotation in the digital environment using visual programming. Figure 4.12 shows the specifications for this prototype example.

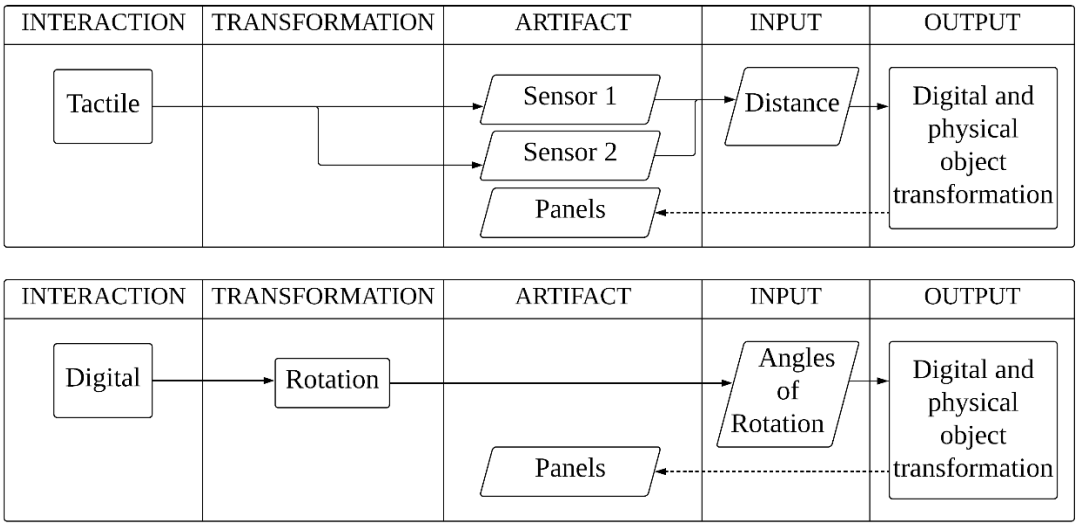

Figure 4.12 Specifications for the prototype. The top image shows the data flow from the physical to the digital environment, and the bottom image shows the data flow from the digital to the physical. The top image also shows that no transformation of the design object is included because there is no direct tactile manipulation of the design object.

An architectural element (design object) is included as part of the TUI to provide a familiar object. Custom elements were designed and fabricated to create the objects, workbench, and assembly details. The same tools, assembly techniques, fabrication machinery, and materials used for this example were reused for developing Prototypes 1 to 5 included in the Prototype Implementation chapter. The two main fabrication

methods used are additive manufacturing using Fused Disposition Modeling (FDM 3D printing) and subtractive manufacturing using laser cutting.

The panels in the artifact are capable of rotating in two axes, which is a type of motion referred to as *Pan* and *Tilt*. Each type of motion is on a single axis, which enables the panels to demonstrate complex behaviors in the physical space. This is achieved by attaching two servo motors using aluminum brackets (provided by Lynxmotion, n.d.) to each panel. Each sensor is used to control one of the two motions for all the four panels. Figure 4.13 shows the sensor and bracket assembly.

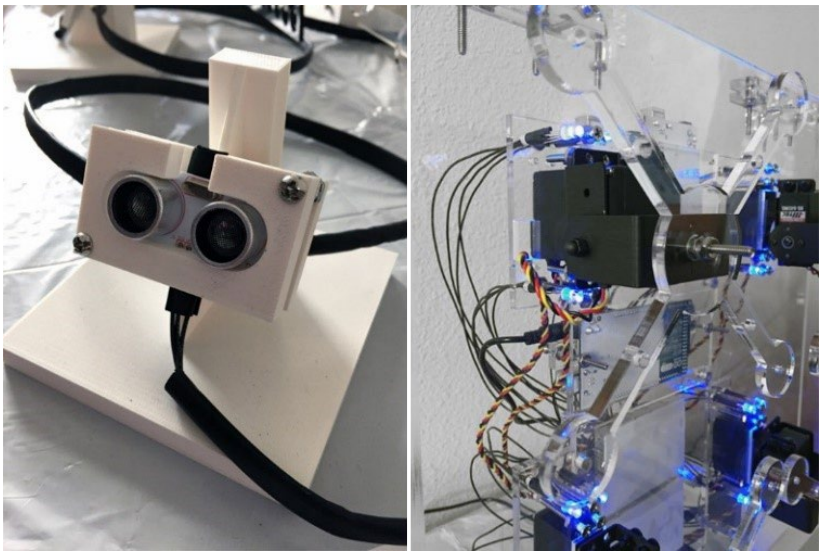

Figure 4.13 The image on the left shows the proximity sensor and its casing, and the image on the right shows the servo motor set up using the aluminum brackets for a single panel. Figure Adapted from Al-Qattan et al. (2016).

The artifact is linked to a digital model in Rhino using Grasshopper and Firefly. Rhino and Grasshopper are used in this example as an alternative set of parametric modeling tools to Revit and Dynamo. The primary consideration of choosing a modeling

tool for this work is its parametric capabilities, and software support. Firefly for Grasshopper includes a broader set of tools to establish the link between the TUI and digital model.

Each sensor provides the angles of rotation to its corresponding parameter in the digital model and servo motors in the artifact. These values are sent to Grasshopper using Firefly as distance values, which are then converted into angles of rotation. These values are treated in a similar manner in both Grasshopper and the artifact, values are sent to their corresponding plane for rotating the panels (Figure 4.14).

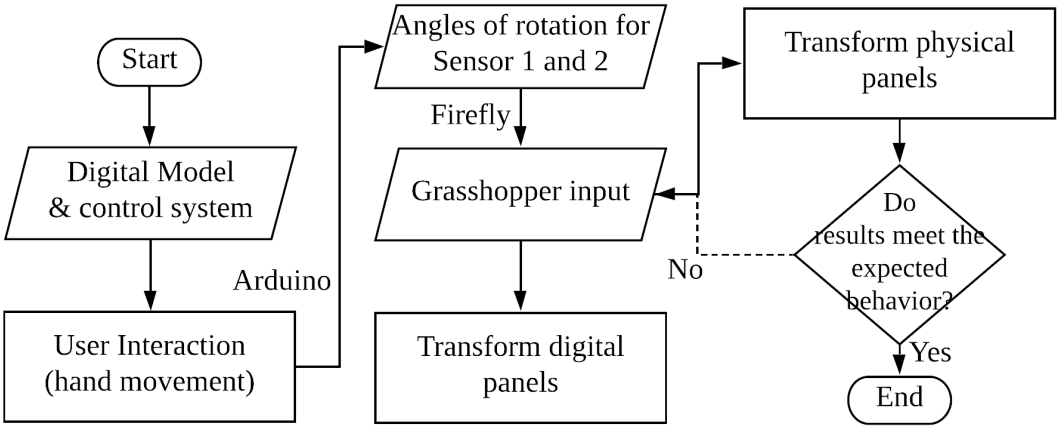

Figure 4.14 The workflow for developing the prototype adapted from Al-Qattan et al. (2016).

2. Testing

The prototype is operated through hand movement, i.e., users move their hands closer or away from the proximity sensors to rotate the panels. The distance input is converted into angles of rotation in Grasshopper. Sensors detect users' hands up to 12 inches away from the artifact. Sensor values are remapped in Grasshopper from 0 to 12

inches to -90 to 90 degrees for rotating the servomotors. The angles of rotation are then sent to their corresponding nodes in the Grasshopper algorithm to transform the Rhino model and to the artifact to display the geometric results (Figure 4.15).

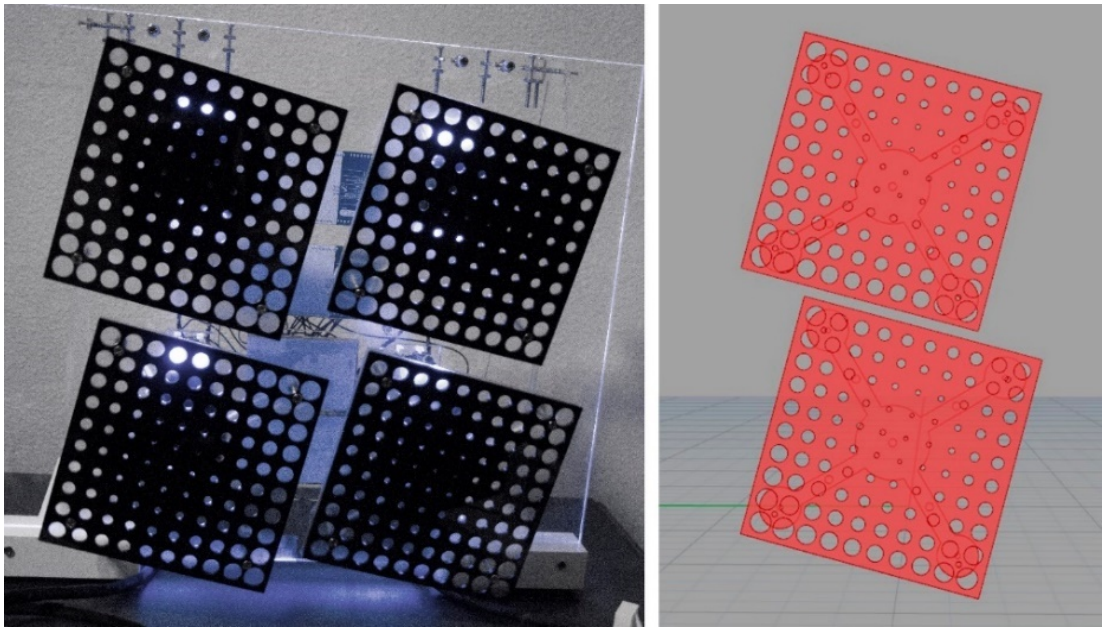

Figure 4.15 Showing the results of testing the prototype. Two panels are shown in the Rhino viewport for monitoring the model's behavior (Al-Qattan et al., 2016).

Figure 4.16 shows the artifact as a standalone interactive geometric model (without the digital model) on display in downtown Bryan, Texas. It was presented as part of the 22nd Viz-a-GoGo exhibition in 2015, an annual showcase of students' work at the Department of Visualization (College of Architecture at Texas A&M University). Users interact with the artifact during this event was not under a controlled setting or part of a formal user-study experiment. It was a simple case of observing people of all ages interacting freely with the device for transforming the panels. Noticeable was users'

hand-eye coordination, a connection established naturally without having any prior training on how to use the device.

Figure 4.16 Artifact on display at the 22nd Viz-a-GoG0 exhibition in Downtown Bryan, Texas. Figure adapted from Al-Qattan et al. (2016).

3. Results

This work tested a bi-directional data flow using a TUI with a full-scale design object. There was no direct physical interaction with design objects in the artifact. The drawback of this set up is that it is difficult to determine the precise degree of rotation based on simple hand movement. Using proximity sensors which measure distance requires practice to establish the connection between the distance from the sensor and the panels' angle of rotation. Conversely, the example of the Physical to Digital workflow in the first test of this section, the potentiometer's handle provided a good indicator for objects rotation when operated by the designer.

Furthermore, the TUI includes two microcontrollers, yet Firefly only communicates with one Arduino at a time. Each Arduino controllers the panels' rotation in one plane (one microcontroller for tilting motion and the other for panning motion)

thus making it difficult to rotate the panels in three-dimensional space when connected to a digital model. However, if the artifact is used as a standalone device, it worked adequately. The artifact is set up to send and receive data from the digital environment. However, data transaction does not co-occur, and the system is either used to send data from the physical to the digital or vice versa. Most importantly, the artifact provided an instrument for working with digital models by merely providing parameter input values, which can be easily achieved by manually inserting them in the digital model using a number slider in the visual program. These issues are considered when developing later prototypes for this research. This chapter explores the development of TUIs and linkage methods between artifacts and digital models, which will be used in the Prototype Implementation chapter.

5. PROTOTYPE IMPLEMENTATION

This chapter discusses the workflows and prototypes developed using tangible interaction for parametric modeling. The workflows demonstrated in these examples are for automatically capturing design intents from physical objects and translating them into parametric frameworks. The term design intent encompasses a broad range of meanings and activities associated with the design process, which can be very complicated to demonstrate entirely through the prototypes presented in this chapter. For this work, the design intent is represented by object relationships and algorithmic rules. The workflows in this chapter focus on the process of constructing parametric frameworks using mathematics and programming procedures for representing design intents digitally. Further prototype improvements demonstrate sophisticated workflows for automating modeling procedures for detecting physical transformations and creating parametric NURBS objects. The prototypes are organized into three categories: Algebraic Constraints, Algorithmic Rules, and TUI Improvements; and each is focused on demonstrating a process for translating tangible interaction with design objects into modeling information as shown in Table 5.1. Algebraic Constraints includes two prototypes, and each is designed to demonstrate a workflow for setting up object relationships in digital models using mathematical equations. Algorithmic Rules includes one prototype tested for two workflows for setting up a CA algorithm for generating geometric patterns in a digital model. TUI Improvements include two

prototypes that demonstrate a higher level of automation for generating geometric procedures and creating NURBS objects.

Table 5.1 Each category explains a workflow for generating specific modeling data to address a parametric modeling problem.

Modeling Input	TUI	Scope of Work
Algebraic Constraints	Prototype 1	Generating line equations
	Prototype 2	Generating polynomial equations
Algorithmic Rules	Prototype 3	Setting up CA rules and initial cell states
TUI Improvements	Prototype 4	Automatic detection of geometric transformations
	Prototype 5	Generating NURBS curves and setting up modeling boundaries

Each prototype included in these categories is composed of two main parts, the artifact and the digital model, which establish the TUI. The artifact, as seen in the previous chapter, is composed of design objects, a workbench, and a physical computing system. Design objects are a physical representation of architectural elements or digital information; and a workbench, which defines the workspace and limits of the artifact and acts as the casing for the physical computing system. The physical computing system is composed of a microcontroller and a set of electrical and electro-mechanical components. The circuit is integrated with the artifact for close monitoring of design objects during user interaction and to create a seamless interface for completing the

different modeling tasks. The artifact in each example is linked to a virtual environment for live data streaming, enabling real-time interaction with digital models when design objects are manipulated.

5.1. Algebraic Constraints

The examples of work in this section address the problem of utilizing mathematical equations for generating object relationships and complex forms in digital models. This work was published in the Proceedings of the *22nd CAADRIA conference,* "Establishing Parametric Relationships for Design Objects Through Tangible Interaction" by Al-Qattan, Yan, and Galanter, 2017a. This section includes updated figures in addition to the published material.

Prototypes 1 and 2 demonstrate a method for automatically generating mathematical equations for digital modeling by analyzing physical object states. The work focuses on generating linear and high-degree polynomial equations for creating parametric object relationships. The results obtained from testing the systems have shown the plausibility for utilizing tangible interaction as an approach for constructing mathematical equations and embedding them in digital models instead of the conventional approach using text-based and graph-based programming applications.

5.1.1. Linear Equations

Prototype 1 tests a workflow for generating linear equations for establishing object relationships in digital models. The TUI is composed of an artifact consisting of

three panels (Figure 5.1, left image), which are assumed to be as a small section or part of a larger tessellated surface. The physical computing system includes sensors that are embedded in the artifact to monitor the changes in the panels' physical state during the process of user interaction and actuators for displaying physical responses. It is important to note that the panels in the artifact are for representational purposes and for providing a familiar object for working with abstract mathematical and algorithmic information. Therefore any configuration of design objects can be used instead of the panels if a similar type of linear relationship is to be established in the digital model. The digital model in Revit includes a duplicate version of the panels, which are linked to their corresponding physical counterparts in the artifact (Figure 5.1, right image).

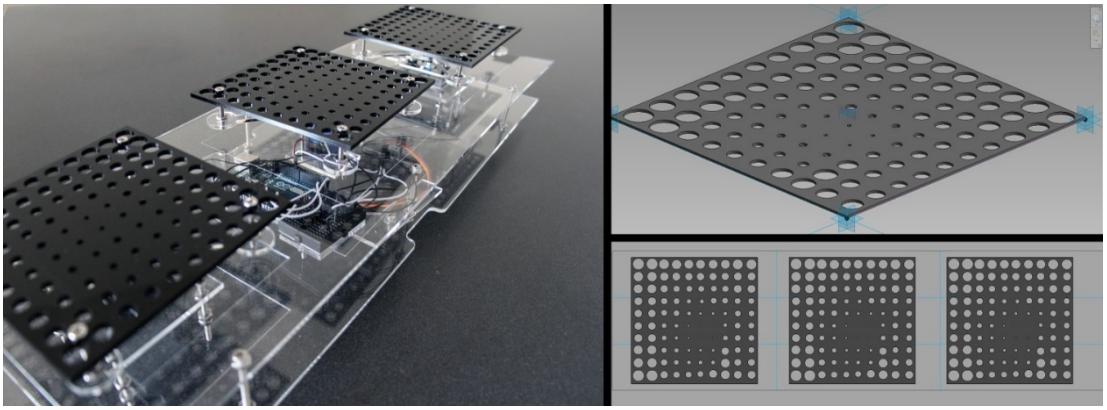

Figure 5.1 Prototype 1 consists of a TUI composed of an artifact having panels and a workbench and a physical computing system having sensors and actuators (left image), and a BIM model created in Revit (right image) (Al-Qattan et al., 2017a).

The prototype is tested for establishing a relationship between the panels through physically transforming the design objects, e.g., rotating the panels. The algorithm created in Dynamo will detect how the objects relate to each other when the direction

and degree of rotation are changed in the physical panels. The system monitors these changes and uses this information to generates an equation depicting the panels' relationship to replicate the physical results in the digital model. The procedures written in the algorithm include analog data logging and analysis which are used for generating the equations. Raw data samples are collected when the designer transforms the panels. A regression model is implemented in the algorithm to analyze data samples, by using a curve-fitting function. The equation generated through these procedures is then used to setup constraints in the digital model.

1. Prototyping

Prototyping includes two phases, determining the list of tools and specifications for developing the prototypes.

Phase 1. The set of tools for developing Prototype 1 includes:

- Software tools: Revit; Dynamo, a visual programming Add-on for Revit; IronPython, a programming language; Firefly; and Microsoft Excel.
- Hardware tools: an Arduino UNO microcontroller, a servomotor, and two rotary potentiometers.

Phase 2. Shaer et al. (2004) have developed an outline for specifying TUIs. This work will adopt a similar approach for developing the list of specifications for the TUIs. The list describes the process of designing the TUI for user interaction; it provides a conceptual framework of the system's operation. The list of specifications includes the type of user interaction, type of transformations, artifact composition, data input, and data output. Each list of specifications is unique to its TUI for achieving the prototype's

objective. For example, Prototype 1 is developed to generate a linear type of object relationship such as having panels rotating accordingly to a specific angle. The designer transforms the panels directly by rotating them during which the sensors will start collecting data samples. These samples are analyzed by the system for generating the equations for setting up modeling constraints in the BIM model. Additionally, Prototype 1 includes a bidirectional data flow; raw data samples are transferred from the artifact to the virtual environment and vice versa, thus enabling the prototype to display digitally and physically the design objects' relationship when modeling parameter values are changed. Figure 5.2 shows the list of specifications for Prototype 1.

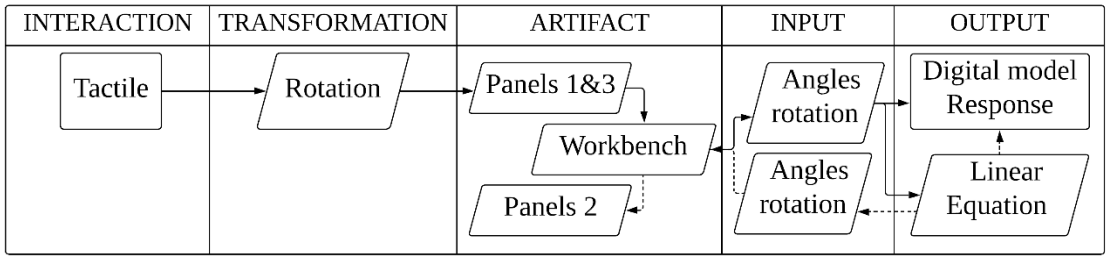

Figure 5.2 TUI specifications for Prototype 1. The angels of rotation for Panel 2 are calculated using the generated equation and sent to both the artifact and digital model. This process is shown using dotted lines.

2. Testing

The prototype is operated by manually rotating Panel 1 and 3 horizontally around their centers (panels at the opposite ends of the workbench). The rotary potentiometers attached to each of the two panels will monitor the changes in the angles of rotation when the two panels are transformed. Analog values are collected by the sensors and

sent to the visual program created in Dynamo as data samples for regression analysis. The curve-fitting function programmed in IronPython and integrated within the algorithm in Dynamo detects patterns in the data sample collected by the sensors. The outcome of this process is a mathematical equation depicting the relationship found in the data samples. The equation then is used for setting up constraints in the digital model and used to rotate Panel 2 in between the other two panels as illustrated in Figure 5.3.

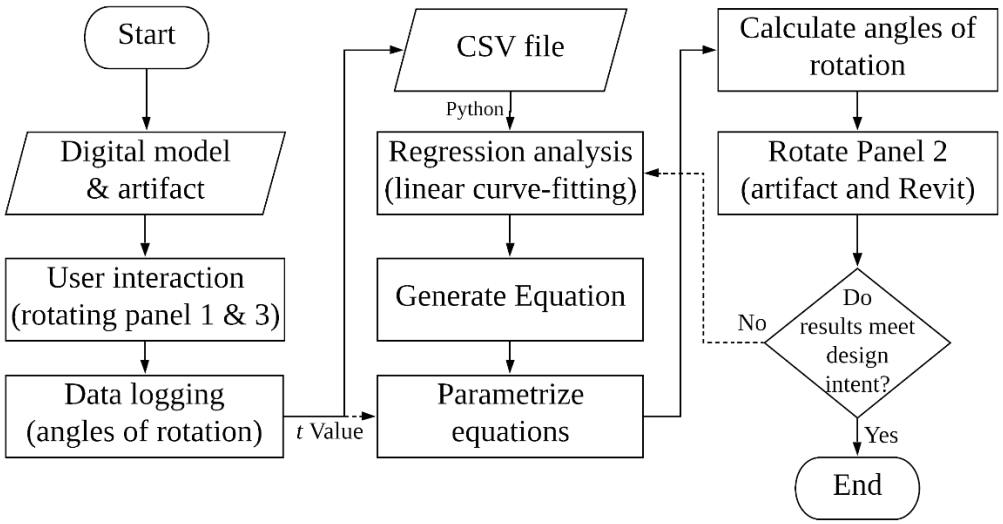

Figure 5.3 The workflow for Prototype 1 adapted from Al-Qattan et al. (2017a).

The process of collecting data samples is shown in Figure 5.4, where Panel 1 and 3 are respectively rotated from 0 to 180 degrees, i.e., Panel 1 rotated by {X1, X2, X3} degrees and Panel 3 rotated by {Y1, Y2, Y3} degrees. The values obtained from each sensor is recorded and stored in a CSV file using the Excel tools in Dynamo.

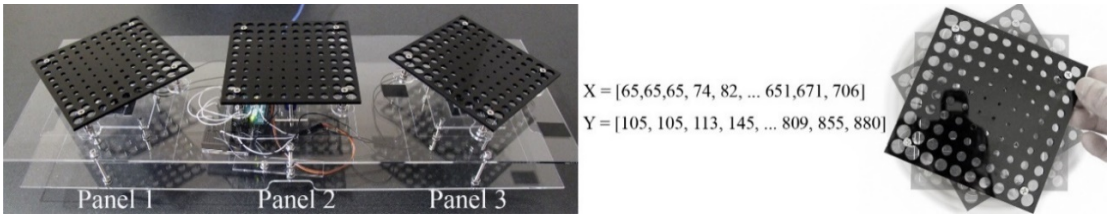

Figure 5.4 The left image shows Panels 1 and 3 rotated respectively in opposite directions. The middle image shows data samples collected and stored in a CSV file; the X data list includes angle values from the sensor attached to Panel 1 and Y data list is from Panel 3. The right image shows the direct user transformation of the panel (Al-Qattan et al., 2017a).

The CSV file including the data samples is imported into Dynamo for curve-fitting and generating the equation. A linear equation in mathematics consists of a constant and the first power variable, which can be represented mathematically as $y = mx + b$, where m is the Slope and b is the y-Intercept. This format of the equation is referred to as Point-Slope form with a single variable.

The constants of the generated equation are calculated as shown in Eq.5.1 (Slope) and Eq.5.2 (y-Intercept) (Yan & Su, 2009).

$$m = \frac{\sum[(x_i - \bar{x})(y_i - \bar{y})]}{\sum[(x_i - \bar{x})^2]} \qquad \text{Eq.5.1}$$

$$b = \bar{y} - (m\bar{x}) \qquad \text{Eq.5.2}$$

The generated equation at this point deduces a relationship between Panel 1 and 3 in the array excluding Panel 2. The angle of rotation value for rotating Panel 2 is obtained by solving x and y in the Point-Slope equation.

Parametric functionality is not supported by the generated equation. i.e., the calculated angle of rotation will provide a single instance for rotating Panel 2 in between the other two panels. Therefore, an additional procedure is implemented in the Dynamo algorithm to parametrize the equation, which involves setting up a free parameter (t) as an independent variable for solving x and y ($x = t, y = mt + b$). The values of t is substituted with the angles of rotation provided by the rotary potentiometer attached to Panel 1 in the artifact. Using this methods of parametrization allows for calculating all the possible angles of rotation for Panel 2 (i.e., interpolations) in the array between the other two panels.

Once a constraint is set up using the parametrized equation, any further changes in the angles of rotation for Panel 1 and 3 will automatically reflect in both the digital model and artifact by updating the parametric equation's coefficients and recalculating the angles of rotation for Panel 2 (Figure 5.5).

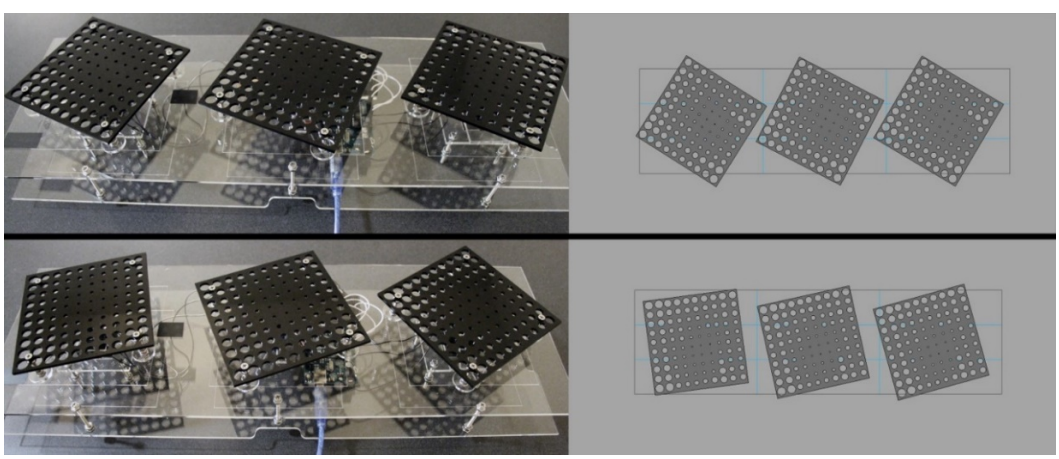

Figure 5.5 The manual rotation of Panel 1 and/or 3 will automatically rotate Panel 2 using the generated parametric equation. Images on the top and bottom show consistent results when the angles of rotation for the panels are changed (Al-Qattan et al., 2017a).

The live link established between the TUI and the digital model enables real-time interaction and response in both environments. Panel 2 in the artifact rotates using a servomotor and in the digital model using an *angle* parameter.

3. Results

Prototype 1 has shown that utilizing curve-fitting in this workflow assisted in generating the mathematical equation that best depicts the relationship between the panels. The objective of Prototype 1 is not to merely determine the angles of rotation for Panel 2, which can be found by simple interpolation between the angles of rotation for Panel 1 and 3 but to determine the linear function and its coefficients representing the parametric relationship in the array of panels. This approach enables the designer to make changes to the model once the relationship is set up, and the design intent – object relationships - will remain intact. For this example, the TUI was linked to a BIM model in Revit using Dynamo and Firefly. BIM authoring tools are useful for modeling and documenting architectural designs, and Firefly offered functionality for collecting, managing, and translating data from the TUI to the digital model.

5.1.2. Polynomial Equations

Prototype 2 explores creating curvilinear geometric configurations using complex types of mathematical equations as an approach for capturing design intents in a digital model. The work explores creating parametric models based on high-degree polynomial equations. Like Prototype 1, this prototype includes three procedures: data

logging, data analysis, and generating the equations. The regression model for this example also uses a curve-fitting function for establishing data relationships.

The artifact consists of a workbench representing an architectural space with its roof made of an array of eight louvers (Figure 5.6, left image). The louvers are the objects used for tangible interaction by the designer to operate the system. The TUI is linked to a digital model of the space created in Rhino 3D (Figure 5.6, right image). The purpose of the artifact emphasizes the architectural properties of the model by providing a spatial context and elements.

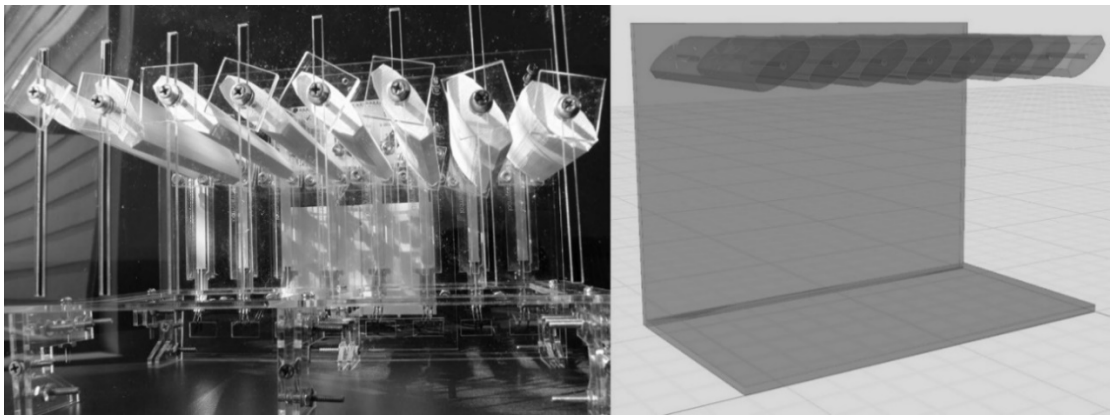

Figure 5.6 Prototype 2, TUI includes an artifact composed of a workbench representing and architectural space and louvers as design objects for user interaction (left image). The architectural model is reconstructed in Rhino and linked to the TUI (right image).

1. Prototyping

Phase 1. The set of tools used for developing Prototype 2 include:

- Software tools: Rhino 3D, a NURBS modeler; Grasshopper, a visual programming plugin for Rhino; CS-script, a programming language; Math.Net Numerics, a module for numeric computation, and Firefly.

- Hardware tools: Arduino MEGA microcontroller and eight linear SoftPot (ribbon) sensors.

Phase 2. The modeling problem involves creating curvilinear geometric configurations using polynomial equations. Prototype 2 will generate the equation, which will be used for creating the roof in the Rhino model by having the designer manually translating the louvers in the artifact. The process is similar to the previous prototype where sensors attached to design objects to monitor their physical changes and collect data for analysis. Sensors measure the distance between their corresponding louvers and the "floor" in the architectural model. The specifications for Prototype 2 are shown in Figure 5.7. Additionally, this workflow includes a unidirectional link allowing for data transfer only from the TUI to the digital model.

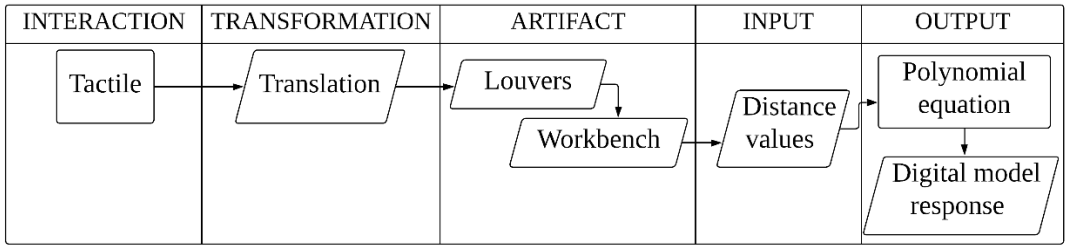

Figure 5.7 Prototype 2 specifications for developing the TUI.

2. Testing

The TUI is operated by having the designer translating the louvers vertically for creating the roof configurations in the artifact. Sensors attached to the louvers start collecting distance values during user interaction. Sensor values are then sent to Grasshopper and are plotted as geometric points in the Rhino viewport using the XZ

plane. Each point generated in the digital model is a geometric representation of the louvers' current position in the artifact. These points are used for regression analysis using curve-fitting. The results of the process include a polynomial equation and a geometric representation of the equation (a curve). The louvers in Rhino will be redistributed along the generated curve, having equal spacing between them, to replicate the roof's configuration in the artifact. Additionally, any changes made to the louvers' layout in the artifact will translate directly into the Rhino model, thus updating the equation and parametric curve. Figure 5.8 shows the workflow for Prototype 2.

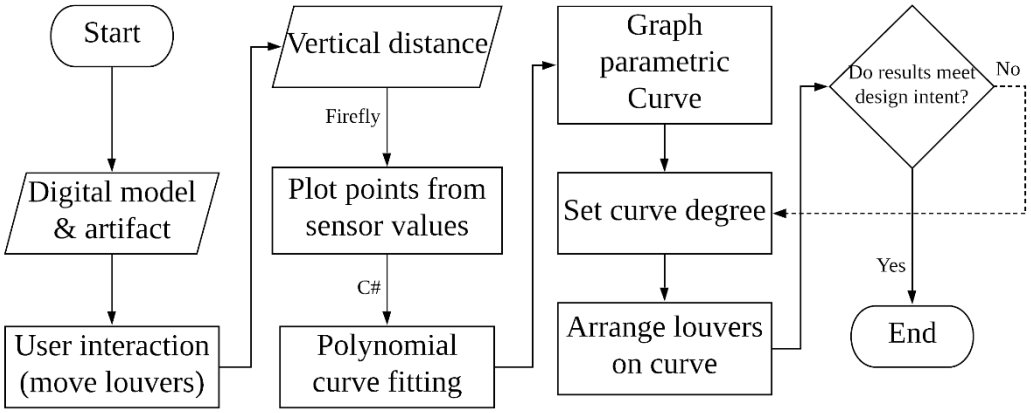

Figure 5.8 The workflow for Prototype 2 adapted from Al-Qattan et al. (2017a).

The implementation of the regression analysis in this example uses CS-script and Math.Net Numerics, which is adapted from the discussion posted on the Grasshopper forum under the title How to Find Mathematical Functions of Curves (Rutten, 2015). The regression model is modified and integrated within the Grasshopper algorithm created for this prototype to generate the equation. Utilizing Math.Net Numerics within

the Grasshopper algorithm assists in avoiding the extensive programming of the curve-fitting procedures, which was done for Prototype 1, in addition to the flexibility of making modifications to the generated equations, i.e., fine-tuning, which will be further discussed later in this section. The equations generated using Prototype 2 are for second-degree orders and higher, e.g., quadratic, cubic, quartic, and so on. Eq.5.3 shows an example of a single variable polynomial (univariate) equation and its format.

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 = 0 \qquad \text{Eq.5.3}$$

Figure 5.9 (Graph 1) shows the process of generating the polynomial equation and curve for reconfiguring the roof in Rhino. Sensor values are plotted as points in the Rhino viewport and represented in the graph as crosses. Each cross is labeled with its corresponding sensor to assist the designer in associating digital information with physical object state. The curve-fitting function used finds the best fit polynomial curve between the eight geometric points. The regression model generates a geometric polynomial curve (Graph 2) and a parametric equation as a mathematical representation of the geometric curve. The curve is then used to rearrange the louvers in the Rhino model (Graph 3).

The prototype is retested for regenerating polynomial equations by having the louvers rearranged in the artifact. The results show that the Rhino model updates instantaneously with the designer's real-time interaction with the louvers. Figure 5.10

shows the rearrangement of the louvers layout for producing a different curve configuration and producing a polynomial equation of fourth-degree (Eq. 5.4).

$$f(x) = -0.007x^4 + 0.123x^3 - 0.540x^2 + 0.122x^1 + 7.057x^0 \qquad \text{Eq.5.4}$$

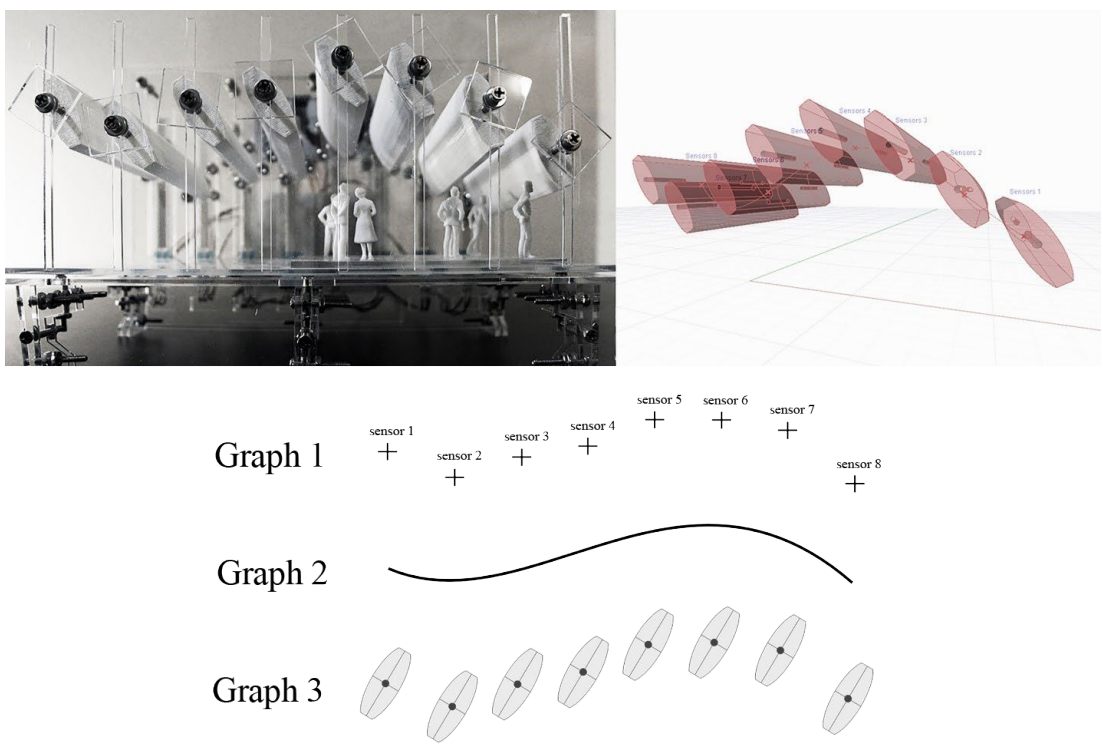

Figure 5.9 The louvers' layout in the artifact (top left), and in the Rhino after performing curve-fitting (top right). Graph 1 shows sensor values plotted in Rhino and labeled with their corresponding sensors. Graph 2 shows the generated best-fit curve between the points. Graph 3 shows the louvers redistributed across the polynomial curve in Rhino. The polynomial curve generated for this example is of third-degree as shown in the equation. Note that the louvers' angle of rotation (Graph 3) was modified later in Grasshopper to match the artifact as seen in the top two images. The generated equation in this example is: $\boldsymbol{f(x) = -0.024x^3 + 0.235x^2 - 0.515x^1 + 5.918x^0}$. Figure adapted from Al-Qattan et al. (2017a).

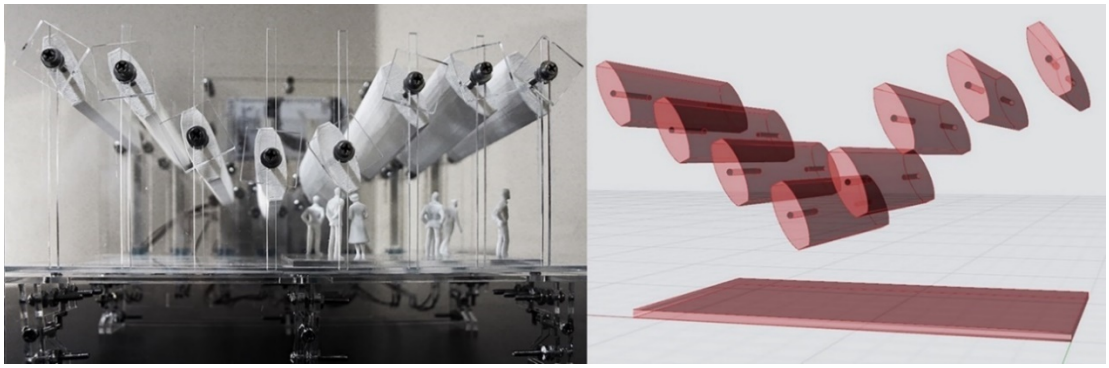

Figure 5.10 A fourth-degree polynomial curve generated by rearranging the louvers in the artifact (Al-Qattan et al., 2017a).

The louver's layout in Rhino can be fine-tuned to match their physical counterpart by increasing or decreasing the degree of the polynomial curve. Figure 5.11 shows that the louvers layout, previously shown in Figure 5.10, can be further adjusted by increasing the polynomial degree of the generated curve from four to eight. The generated equation for a fourth-degree polynomial is shown in Eq. 5.5 and for eighth-degree is shown in Eq. 5.6.

$$f(x) = -0.007x^4 + 0.123x^3 - 0.540x^2 + 0.122x^1 + 7.057x^0 \qquad \text{Eq.5.5}$$

$$f(x) = 0.001x^8 - 0.031x^7 + 0.380x^6 - 2.817x^5 + 13.171x^4 - 38.432x^3 + 66.612x^2 - 61.928x^1 + 31.312x^0 \qquad \text{Eq. 5.6}$$

The result also shows that an eighth-degree polynomial curve is the best fit among the points, however high degree order polynomials are undesirable for design applications as

they are complicated to construct and represent, therefore NURBS curves are utilized widely and are further discussed in Prototype 5.

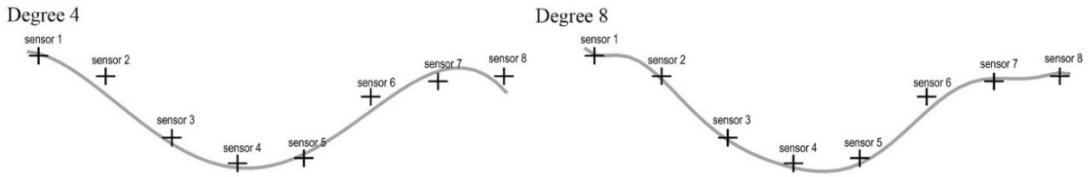

Figure 5.11 The fine-tuning process of the polynomial curve from fourth-degree to eighth-degree (Al-Qattan et al., 2017a).

3. Results

The work has shown another approach for interpreting tangible interaction using regression analysis. This example does not focus on the individual relationships of objects as in Prototype 1 but on the process of creating complex parametric forms. Additionally, the louvers' overall relationship is associated with the curve, as the curve is reconfigured the louvers' layout responds accordingly. For example, if the designer changes the number of louvers by increasing or decreasing their number (in both the digital model and artifact), the updated array of louvers automatically relocates along the curve maintaining the overall configuration and degree of the curve. The coefficients of the polynomial equation are considered as the parameters, changing them enables the fine-tuning of the curve to explore unique design options for the roof.

This method of generating curves for parametric modeling is different from the widely used method for generating NURBS curves. User input points in this example are a direct translation of the louvers' configuration, and the polynomial curve is created by

generating an interpolated curve that passes through them. As for NURBS, input points are used as control points for creating low degree Bézier curves, which then are pieced together for creating a compound curve. The method for constructing NURBS curves will be discussed in the TUI Improvements section of this chapter in Prototype 5.

5.2. Algorithmic Rules

Prototype 3 explores the potential of tangible interaction to setup algorithmic rules for parametric modeling to create and manage complex geometric patterns. The purpose of the study is to address the challenges of digital modeling associated with computer programming. This work was published in the Proceedings of *the 35th eCAADe conference*, "Tangible Computing for Establishing Generative Algorithms: A Case Study with Cellular Automata" by Al-Qattan, Yan, and Galanter, 2017b. updated figures are added to this work in addition to the published material.

The workflow created for Prototype 3 (Figure 5.12) tests the TUI for providing two types of data inputs; the rules and initial cell state for a CA algorithm.

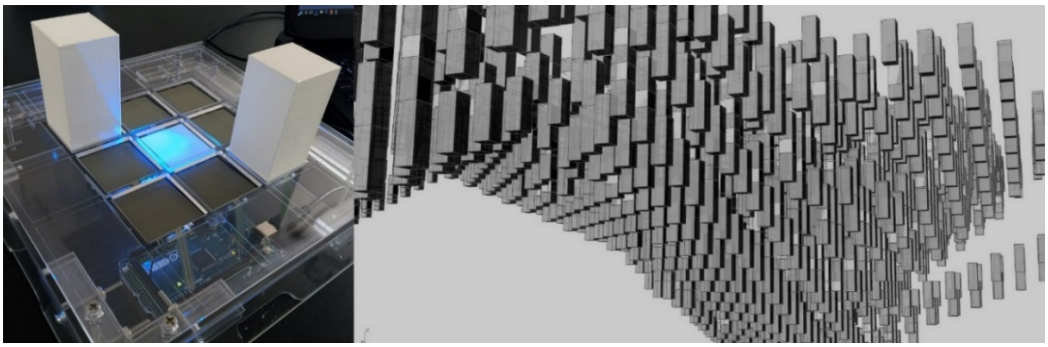

Figure 5.12 The TUI developed for Prototype 3 (left) which uses CA for generating 3D geometric patterns (right) (Al-Qattan et al., 2017b).

The prototype like the previous examples links a TUI to a digital modeling environment for generating three-dimensional geometric patterns using CA rules. The artifact in this example demonstrates a level of abstraction as design objects are represented as solid geometric forms with no distinct architectural design features. However, the objects and workbench are direct representations of CA elements (the grid and cells), which can assist in making the association between both the physical and digital models.

CA is an EA, which, has been extensively explored as a design tool in architecture (Cruz et al., 2016). Conway's Game of Life (Life) is an example of CA's applications, and it is a digital simulation of patterns that exhibit emergent behavior much like living organisms (Gardner, 1970; Krawczyk, 2002). Life is represented by an infinite two-dimensional lattice where each cell can have one of two possible states, alive or dead. Life also implements a simple set of rules, which determines the state of a cell in future generations (Fazer, 1995). Below is a sample set of CA rules which Conway developed for Life:

- A cell is *born* if it has *three* live neighbors.
- A cell *remains alive* if it has *two* or *three* neighbors.
- A cell *dies* if it has fewer than *two* live neighbors.
- A cell *dies* if it has more than *three* live neighbors.

5.2.1. Prototyping

Phase 1. The set of tools used for developing Prototype 3 are:

- Software: Rhino 3D; Grasshopper; Firefly; and Rabbit, a CA component and plugin for Grasshopper.
- Hardware: Arduino UNO, and eight pressure-sensitive sensors.

The workbench for Prototype 3 consists of a 3-by-3 square grid representing a single cell neighborhood, which is considered as a simplified version of CA's infinite lattice, and a total of 3 blocks representing the alive cells. Each cell in the 9-square grid includes a pressure sensor, which links each cell to its corresponding square in the Rhino model as seen in Figure 5.13.

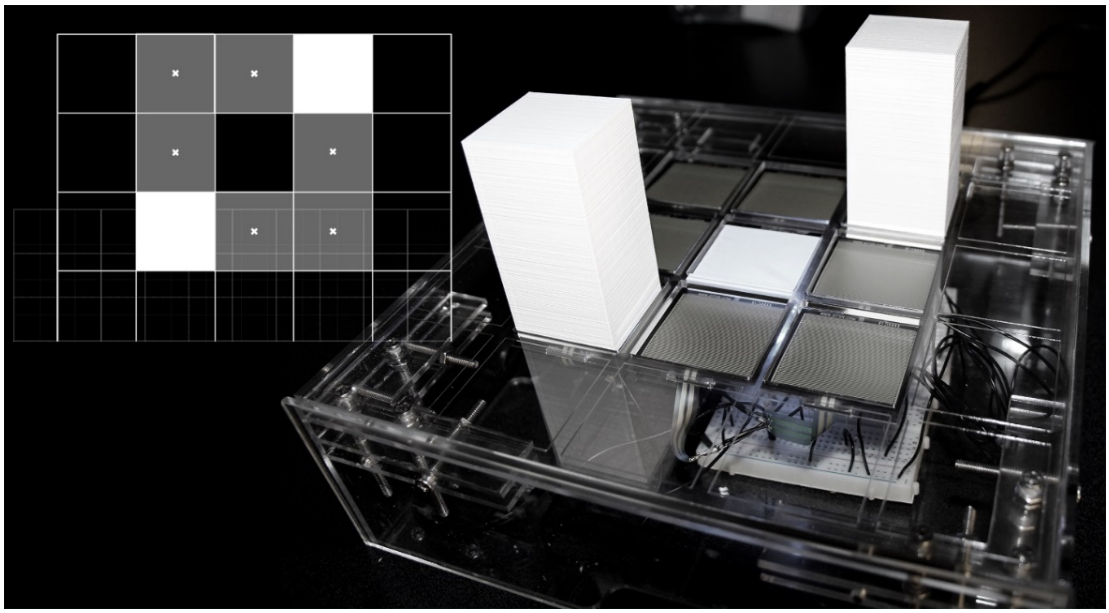

Figure 5.13 The 9-square grid in Rhino (left) and TUI (Right). Each cell in the grid is linked to its corresponding sensor in the workbench. The two highlighted cells (white) in the Rhino model show the alive cells as an example of how the TUI communicates with the digital model. The ninth cell (center square) in both the model and the artifact is the initial cell which will be generated based on the rules (Al-Qattan et al., 2017b).

Phase 2. The designer interacts with the artifact by adding and removing blocks on the workbench. The artifact will provide two types of inputs (1) the number of blocks (alive cells) and (2) their configuration. Two workflows are tested using the TUI for Prototype 3.

Workflow 1 tests the prototype for setting up the initial cell configuration for the CA algorithm and is referred to as the seed (Figure 5.14). Unique seeds are generated by changing the blocks' layout on the workbench, moving them from one cell to another. It is important to note that, this work is intended to generate a three-dimensional geometric configuration using the generative algorithm. Unlike two-dimensional CA, which has only two possible states, this work will include a third state, a surviving cell that will inform later generations of the pattern's evolution.

Workflow 2 tests the prototypes for generating CA rules by counting the number of blocks placed on the workbench (Figure 5.15). The number of blocks used will determine the number of surviving cells in the algorithm's evolution in the digital model. The initial assumption is that these inputs will enable the generation of geometric compositions in Rhino, and as these inputs are changed, the overall geometric system will respond accordingly.

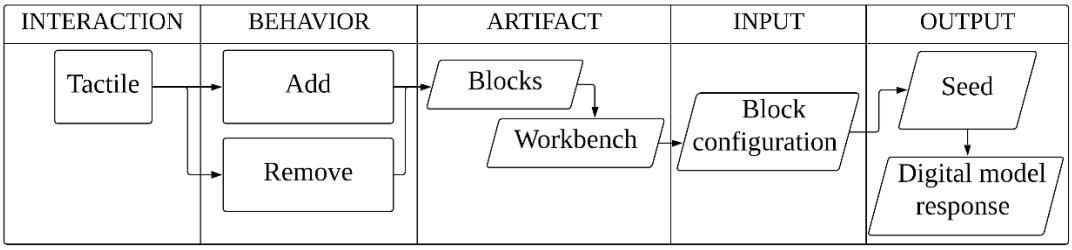

Figure 5.14 TUI specifications for Prototype 3 - Workflow 1.

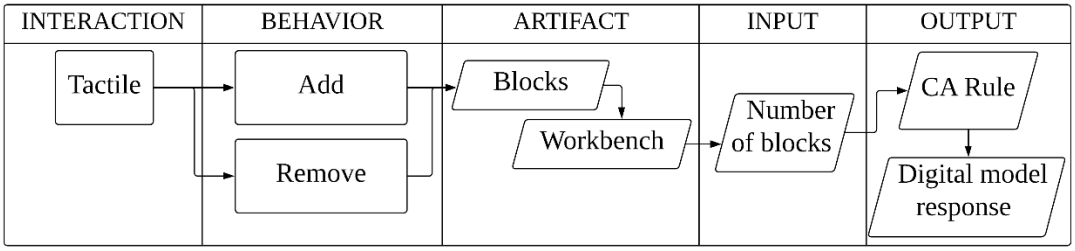

Figure 5.15 TUI specifications for Prototype 3 - Workflow 2.

5.2.2. Workflow 1

Block configuration in Workflow 1 is set by using the artifact, and the number of neighboring cells is set manually in Grasshopper. The objective of this work is to test the TUI for generating and manipulating the seeds for the CA algorithm. The Grasshopper algorithm utilizes the plug-in Rabbit, which is a three-dimensional CA generator. The CA rule set in Grasshopper for the generator to start the evolutionary process is: *A cell is born and survives if it has at least two neighbors, else it dies in the next generation*. As for the seed, the TUI will use the artifact to determine the blocks' configuration. The seeds will guide the geometric composition's evolution in Rhino. Figure 5.16 shows the workflow for generating the seeds using Prototype 3.

The workbench detects which sensors in the grid are used. Sensors are activated by weight when blocks are placed on them. Sensor data provides the system with the current locations of the blocks, which are then used as an input parameter for Rabbit to create the custom seed. Activated sensors represent alive cells in the CA grid and changing the blocks will reconfigure and update the seed as shown in Figure 5.17. Three seeds are generated using the artifact with each having a different number of neighbors and a unique cell configuration. Seed inputs must follow the CA rule set in Grasshopper, which requires having at least two alive neighboring cells to start the evolutionary process.

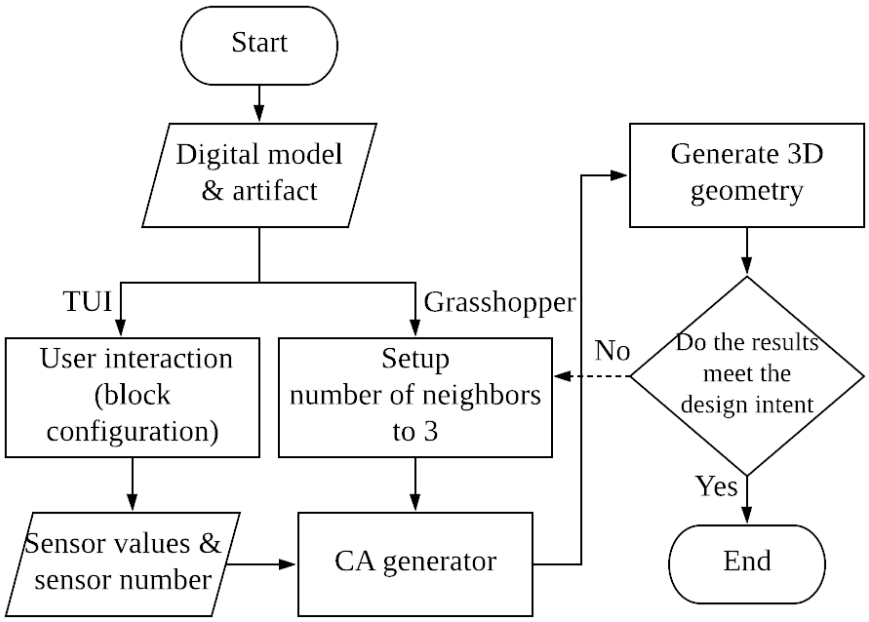

Figure 5.16 Workflow 1, the inputs for generating the Seeds. Sensors are used to indicate which cells are used in the grid to determine the seed's configuration. Figure adapted from Al-Qattan et al. (2017b).

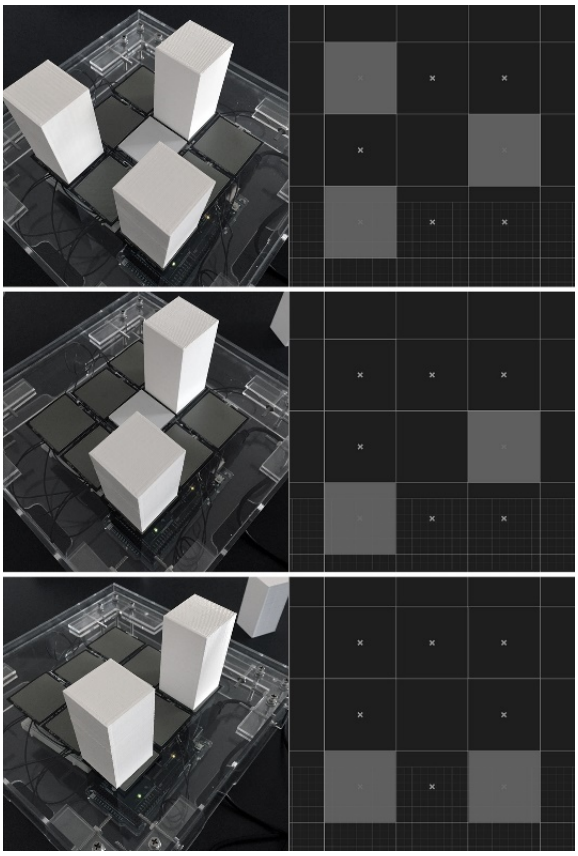

Figure 5.17 Block configuration for generating custom Seeds. The top image shows a configuration generated having three blocks, and both the middle and bottom images having only two blocks but with a different layout (Al-Qattan et al., 2017b).

The seeds are then used for generating three-dimensional patterns in the Rhino model. The workflow is tested using three seeds as shown in Figure 5.18. The test shows that when physical blocks are relocated on the workbench to create a different configuration, the digital model responds by producing a new three-dimensional pattern in Rhino.

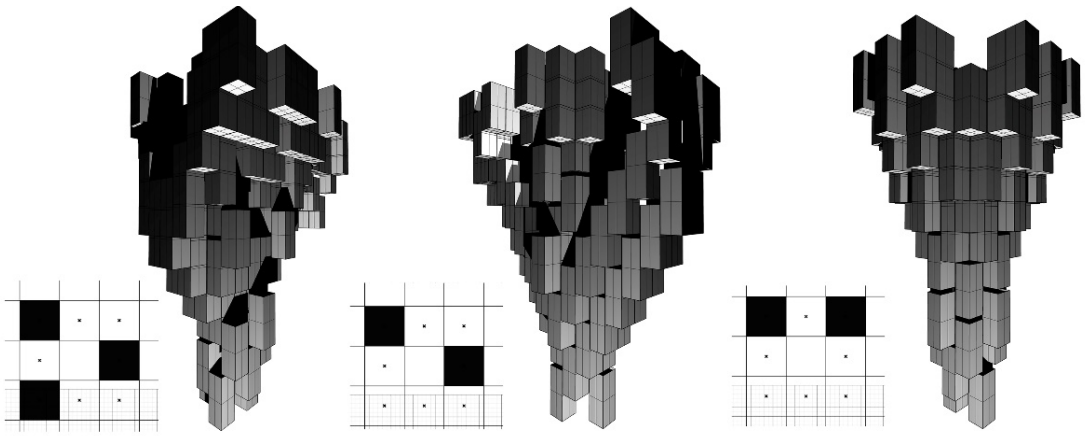

Figure 5.18 Geometric patterns generated in Rhino using three Seeds. Each level of blocks arranged vertically is one generation of the CA evolutionary process (Al-Qattan et al., 2017b).

5.2.3. Workflow 2

This test focuses on setting up CA rules using the TUI by providing the number of alive cells as the input for Rabbit. The graph illustrated in Figure 5.19 explains the process of setting up the rules for the algorithm. The rules for the CA component which is set up using the TUI are:

- Rule 1: a cell is Born and survives if it has one neighbor.
- Rule 2: a cell is Born and Survives if it has two neighbors.

As for the seed, it is generated using a 15-by-15 grid lattice and by selecting a random neighboring cell configuration in Grasshopper.

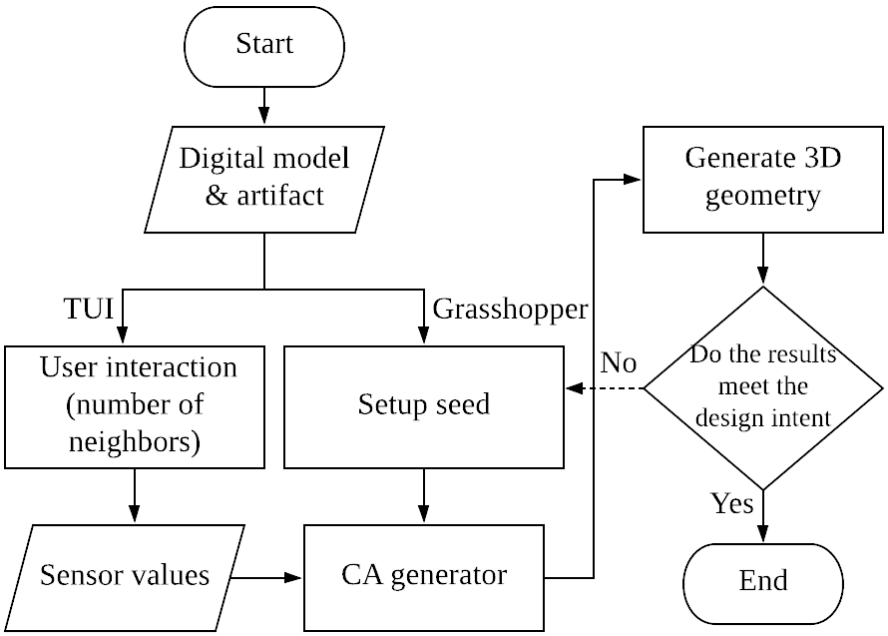

Figure 5.19 Workflow 2, procedures and inputs for generating the rules for the CA algorithm adapted from Al-Qattan et al. (2017b).

The TUI counts the number of blocks when added or removed from the workbench to determine the number of cell neighbors. The increase or decrease of neighbors changes the algorithm's rule and the overall geometric outcome in the Rhino model. Unlike Workflow 1, the location of the blocks on the workbench has no effect on the rule. Figure 5.20 shows two geometric patterns generated by implementing the two rules. The left image shows the seed consisting of four randomly placed neighbors to determine the initial cell state, middle image shows the geometric pattern of Rule 1, and the right image shows the change in pattern configuration using Rule 2.

It is important to note that, the increase in the number of neighbors, adding more blocks, will also affect the evolutionary process. For example, if three neighbors are used to set up the rule, the CA evolution will only produce two generations, because the

cells will die out of the specific seed and lattice length. Moreover, if further flexibility is required by the designer to set up rules, e.g., a cell is *born if it has one neighbor and survives if it has two or more neighbors*; then such a rule must be set up in Grasshopper using a conditional statement. Using conditional statements provides the designer with the flexibility to create more complex rules to meet different design conditions.

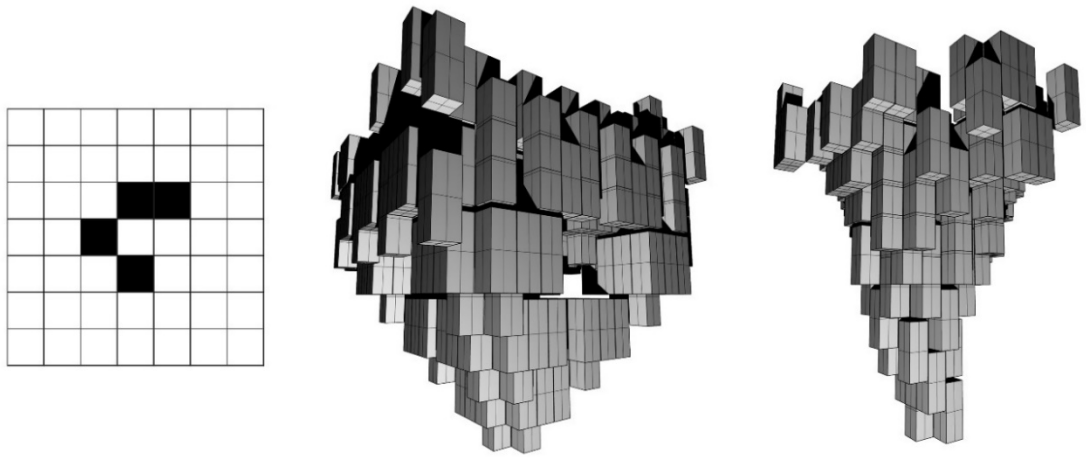

Figure 5.20 15-by-15 lattice using a random Seed (left image showing part of the lattice), the geometric pattern produced using Rule 1 (middle image) and Rule 2 (right image) (Al-Qattan et al., 2017b).

5.3. Results

Prototype 3 has shown that tangible interaction can be used for establishing algorithmic rules for digital modeling. The TUI provided the parameter inputs for generating CA custom seeds and setting up the number of neighbors for generating CA rules. The prototype has shown to be limited to setting up either seeds or rules using the TUI in this setting. A procedure must be set up in the visual program if both were required to be simultaneously set up using the same artifact.

5.4. TUI Improvements

Prototypes 1, 2, and 3 have shown a method for establishing object relationships by providing parameter inputs such as mathematical equations and algorithmic rules. A parametric framework is partially defined prior to linking the artifacts to the digital models. This section of the research focuses on expanding on the method by automating the process of constructing parametric frameworks and modeling procedures. This work was presented as a research poster at *the 8th DCC conference*, "Utilizing Tangible Computing for Parametric Modeling: Case Studies for Detecting Types of Geometric Transformations and Setting Up Constraints Through Tangible Interaction" by Al-Qattan and Yan, 2018. Updated figures and method description are added to this work in addition to the material included in the poster and abstract in the conference preprints.

Prototype 4 (Figure 5.21, left image) tests a method for detecting physical object transformations and automatically translating them in digital models. The system detects the different types of physical transformation applied to design objects and applies them to a digital model without the extensive programming of such procedures. Prototype 5 (Figure 5.21, right image) tests a method for constructing NURBS curves, establishing boundaries, and generating design options for modeling an architectural element.

Prototype 5 covers a broader range of modeling tasks usually found in a digital design process. The two prototypes expand the capabilities of earlier workflows by introducing a higher level of automation of parametric modeling procedures.

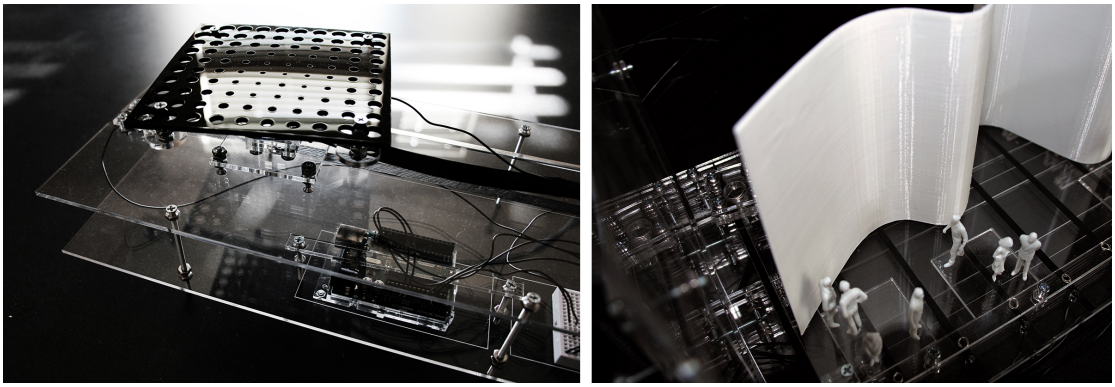

Figure 5.21 The TUIs developed for Prototype 4 (left) and Prototype 5 (right).

5.4.1. Physical Transformation Detection

The prototype is developed for automatically detecting the different types of geometric transformations applied to a design object through tangible interaction. The workflow enables designers to manipulate a Rhino model using an artifact. The improvement in this workflow, which distinguishes it from prior examples is that the algorithm created in Grasshopper does not include a set of predefined geometric operations. The algorithm is designed to distinguish between the different types of analog data inputs and use this information for manipulating digital objects using a *transformation matrix*. Figure 5.22 shows the TUI for Prototype 4 which consists of a single detachable panel, a workbench, and a Rhino model of the panel. Testing the prototypes have shown that tangible interaction can be used for applying both compound and non-compound geometric transformations instead of having to create programming graph for each type of transformation.

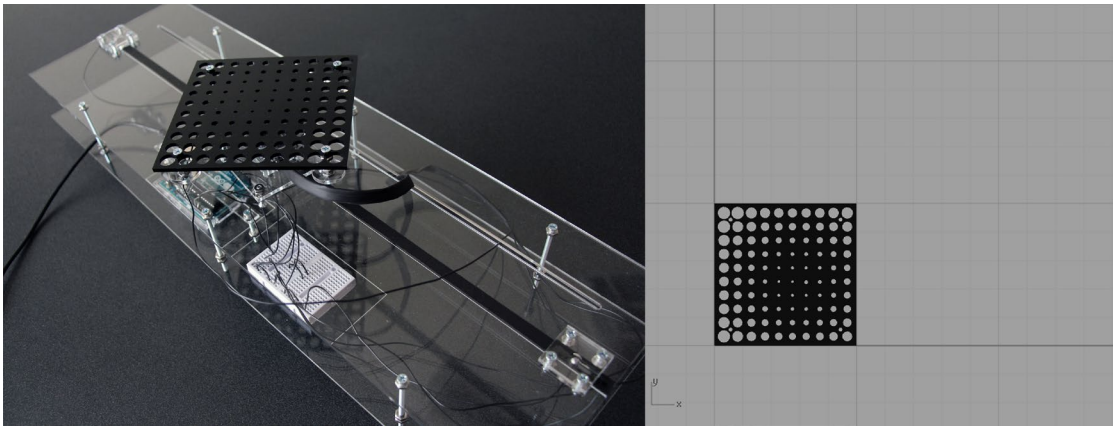

Figure 5.22 Prototype 4, the TUI consisting of a detachable panel and workbench (left image) and a Rhino model of the panel (right image).

1. Prototyping

Phase 1. The set of tools used for developing Prototype 4 includes:

- Software: Rhino 3D, Grasshopper, and Firefly.
- Hardware: Arduino MEGA microcontroller and custom-made sensors using conductive paint (Bare Conductive, 2009).

Phase 2. The design object for Prototype 4 is represented by a square panel, and unlike previous prototype examples in this research, the panel is not considered as part of a more significant design problem (e.g., the panel being part of a tessellated surface), only as an object for facilitating tangible interaction. Therefore the artifact in this example can include any geometric configuration for testing the system for detecting transformations.

Prototype 4 is operated by having the designer manually translating and rotating the panel. The artifact provides the digital model with two types of inputs, the panel's angles of rotation and distance from the horizontal length of the workbench. The system

processes this information to replicate the panel's physical state in the digital model. The specifications for Prototype 4 are shown in Figure 5.23.

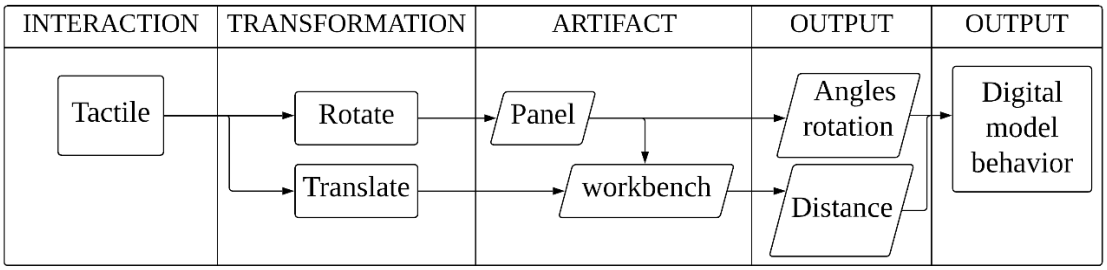

Figure 5.23 TUI specifications for Prototype 4.

An essential feature of the artifact is the detachable panel, which creates a more flexible interactive medium. The designer can remove or add additional panels to the artifact. Figure 5.24 shows the detachable panel. Prototype 4 includes a new method for embedding the sensors and the electrical circuit within the artifact using conductive paint. This approach allows for a higher level of physical computing systems and design objects integrations, and TUI customization. Previous models used off-the-shelf sensors, which did to some extent limit the design and fabrication of the physical objects. For this example, the panel is designed to include a rotary potentiometer to monitor changes in the panel's angles of rotation. The workbench is designed and fabricated to include a linear ribbon sensor to monitor the panel's position when translated across its surface.

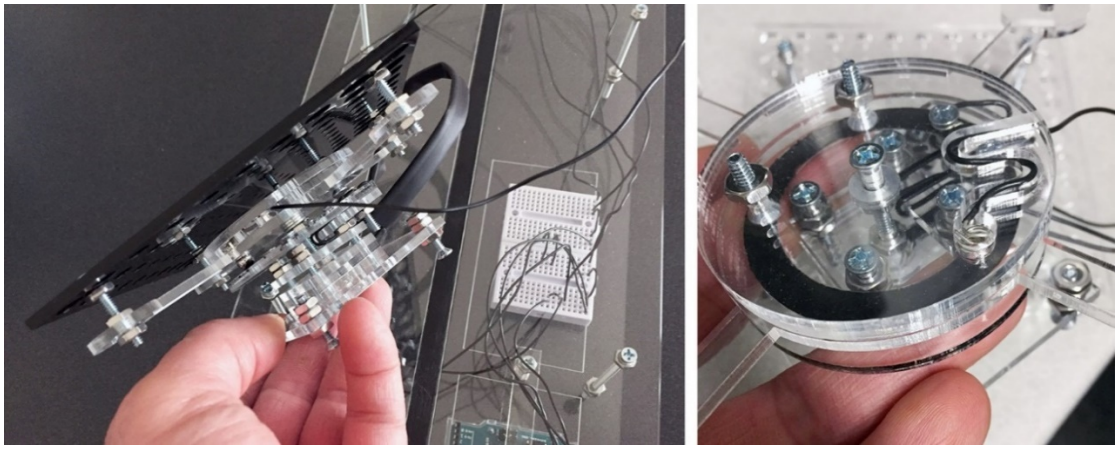

Figure 5.24 The detachable panel and workbench: left image shows the panel lifted from the workbench below, and the right image shows circuit-object integration (rotary potentiometer).

Figure 5.25 describes the digital workflow for Prototype 4 and the mechanism for detecting physical transformations.

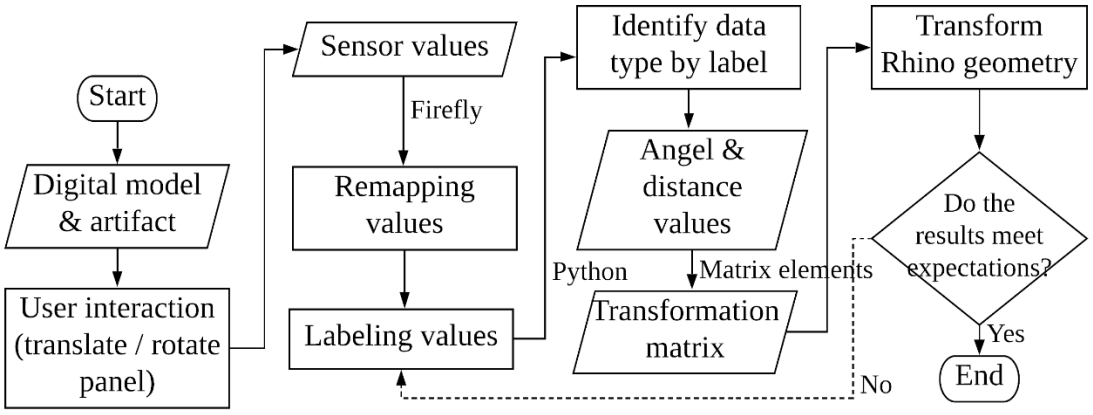

Figure 5.25 The workflow for Prototype 4.

The sensors monitor the panel's rotation around its center and translation across the workbench's length. Values from both sensors are sent to the algorithm written in

Grasshopper where they are remapped and matched with units of measurement, referred to as labels; i.e., the labels “degrees” is used for rotation values and “inches” for translation values. The algorithm will then use the units of measurement to identify each incoming value and send it to its corresponding element in an original identity matrix to create the transformation matrix for manipulating the Rhino panel. Regardless of which of the two sensors is operated, linear or rotary sensors, the algorithm will automatically detect the types of transformation using the labels given to each of the two values.

In mathematics, a Transformation Matrix is a method for working with and representing linear and non-linear transformations, such as rotation, translation, scale, shear, perspective, and their combinations. A transformation matrix is a useful tool in digital modeling, as it allows for multiple transformations to co-occur when manipulating the same geometry. A matrix is represented as a rectangular array of numbers, with its dimension *m*-by-*n*; *m* being the number of rows and *n* being the number of columns (Issa, 2013). Prototype 4 is tested for rotation and translation, which are two types of Affine transformations. *Affine transformations* modify a geometric object’s shape while keeping the rest of its properties unchanged, e.g., parallelism of lines (Mitchell, 1990). Figure 5.26 shows a 4-by-4 matrix in Grasshopper for three-dimensional transformation. The 16 values of the matrix are set manually. The matrix example shows a compound type of transformation for translating and rotating a panel from its Start Position (*P*) to its Target Position (*P’*). For panel translation, the cells of the far-right column of the matrix are changed to 8 inches along the X-axis and -8 inches along the Y-axis. For panel rotation, the two left columns are changed to 17 degrees.

Rotation requires matrix multiplication to obtain the Target Position's angle of rotation in degrees prior to inserting the number value in its corresponding cell.

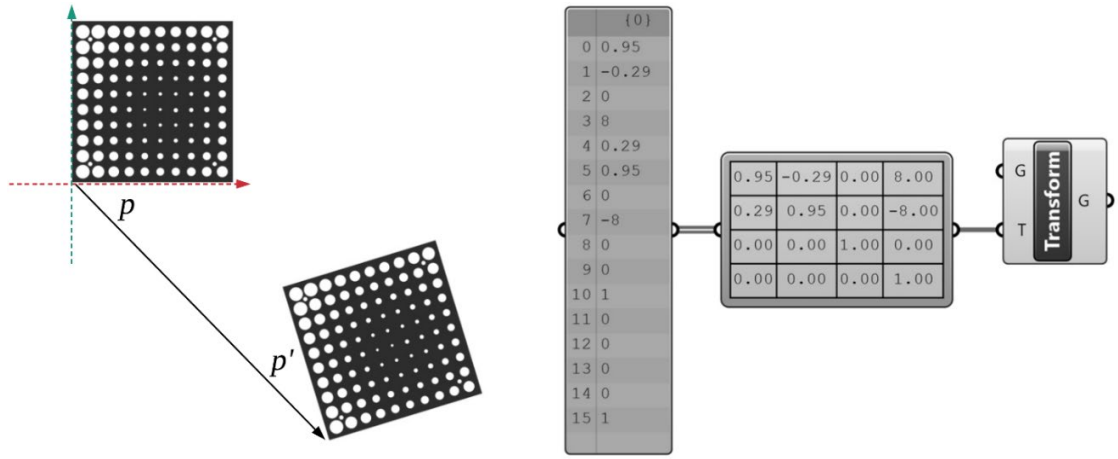

Figure 5.26 Shows a compound transformation, a Rhino panel is translated along the X and Y axes and rotated counterclockwise.

2. Testing

The prototype is tested for three transformation scenarios; translation, rotations, and their combination. For translation, the panel is moved across the workbench along the linear ribbon sensor. The Grasshopper algorithm automatically detects the panel's physical state. Figure 5.27 demonstrates the panel's translation from *P* to *P'* on the workbench. The figure also shows that the sensor value is inserted in the top right cell, which is used for translation along the X-axis. The example shows the panel moving from 5.12 inches (*P*) to 16.81 inches (*P'*) starting from the left side of the workbench moving towards the right.

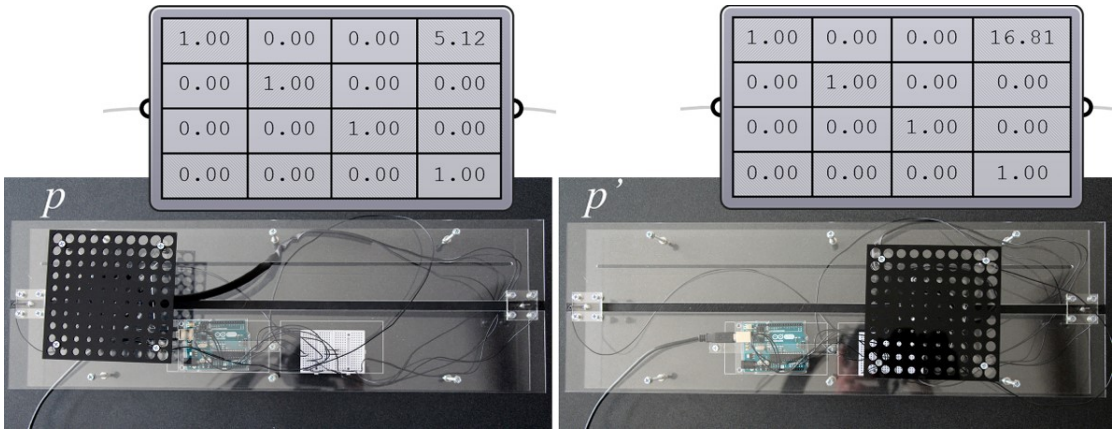

Figure 5.27 Panel translation from the Start Position (left) to the Target Position (right).

For rotation, the panel is rotated around its center, activating the rotary sensor. The Grasshopper algorithm automatically detects the type of transformation applied to the panel using the units of measurement. Unlike translation where sensor values are inserted directly into the matrix, the obtained angle values are used to calculate the corresponding cell values for the panel's Target Position. A matrix multiplication equation for counterclockwise rotation is implemented in Grasshopper for calculating the panel's Target Positions prior to inserting the values in the matrix. Figure 5.28 shows the panel being rotated from *P* to *P'* (top three images), the calculated values inserted in the transformation matrix and the panel's response in Rhino (bottom right image).

For compound transformations, Prototype 4 is tested for both translation and rotation. The two types of transformations are applied using the physical panel. This process was repeated several times while monitoring both the physical and digital panels. The values obtained from each type of two sensors, as shown in Figure 5.29,

indicate that the system is correctly identifying each type of physical transformation. The values obtained from the artifact are transferred to the digital model and are inserted in their corresponding matrix cells to generate a compound transformation. Both digital and physical panels in Prototype 4 show consistent results when the designer interacts with the TUI.

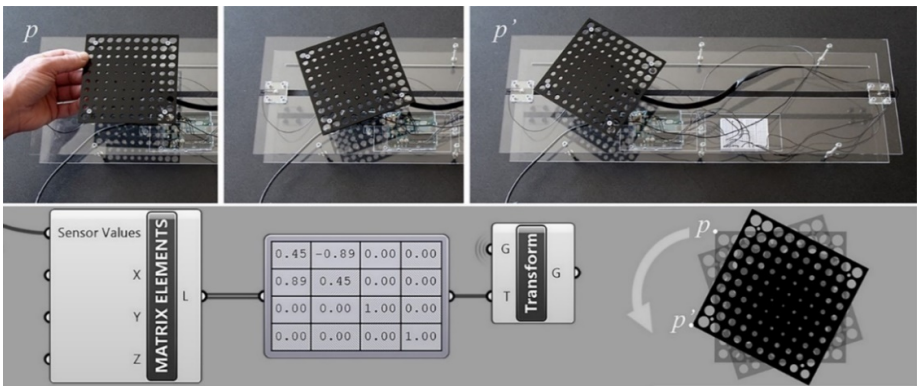

Figure 5.28 The system is tested by rotating the panel approximately 63 degrees (*P'*) around its center (top images). The angles of rotation obtained from the artifact are used to calculate the cell values before having them inserted in the upper left 2x2 cells of the transformation matrix in Grasshopper (bottom images). Adapted from Al-Qattan and Yan (2018).

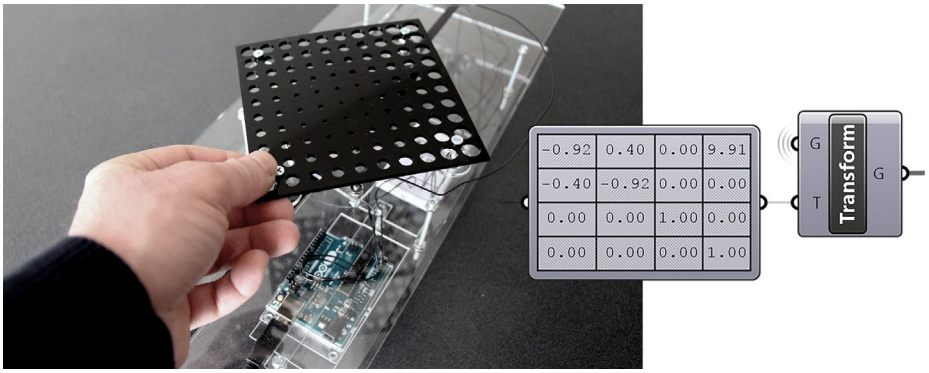

Figure 5.29 The process of generating compound transformations using the TUI. The panel rotated and translated (left image), and the values form each sensor are inserted in the transformation matrix (right image).

5.4.2. Automating Modeling Procedures

The objective of developing Prototype 5 is to expand the functionality of the workflow demonstrated in Prototype 4 for creating parametric models using NURBS curves. The artifact for this example includes a workbench which is represented as an architectural space and several blocks located on its "floor" that represent a NURBS curve control points (Figure 5.30). The intent is to use the TUI to design an interior wall within the architectural space. Although the design object for manipulation is abstracted, having no distinct design features, the architectural context is established through the physical boundaries of the model (workbench).

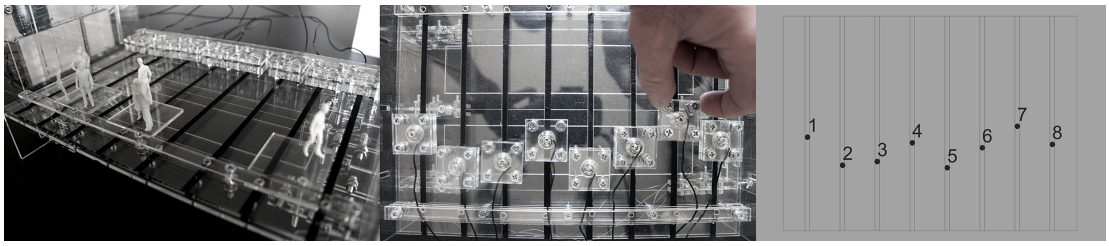

Figure 5.30 Shows Prototype 5, the TUI representing an architectural space (left), blocks representing a NURBS curve control points (middle), and a Rhino model showing the eight control points (right). Adapted from Al-Qattan and Yan (2018).

1. Prototyping

Phase 1. Similar to Prototype 4, the set of tools includes:

- Software: Rhino 3D, Grasshopper, and Firefly.
- Hardware: Arduino MEGA microcontroller and custom-made sensors using conductive paint.

Phase 2. The Blocks in the artifact are a physical representation of control points and are used to provide the input data for constructing NURBS curves in Rhino. The number of blocks added/removed on the workbench are used to determine the number of control points in Grasshopper. The artifact is designed to hold up to eight blocks. Each block is placed on one linear ribbon sensor on the workbench. These ribbon sensors are spaced out equally across the length of the workbench and are activated when a block is added to them and moved. The blocks provide the system with two types of inputs, the number of control points and their configuration. The list of specification for Prototype 5 is illustrated in Figure 5.31.

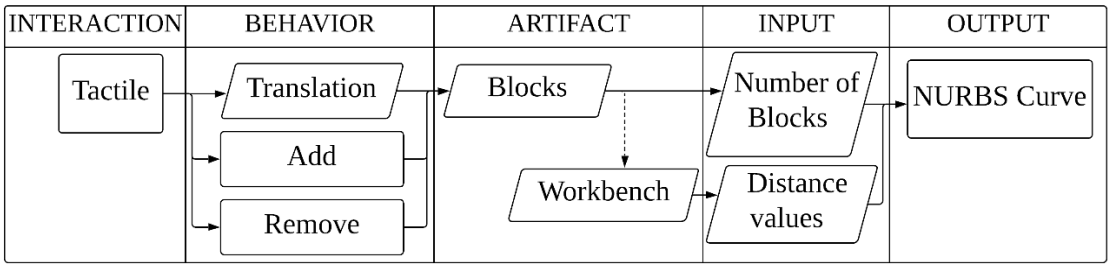

Figure 5.31 TUI specifications for Prototype 5.

The Grasshopper algorithm detects the number of blocks used and their position on the workbench to generate a corresponding control point in the Rhino scene. *Block position* refers to which sensors are used in the artifact (for example sensors 1, 3, 5, and 7) and their geometric configuration (layout on the model's floor). The block count will provide the number of control points, which is the input data required for the algorithm to construct the NURBS curves. The physical transformation detection method

developed in Prototype 4 is reused in this example for monitoring the blocks position when translated across the workbench for manipulating the generated curves' configuration.

For establishing curve boundaries, the algorithm records the preferred curve configurations and uses them as constraints. *Preferred curves* are the configurations of NURBS curves that meet the designer's intent when modeling the interior wall. Preferred curves are generated by translating the blocks in the artifact, then recorded and referred to as *Special Case* curves. These recorded curves are set automatically as boundaries in the digital model and are used for generating curve interpolation between them to explore design options for the proposed interior wall. This process is shown in Figure 4.31.

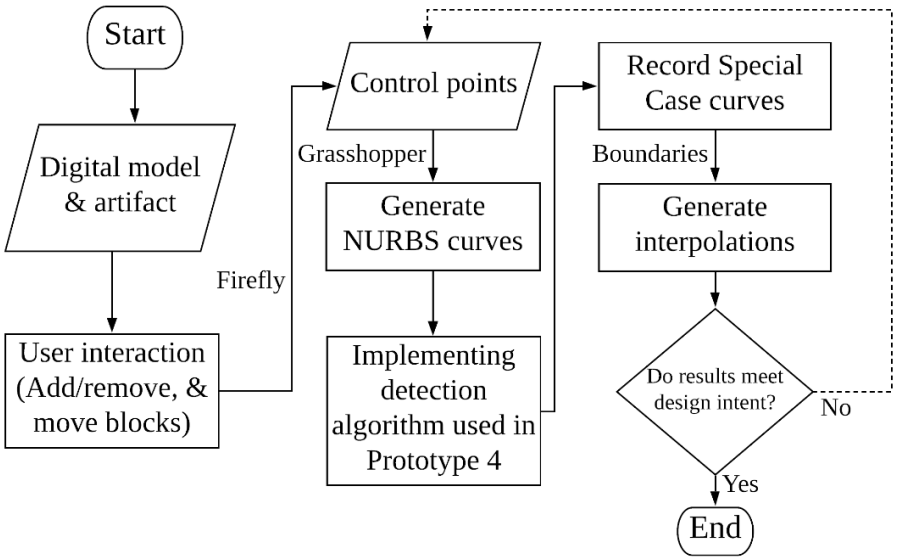

Figure 5.32 Digital workflow developed for Prototype 5.

In computer graphics, NURBS are described as mathematical representations of curves and surfaces. NURBS extensive use in digital modeling is due to their mathematical precision and intuitive control features. Designers can reconfigure NURBS objects by just dragging their control points in the 3D model's viewport (Figure 5.33). NURBS curves can be constructed using programming by providing the following inputs: dimension, degree, control points, and their weights and knots (Issa, 2013). The artifact in Prototype 5 is used to provide the input data for generating the control points to construct the NURBS curves while having the rest of the parameter inputs set with default values in Grasshopper.

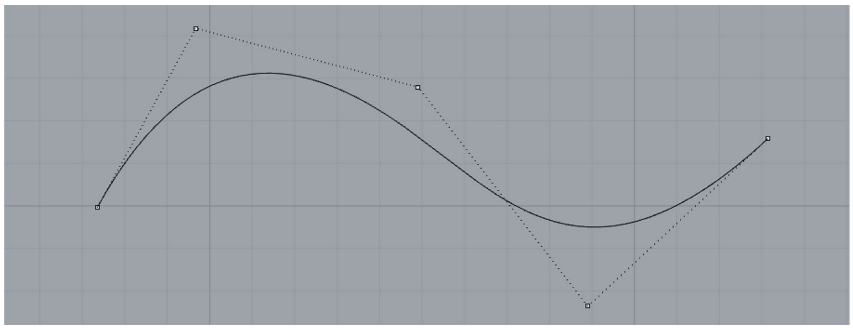

Figure 5.33 A degree 3 NURBS curve with five control points, created in Rhino.

2. Testing

The TUI provides Grasshopper with the following information: the number of control points, and their current location on the workbench. A corresponding control point is generated for each block in Rhino using the XY plane. Figure 5.34 shows constructing a NURBS curve using these inputs. The figure also shows how the number of control points is increased from four to eight by adding more blocks on the

workbench. Sensors are automatically activated when blocks are added to the workbench. Regardless of the sequence of adding the blocks, the Grasshopper algorithm detects which sensors are used. The artifact demonstrates a level of flexibility for making changes to the artifact for acquiring greater precision and control over the generated geometry. Once the number of control points is set, the sensors then start to monitor the blocks' physical state. The curve's configuration is modified by moving its control points using their corresponding blocks on the workbench to set up the Special Case curves.

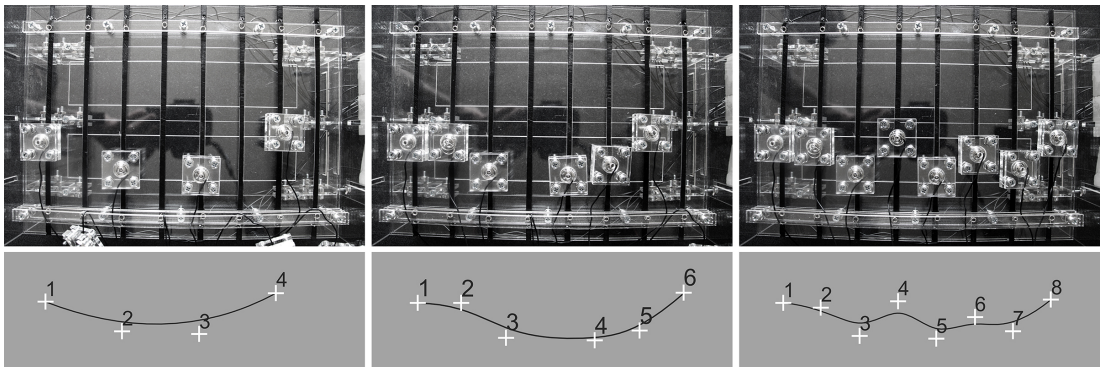

Figure 5.34 Shows the blocks and their corresponding points in Rhino for generating the NURBS curves. Figure adapted from Al-Qattan and Yan (2018).

The algorithm allows the recording of the different preferred curve configurations, (Special Cases), to be used as boundaries for limiting the interior wall's behavior. Figure 5.35 shows the step-by-step process for setting up the boundaries in Rhino using the artifact. First, the NURBS curve is constructed in Rhino then recorded as shown in Figure 5.35, Image 1. Second, the process is repeated several times depending on the number of preferred curve configurations needed. However at least

two curves must be provided to define the two extremes (minimum and maximum bounds) as shown in Figure 5.35, Image 2. The control points of each of the Special Case curves are connected using a series of lines as shown in Figure 5.35, Image 3. The lines connecting the control points between the Special Cases curves establish the constraints for modeling the interior wall.

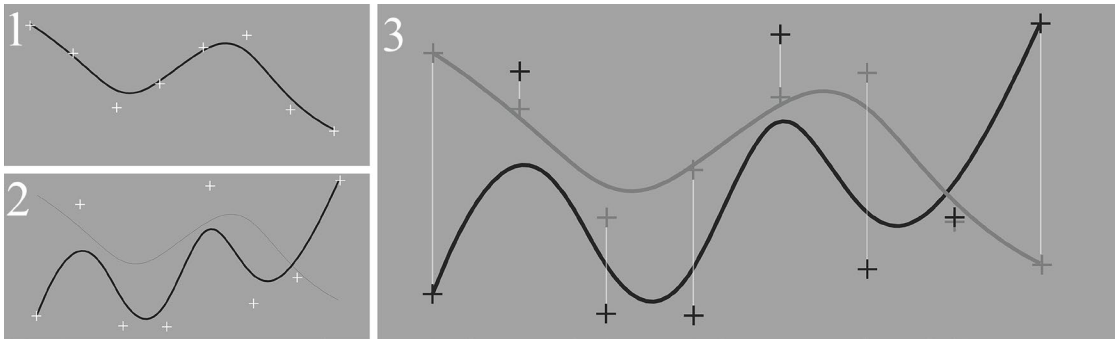

Figure 5.35 Shows the process of setting up constraints using the artifact. For the clarity of illustration, only two Special Case curves are generated and are shown in this figure.

After setting up the boundaries, they are used for generating intermediate curves, i.e., design options for the interior wall's configuration. These design options are produced by generating new control points on the series of lines connecting the Special Case curves' control points. These new control points are used as inputs for generating the intermediate curves. The process of generating NURBS curves as design options is automated by the system using the analog inputs provided by the artifact. The main feature of these intermediate curves is that they are always restricted by the Special Case curves, which assists in preserving the design intent. This process can be explained by

giving the example of a designer drafting several curves for a design study before making the final decision about a curved object.

Figure 5.36 shows two constraint scenarios (Option 1 top images and Option 2 bottom images) for generating curve interpolations for modeling the interior wall. Images on the left in Figure 5.36, show the model in the Rhino viewport (left) and images on the right show the artifact with the interior wall 3D printed and placed in the artifact.

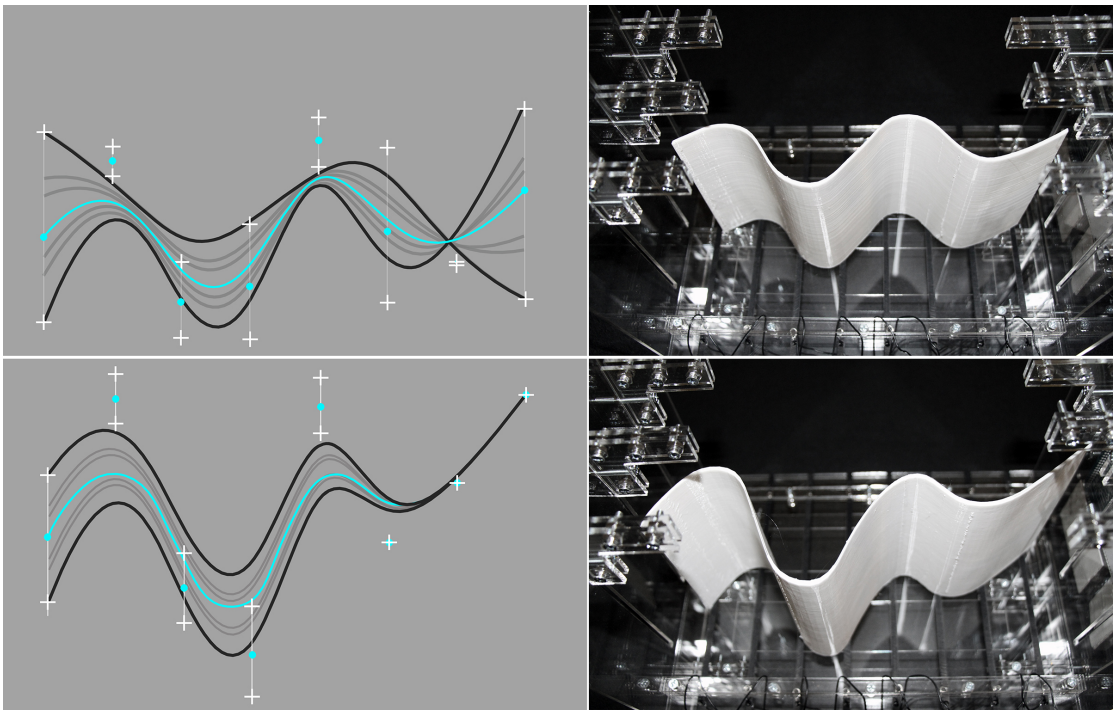

Figure 5.36 Shows the process of generating interpolated curves using the artifact (left) and the 3D printed interior walls (right). Curves are color-coded; Special Case curves (black color), samples of interpolated curves (grey), and selected curve (cyan).

As can be seen in Figure 5.36, the control points of the intermediate curves (cyan color) are on the lines that connect the control points of the two Special Case curves. The

points' location on the connecting lines is controlled by a Number Slider. The slider can move all the points of the intermediate curves together, or individually by adding more than one slider in Grasshopper. Changing the values of the Number Slider(s) will reconfigure the interpolated curves in Rhino. In the Rhino scene, several intermediate curves are generated for both design options (gray color curves). The selected curve is highlighted (cyan color) and then fabricated as the chosen option for the interior wall.

5.5. Results

Prototype 4, the workflow has shown that utilizing artifacts with transformation matrices can assist in automatically translating physical interactions into digital models for transforming design objects. Matching sensor values with units of measurement enabled the system to detect distinct types of transformations and to transfer them into the digital model correctly. This approach to modeling streamlines the manipulation of geometry without the extensive programming of such operations in the digital model. The artifact is designed for two types of transformations, and for translation only along a single axis. However, the Grasshopper algorithm can include additional transformations by simply matching sensor data with their corresponding units. Moreover, the current algorithm can also detect translation along all three axes and for multiple objects. The algorithm was reused for Prototype 5 for detecting translations for multiple objects.

Prototype 5, Each example of the boundary options shown in Figure 5.36 produces several configurations for the same wall, constrained by the special case curves. This test has shown a benefit of using TUIs for parametric modeling, creating

boundaries for preserving a design intent. The artifact also provides the flexibility of adding/removing objects (control points) as inputs for the Grasshopper algorithm. The Prototype demonstrates an interplay between physical and digital mediums, which helps in providing a platform for automating algorithmic processes and visualizing designs both physically and digitally. The Special Case curves are generated using the artifact; and once the parametric digital model is set up, it can be easily controlled in Grasshopper. The 3D printed interior wall placed in the artifact is not linked to the digital model; it is used to visualize the product of the digital workflow in its physical context.

6. EVALUATION

The prototypes developed for this work demonstrate an innovative approach for parametric modeling using tangible interaction for automating the generation of digital information and modeling procedures. This chapter focuses on providing a comparative study to evaluate the performance of the prototypes. Each prototype will be compared to the typical process of using conventional programming methods for establishing and controlling parametric models. Example algorithms and programming workflows are used for the comparison.

6.1. Algebraic Constraints

The prototypes are compared to the conventional approach of constructing, parametrizing, and embedding mathematical functions, equations, and formulas in digital models for establishing object relationships representing design intents.

6.1.1. Prototype 1

Establishing a simple linear relationship in a digital model can be a straightforward process. In Revit, a designer can set up a dimensional constraint such as *Length* between two geometric objects. The input for this relationship can be as simple as a numerical value or a function to create a more sophisticated parametric model. In complex models, relationships are established between several objects, which can be challenging for designers to express mathematically. An example is the *Reactor* model

by Woodbury (2010), a pattern example from his book *Elements of Parametric Design*. Reactors demonstrate a linear type of object relationships. In Woodbury's example, a circle's radius is controlled by a *free point*, referred to as a controller. The radius of the circle is the function of the distance between the controller and the center point of the circle. The designer changes the size of the circle by moving the controller closer or away from its center point; the circle gets smaller when the controller is close to its center and larger otherwise. The function is relatively simple for novice-programmers to create between two objects. However, using the same Reactor example for several circles (an array distributed across a two-dimensional grid) requires some programming skills, especially when the distance between the centers of these circles is not equal to the controller**.**

Considering the panel array example used in Prototype 1, in a conventional workflow, the designer sets up a parametric model by writing an algorithm in Dynamo (or any other algorithmic editor supported by the modeling platform) as illustrated in Figure 6.1. The figure shows that a relationship between the panels can be created by averaging the angles of rotation values of both Panels 1 and 3 and using the result for rotating Panel 2.

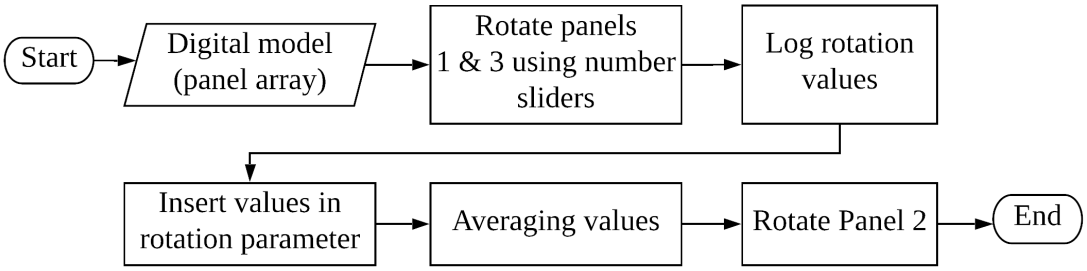

Figure 6.1 Workflow for establishing a relationship between a panel array using visual programming.

Similar to the Reactor example, a designer can easily visualize the parametric framework for controlling a single object in a digital model. However, it is challenging to write the algorithm for controlling multiple circles that are spread across a two-dimensional plane, because each circle will demonstrate a distinct responsive behavior according to its location. This is a similar problem found in multi-dimensional arrays.

Conversely, the workflow for Prototype 1 uses a regression model to deduce the relationship between the panels in the array. If the array includes three or more panels, the workflow can generate a mathematical equation for setting up the relationship. The designer can modify the current relationship by rotating the panels and having the algorithm automatically regenerating the equation with the updated coefficients. Generating and parameterizing a linear equation can be trivially done, yet the challenge is in the process of finding the mathematical coefficients for representing object relationships. Simple physical interaction, such as rotating the panels in the TUI, allows for hand-eye coordination, which provides designers with a natural way to create and modify geometric relationships, instead of manually constructing equations and inserting them in digital models.

6.1.2. Prototype 2

Testing of Prototype 2 has shown that the regression model using curve-fitting generates the correct geometric profile for replicating the louvers' physical layout in the digital model. The results other than being visually accurate in both models, the generated equation was also plotted in GeoGebra (2018) to verify the results of the algorithm (Figure 6.2). This method for validating the mathematical result was not performed for Prototype 1, because plotting a linear equation produces a straight line, which makes it difficult to immediately establish the connection between the panels' rotation and the straight line representing the mathematical equation.

Establishing a parametric model using a polynomial equation can be a complicated process. A designer can create a curvilinear profile by simply using geometric points in Rhino and then connecting them using an Interpolated Curve or NURBS Curve nodes. These points can be moved using a mouse or Number Slider in Grasshopper to modify the overall configuration of the curve. Another example of creating such curve profiles with a higher level of control is through embedding a *sine* function in the visual program, as shown in Figure 6.3.

Figure 6.2 The results of plotting the mathematical equation generated using Prototype 2 (Prototype Implementation chapter, Figure 5.9). The third-degree polynomial curve shown here matches the roof's configuration in the artifact and Rhino model.

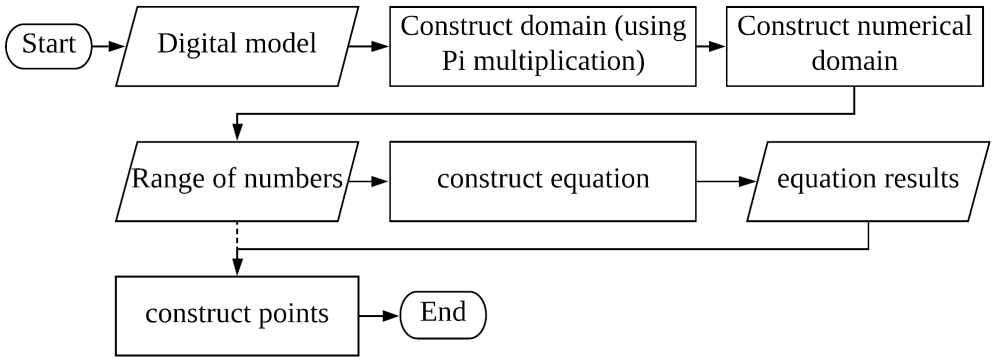

Figure 6.3 workflow in Grasshopper for constructing a curvilinear profile using a sine function. An example equation is ($\boldsymbol{a} * \boldsymbol{sin}(\boldsymbol{t})$), where *a* is the amplitude and *t* is the angle parameter. The amplitude can be controlled by inserting numerical values through a Number Slider or manually.

The use of mathematical functions for creating curvilinear objects allows for accuracy, flexibility, and additional control over the geometric object by increasing parameter inputs. For example, the designer can include variables to control amplitude,

wavelength, and frequency; which generally require a level of mathematical knowledge and programming skills.

Autodesk Knowledge Network provides a description of how to use and parametrize polynomial equations in digital models (The Mathematics Behind NURBS, 2016). This process of creating a curvilinear object follows a similar approach to the previously described method using a *sine* function. For a polynomial equation, the designer must construct the equation and solve it for both x and y using a free parameter (t) and then translate it into computer code. In this process, the assumption is that the designer is familiar with such equations and mathematical procedures.

In other occasions, the designer might already have created a curvilinear object (digitally or physically) and wishes to use it in a parametric model. In this case, the designer can approach this modeling problem by using a graphing software such as GeoGebra to find the polynomial equation and then manually embed the equation in the digital model, or to use a mathematical library such as MathNet.Numerics (Ruegg et al., 2002). Other modules such as SciPy (Jones, Oliphant, & Peterson, 2001) is commonly used with programming languages like Python for generating mathematical information from geometry.

Prototype 2 utilizes the module MathNet.Numerics in the Grasshopper algorithm. The modeling process is streamlined as the algorithm automatically translates the louvers' layout in the digital model using a curve-fitting function, especially when compared to the previous methods of creating similar geometric configurations.

Additionally, the designer can fine-tune the curve and its equation by merely increasing the polynomial degree to obtain a curve with a higher degree of accuracy.

6.2. Algorithmic Rules

The complex and distinct behavior of EAs is generated by a specific set of rules. Although the rules, especially in the case of CA, appear to be straightforward and can be described using natural language (e.g., a cell is born if it has one or two neighbors and dies if it has three neighbors), their adaptation for digital modeling requires a level of programming skills. An implementation of CA is seen in the Game of life example by Soler-Adillon (n.d.) posted on the Processing (Reas & Fry, 2001) website. The Processing code is not shown here due to its length. The algorithm generates a two-dimensional pattern based on Conway's rules of the game. This Processing example shows the level of programming skills required for implementing CA and the challenge of translating the rules from what Conway described in natural language into computer instructions.

In Grasshopper the CA generator Rabbit provides a user-friendly approach to generate CA patterns. The designer provides numerical values as inputs to define the number of neighbors for a cell to be born and for it to survive, and to create custom seed configurations to initiate the evolutionary process. However, setting up the rules by inserting numbers does not immediately establish the connection between the rules and the geometric results.

Prototype 3 was tested having two workflows, one for generating the seeds and the other for setting up the rules for a CA component in Grasshopper. Each workflow is tested separately.

The first workflow utilizes the Custom State Configuration node for the CA generator, which requires a point configuration on a 2D grid to create the seed. The blocks' configuration on the artifact was translated into Rhino as a list of points by converting sensor values from the numerical range of 0 to 1023 to an ON/OFF state; 1 for ON, if a cell on the workbench has a block on it, and 0 for OFF, when the cell is empty. Testing the system has shown consistent results between both the artifact and digital model and for creating multiple seed states.

As for the second workflow, the Life-Like Cell node requires a numerical value indicating the number of neighbors for a cell to be born and another for it to survive during the evolutionary process. These values were generated by counting the number of blocks. Counting the number of cells enabled setting up the rules for CA and changing them can generate multiple geometric configurations in Rhino.

Prototype 3 allows for a more familiar approach for utilizing CA in a digital model. The designer uses geometric objects (i.e., the blocks) to define the seed and rules. The designer physically arranges blocks on the grid by adding/removing them on the workbench to create and iterate the seed and rules. This approach of using tangible interaction may provide an approach for developing a better understanding of the logic behind some of the abstract notions of programming, and to streamline the application of computer algorithms for digital modeling.

6.3. TUI Improvements

The prototypes were tested for interpreting analog data to automatically generate geometric operations and modeling procedures instead of defining them using mathematics and computer programming in digital workflows.

6.3.1. Prototype 4

Creating compound transformations in digital models is generally done by defining a workflow having a specific sequence of geometric transformation operations. These operations must maintain the order of data nodes to avoid disruptions in the algorithm. For example, Figure 6.4 shows a workflow for using rotation and translation to transform a geometric object. In this example, a rotation is applied and followed by a translation. The designer sets up the Rotation node by providing parameter inputs such as Geometry, Rotation Plane, and Rotation Angle; and sets up the Translation node by providing the following inputs: Geometry and Translation Vector. The designer transforms the base geometry from its Start Position (*P*) to its Target Position (*P'*) to create the new geometry. Rotation is applied first in this sequence to the original (base) geometry, and the translation is applied to the resulting geometry generated by the rotation transformation. Parameter inputs are provided for the geometric operations (i.e., angles of rotation and translation values) using a Number Slider or by manually inserting the values. As shown in the figure, the process is linear and must be defined according to the sequence of operations. This example is a simple demonstration of creating

compound transformations for a single object, which must be repeated for every geometric object in a digital model, especially if each piece has a unique configuration.

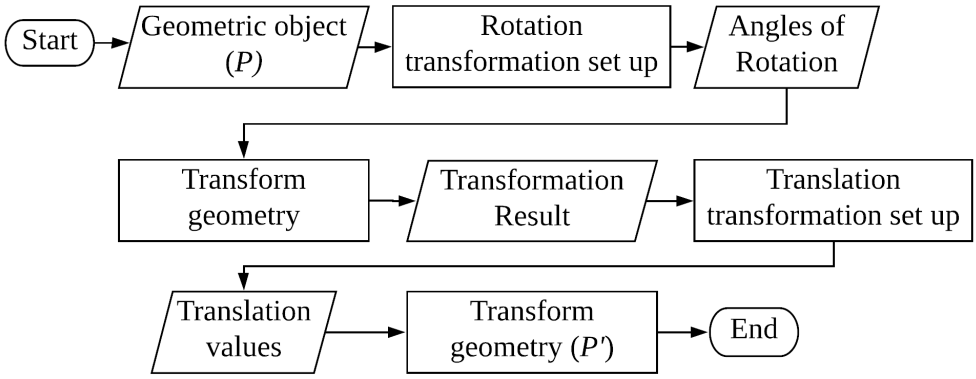

Figure 6.4 Workflow is showing the process of applying compound transformations for a single object in a digital model.

Conversely, in Prototype 4 the designer physically applies the transformations to the objects. The designer uses tactile and visual senses to provide the algorithm with transformation values. The algorithm automatically detects the type of transformation and applies it to the corresponding object in the digital model using a transformation matrix. It is important to note that the sequence of operations, which transformation is applied first, effects the result in both cases, using the TUI and the algorithmic editor. The difference is that the designer using TUIs is not concerned with defining a workflow that is strict to a finite number of objects and operations. The artifact can include more than one object and apply transformations to them simultaneously, and the algorithm can detect rotation and translation in all three axes and applying them to the digital objects.

6.3.2. Prototype 5

Prototype 5 addresses the problem of creating parametric frameworks from scratch. The modeling problem used for this example is creating a curvilinear wall using NURBS curves. The example demonstrates a typical architectural design process, modeling an object within a context. The standard practice for modeling the curvilinear wall in Grasshopper is through defining the NURBS curve's control points. Once the curve is constructed, the designer can modify its profile by merely moving the control points. A boundary is also created to limit the NURBS curve's behavior. The boundary defines the wall's range of motion through a numerical value. Boundaries can not only maintain the object within its context but also provide the inputs to explore and generate design options automatically. This is a conventional digital modeling practice for creating parametric models (creating a geometric object, setting up constraints for it, and modifying it for generating design options) which requires the designer to visualize the workflow and construct it using computer programming.

Prototype 5 demonstrates a more natural way of modeling objects and setting up constraints for them. The designer during interaction with the TUI is constructing the parametric framework of that model. The designer adds/removes blocks to create the curve's control points then starts to create different curve profiles and records them to define the boundaries. This process is similar in such a way to the analog process of sketching design elements. During this process, the designer is exploring the different configurations for the wall, rather than strictly defining it. Once the boundaries are set up, the designer starts generating the different curve interpolations in between the

boundaries, i.e., design options for the wall. This example shows a digital-physical workflow that utilizes the designer's skills to construct digital models, and the computer's power to further experiment with the design object.

6.4. Summary

Myers (1999) mentions that visual programming is widely accepted amongst designers as it takes advantage of the users' visual system. Myers (1990) further explains, "The human visual system and human visual information processing are clearly optimized. Computer programs, however, are conventionally presented in a one-dimension textual form, not utilizing the full power of the brain" (p. 3).

GUI-based programming methods do have their advantage in the parametric modeling process. However, they do demonstrate a number of challenges as research has shown in the Literature Review chapter. As Ishii (2008) explains, "Interactions with pixels on these GUI screens are inconsistent with our interactions with the rest of the physical environment" (p. xv), they do not take advantage of our haptic skills, unlike a TUI, which makes "digital information directly manipulatable with our hands, and perceptible through our peripheral senses by physically embodying it" (p. xvi).

This research has shown the plausibility for using tangible interaction for parametric modeling. In these examples, it was essential that the artifact retains the physical characteristics and design qualities in the objects. Physical design representations support the intuitiveness of interaction, as claimed and as shown in earlier research. A study by Dünser, Looser, Seichter, and Billinghurst (2010) has shown

that users were less familiar with how to operate tangible systems such as sliders, track paddles, and their variations when compared to a typical computer mouse. Tangible interfaces in their study were represented as controllers having no distinct geometric or design features. These types of computer input devices as Düser et al. (2010) state, have affected participants accuracy and time for completing digital tasks. Conversely, participants were faster and more accurate when using a mouse, which they were familiar with and use on day-to-day bases (Düser et al., 2010). In other words, context and representation make the interaction with the objects natural and intuitive as they establish meaning and substance for the designer (Dourish, 2001).

This chapter provides a qualitative comparison between the conventional parametric modeling process using computer applications and tangible interaction. The objective of the work is to demonstrate the benefits of utilizing data processing procedures and TUIs in a single workflow to automate the generation of modeling information and procedures. At this stage, the research prototypes have demonstrated their potential for parametric modeling, yet for future validation, user studies will be conducted to provide further insight into the application of the proposed workflow in a parametric modeling process.

7. DISCUSSION

An essential component of the prototypes presented in this research is the artifact's representational characteristics as it facilitates users' interaction with digital models. Ishii (2008) explains that "In the design of TUI, it is important to give an appropriate form to each tangible tool and object so that the form will give an indication of the function available to the users" (p. xxii)

The artifacts used for each prototype provide a clear indication of the design element in use (panels, louvers, and blocks) and the way they should be interacted with to complete the modeling task. Ishii (2008) also mentions that:

> *This special-purpose-ness of TUIs can be a big disadvantage if users would like to apply it to a wide variety of applications since customized physical objects tailored to certain application cannot be reused for most other applications. By making the form of objects more abstract… you lose the legibility of tangible representation and the object will become a generic handle rather than the representation of underlying digital information. It is important to attain a balance between specific/concrete vs. generic/abstract to give a form to digital information and computational function. (p. xxii)*

Later prototypes do include a level of abstraction in the artifacts' representation; for example, the physical objects in Prototype 3 are shown as solid geometry (blocks)

without having any distinct architectural features. These blocks, however, are a direct physical translation of the CA cells and lattice, which are essentially abstract geometric representations. Prototype 5 also shows similar qualities to Prototype 3 as the design objects represent digital information (i.e., blocks as control points) instead of design elements. The artifact in these examples creates a balance between "specific" and "generic" representations, i.e., the blocks representing NURBS control points that are placed within a workbench representing an architectural space. Prototypes 3 and 5 provide a flexible modeling platform with generic features for a wide range of modeling applications when compared to Prototypes 1 and 2, with CA being a generic algorithm for creating pattern configurations, and NURBS being a generic mathematical representation.

Conclusively, the aesthetical and functional qualities of representations are essential for developing TUIs as they clarify the association between physical objects and digital information. The TUI loses its directness and intuitiveness if digital information is not given a proper physical form (Ishii, 2008). Design object's physicality can be taken advantage of to support decision making as it provides significant haptic feedback and insight into the mechanical behavior of parametric models. Kolarevic (2000) explains that Gehry's design practice, which has shaped the building and construction industry through digital production processes, starts his form finding process through physical models. These models are helpful for testing the constructability of sheet-material that will be used for the actual building (Pottmann, Asperl, Hofer, & Kilian, 2013).

7.1. Thematic Progress

The work involves developing digital-physical workflows that utilize custom-made haptic-based interactive systems for embedding design intents in digital models. The example prototypes focus on addressing the issues of utilizing mathematics and computer algorithms for establishing parametric frameworks. Throughout this study, prototypes were iterated to improve workflow functions for achieving higher levels of modeling task automation through implementing analog data interpretation methods (using regression models, generative algorithms, and transformation matrices). Figure 7.1 shows a diagram of the features included for each prototype, in addition to providing a visual illustration of the work's progress. The prototypes in the diagram are color-coded and chronologically arranged. The diagram also shows the prototypes connected to the tasks they are designed to complete from both categories located at the top and bottom. Example combines multiple features together to create a unique workflow to address the parametric modeling problems discussed in this research.

In the figure there are low- and high-level automated tasks, which are defined for as follows:

- *Low-level*: automation features enable the generation of single parameter inputs for completing specific modeling tasks, e.g., generating a mathematical equation for setting up a geometric constraint.
- *High-level*: automation features enable the generation of multiple inputs for creating parametric frameworks including modeling procedures and geometric operations.

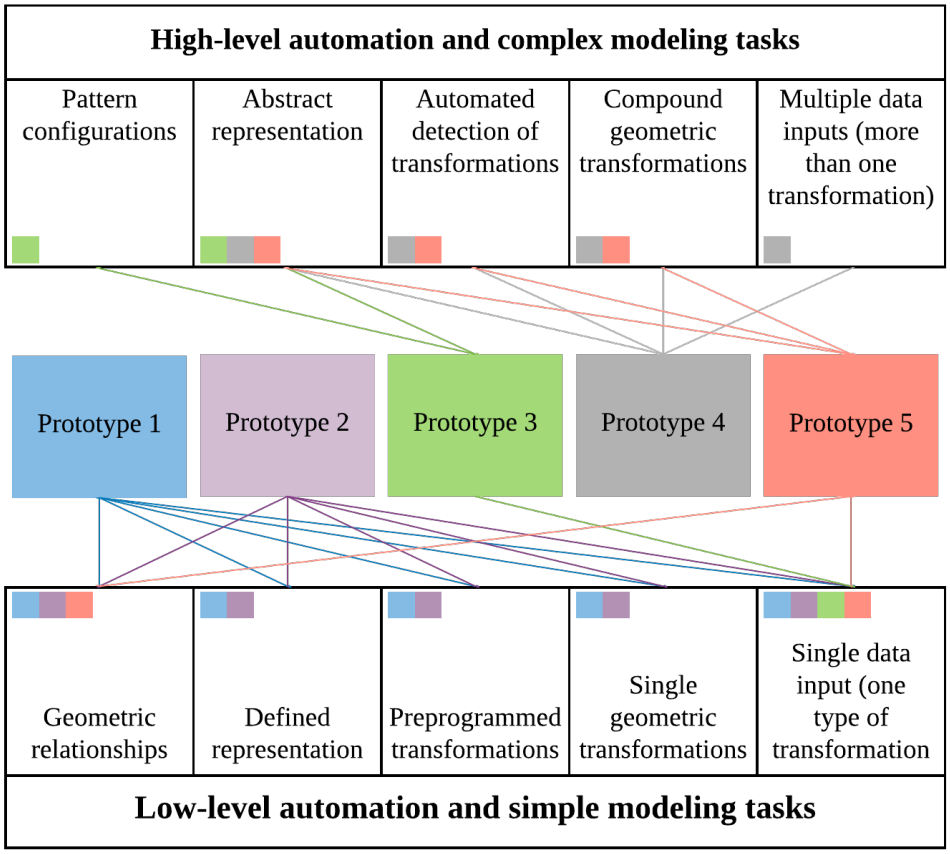

Figure 7.1 Diagram showing the progress of the work. Each prototype is designed to address a specific modeling problem. As they progress from 1 to 5, they include more sophisticated workflows that allow for higher levels of automation to assist in the modeling process.

7.1.1. Prototype 1 & 2

Prototype 1 and 2 are two examples for setting up algebraic constraints in digital models. Both examples include similar low-level automation features: geometric relationships, defined representations, preprogrammed transformations, single geometric transformations, and single data input. The single data input, which is generated using the workflow in both examples is a mathematical equation depicting physical object relationships; i.e., Prototype 1 a linear equation and Prototype 2 a polynomial equation.

The workflow utilizes tangible interaction and data mining (regression analysis models) to generate mathematical representations (equations and curves) for digitally constructing object relationships. The prototypes have been tested for creating simple and complex geometric relationships and for exploring curvilinear geometric configurations. The workflow demonstrates great value for parametric modeling as they release the designer from the burden of manually expressing implicit design knowledge into explicit mathematical functions.

7.1.1.1. Limitation of Prototype 1 & 2

The prototypes show a pre-structured setup using a single type of geometric transformation. The predefined sequence of geometric operation in the visual program do limit the prototypes' modeling potential and the flexibility to adjust it for other types of design scenarios. Prototype 1 which includes rotation, uses a rotary potentiometer in the artifact for monitoring the panels' angles of rotation and a Rotation parameter node in the visual programming graph to transform the panels' digital counterparts. This is done to maintain consistency between the inputs and outputs in the prototypes. The Rotation parameter inputs include 1) base geometry, the panels' configuration; 2) angles of rotation, using the rotary potentiometer sensors of Panel 1 and 3; 3) plane of rotation; 4) center of rotation, etc. that were set directly in Grasshopper except for the angles for rotating Panel 2, which were provided by the TUI. Prototype 2 includes a similar system setup, except for the Rotation node which is replaced by Translation node for moving the

louvers. The limitations found in these examples can be summarized in the following two points:

- A limited number of inputs, i.e., geometric transformation applied by physically manipulating the objects.
- Strict data flow, which is a common problem using history-based programming. Davis (2013) explains data and operation nodes must be arranged in a specific way to avoid any interruption in the program. In these current prototype examples, the designer must visualize the workflow and the expected results to construct the TUI accordingly.

7.1.2. Prototype 3

Unlike constraints, which associates geometric entities together by applying them one at a time, CA patterns establish more complex geometric configurations through creating a relationship between objects through a set of rules. Prototype 3 includes high-level automation features such as *pattern configuration* and *abstract representation*. However, some of its features does overlap the previous two prototypes such as having a *single input* (block location). The artifact's purely geometric representations offer a more flexible medium for digital modeling, as it can be interpreted in several ways (e.g., building components, spatial and urban layouts, etc.). Although the artifact includes a level of abstraction, it is a direct translation of the CA grid and cells, which can assist in making the connection between the different block layouts and number, and the generated rules in the digital model.

7.1.2.1. Limitation of Prototype 3

The artifact, the 9-square grid (i.e., the single cell neighborhood) in both the artifact and the digital model, can be used to explore pattern configurations. The visual program in Grasshopper includes two workflows; both tested using the same artifact, one for generating the seeds and the other for generating the rules. However, if the designer wants to simultaneously set up the seeds and rules in a single workflow, then a procedure must be written in the visual programming environment to enable this action.

7.1.3. Prototype 4 & 5

Prototypes 4 and 5 demonstrate significant improvements by automating essential modeling tasks. The improvements also include a new approach to creating artifacts that are more flexible for customization and operation. These two examples have similar features including *abstract representations, automatic detection of transformations,* and *multiple data inputs*. Additionally, Prototype 5 enables setting up geometric constraints, which was a feature in Prototype 1 and 2. The improvement in Prototype 5 is that it assists in creating a parametric framework as it automatically provides several inputs to the Grasshopper algorithm during the interaction process.

7.1.3.1. Improved Workflow & TUI

Prototype 4 has shown that artifacts linked to Transformation Matrices can provide a practical approach for making compound transformations and apply them simultaneously in parametric models. Non-compound transformations as explained in

Prototypes 1 and 2 are applied sequentially in the visual programming graph, which Hearn and Baker (1997) mention that it can be computationally expensive. If the order of these nodes in these examples is to be rearranged then the visual program, as Hoffmann and Joan-Arinyo (2002) mention, must be rebuilt to accommodate the changes. Stavric and Marina (2011) state that, it is difficult to make procedural changes to existing visual programs; a data flow must be defined for every design task (Sharp, 1992). However, using artifacts with transformation matrices in the TUIs helped in reducing the time spent creating parametric frameworks, more precisely it helped in avoiding the use of specific transformation nodes; like Move, Rotate, Scale, etc. explicitly in the visual program.

Prototype 5 extended the functionality of the workflow used in Prototype 4 to include 1) modeling NURBS objects, 2) setting up boundaries for them, and 3) generating design options. The digital model and its parametric functions are generated during the designer's interaction with the TUI. This attempt demonstrates an approach to creating a more flexible and generic system for parametric modeling using tangible interaction and NURBS curves.

The algorithms in the TUI Improvements section show the potential for a broader range of design scenarios. The parametric framework in previous examples do have a level of specificity; e.g., Prototype 1 establishes a relationship using linear equations. However, Prototype 4 demonstrates an algorithm for detecting physical interaction (mainly geometric transformation), and Prototype 5 demonstrates a workflow for modeling NURBS objects and setting up constraints. These workflows provide a

straightforward approach for designers to work with parametric models when compared to the conventional approach of only using text-based and graph-based programming tools.

Out-of-the-box sensors do restrict to some extent the design of the artifact; e.g., the artifacts in Prototypes 1 to 3 were designed according to sensor specifications. The use of conductive paint in the place of out-of-the-box sensors provided several advantages to the work such as 1) achieving a higher level of integration between the sensors and objects, which made it easier to customize the artifacts' design, including adding/removing objects on the workbench during operation; 2) sensing different and multiple types of transformations; and 3) cutting down the cost margin for constructing the TUIs for this research. Unlike conventional sensors, which were inserted in the workbench and were handled as separate entities, the conductive paint was applied (using a paintbrush) directly on to the objects to monitor their physical changes, which made the sensors as an integral part of the objects. Furthermore, conductive paint enabled creating multiple sensing mechanisms to simultaneously monitor the different types of object transformations, which was difficult to achieve in previous TUI models. As seen in Figure 5.24 in the Prototype Implementation chapter - Prototype 4, the panel and circuit were designed and assembled as a single unit.

7.1.3.2. Limitation of Prototype 4 & 5

Prototype 4 shows the transformation of a single object using translation and rotation, which may suggest limited use of the TUI. However, the algorithm in

Grasshopper is set up for detecting translation in all three axes and rotation transformations for multiple objects. Similarly, the blocks in Prototype 5 show a similar limitation as the blocks are spaced out at fixed intervals, and the objects can only be translated in one direction. The pre-structured setup for the TUI in these examples allows for controlling the number of inputs for testing the workflow, and the strictness of the TUI is not a reflection of the algorithm's capabilities. Other types of affine and non-affine transformation can be included in this workflow for manipulating digital models.

These prototypes demonstrated more sophisticated workflows than earlier examples, yet they do require some programming on behalf of the designer for implementation. Algorithmic procedures in these examples, such as analog data matching, do require designers to be acquainted with programming languages such as Python or other text-based scripting platforms. This might be challenging as some basic training is needed for managing data lists and writing conditional statements in the computer code.

7.2. Complexity

The design of the physical part of the TUI (artifact and physical computing setup) demonstrate a level of complexity in comparison to the modeling task it is intended to complete (e.g., creating a single type of object relationship or rules for CA). To set up a system with similar features requires both the knowledge and the skillset in parametric modeling, circuitry, and electronics. Nevertheless, the physical computing system in all five examples follows a similar setup, with small variations related to the

electronics to achieve the different modeling tasks. The design, fabrication, and assembly of the artifact require a similar set of physical modeling skills as in architecture, in addition to some knowledge of motion and behavioral dynamics to construct the interactive features of the artifact. Prototypes 4 and 5 show a more cost-effective and flexible approach for setting up the physical computing system using conductive paint for creating custom sensors. These sensors are easily integrated with the artifact and connected to the electrical circuit and the microcontroller board following the same procedure as in prototypes 1 to 3. More importantly, the prototypes as shown in Figure (7.1) progress from achieving single to multiple and simple to complex modeling tasks while the complexity of the physical system remains almost at the same level.

The algorithm differs from one example to the other, yet they all share similar logic, which can be further explained in Figure 7.2. The algorithm in these examples takes in raw sensor data and sends it to the data interpretation segment of the algorithm for generating the required information to set up the parametric model.

7.3. Framework

Figure 7.2 shows a framework of the system induced from the prototypes demonstrated in this research. The diagram explains the logic followed in this research for reproducing the TUIs. The grey components show the different physical states of a physical object. Raw analog data is transferred during interaction with a design object to the digital model. Physical properties may include geometric transformations, material

behavior under environmental conditions such as heat levels, light levels, etc. The blue components show the data interpretation process for generating digital information required for setting up a parametric model. The Orange components show the digital model, which can include geometric objects, or objects and their extended BIM database.

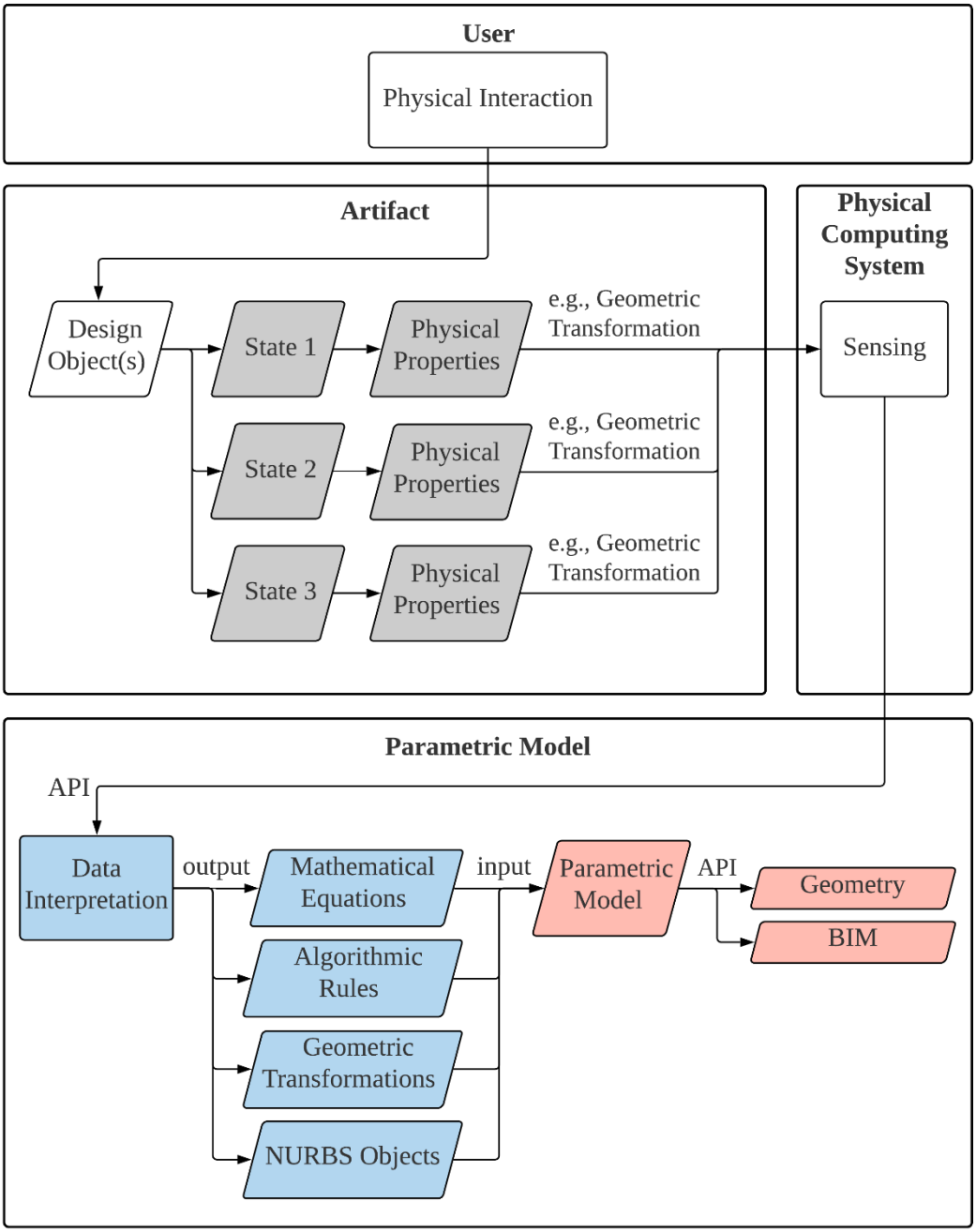

Figure 7.2 Diagram showing the induced general framework for establishing a TUI for generating mathematical and algorithmic information for setting up parametric models.

7.4. Prototype Applications

Aish (2005) mentions that "Our expectation is that geometric skills, compositional skills, and algorithmic skills will be the key to future design" (p. 12). For developing students' skills in these areas, it is essential to reformulate the framework of architectural design theory (Oxman & Gu, 2015) to develop a level of algorithmic thinking. For Karle and Kelly (2011) algorithmic thinking "is a way of relating tangible and intangible systems into a design proposal removed from digital tools specificity and establishes relationships between properties within a system" (p. 109). Karle and Kelly (2011) also suggest an educational structure that is parametrically driven by asking designers to focus on establishing rule-sets and associating variables to create a generative design instead of the conventional practice of designers starting by seeking the "right" design.

McNerney provides an overview of the development of tangible interfaces at MIT for supporting computing education (2004). Similarly, works by Horn and Jacob (2007), Zuckerman, Arida, and Renick (2005); Klemmer, Hartmann, and Takayama (2006) focus on teaching children computer programming concepts and other related skills through artifacts, which are not necessarily computationally enhanced such as having sensors or microcontrollers embedded in them. These examples may not relate directly to architecture or computational design in general, yet they demonstrate new and innovative ways of teaching algorithmic thinking.

Vermillion (2014) experiments with using artifacts to teach design students the fundamentals of L-Systems. The work initially started with integrating artifacts with

physical computing systems. However, students, later on, were asked to remove the physical computing system and focus on the high-level design goals of L-Systems instead of the low-level technical problems of setting up the circuits. Abell (2013) provides a similar approach where students combined traditional skills like handcrafting with parametric modeling. The objective was to find a balance between traditional and non-traditional design skills for educating students. The previous examples demonstrate some of the attempts some educators have made for teaching students some of the abstract concepts of computer programming. The combination of analog methods with digital tools has shown to promote a level of algorithmic thinking.

CA as a design method has been extensively explored in research (Cruz et al., 2016). The work by Herr (2008) provides a comprehensive overview of the applications of CA in architecture and urban design. CA includes unique features that make it accessible as a design tool, Frazer (1995) explains; CA is straightforward to implement in a design context because designers can use simple rules to produce complex geometric patterns rapidly. Prototype 3 contributes to these investigations by providing a tactile component to the algorithm. CA and TUIs can benefit digital design in several ways: the blocks can be seen as 1) architectural elements, or 2) entire buildings in an urban context. The former, designers can use the lattice as the layout of the building and the blocks as design elements or spaces. The latter, designers, can use each block as an entire building and the lattice as a neighborhood. Other possibilities for the TUI can be inspired by the work of Herr (2008), where CA is used to analyze urban growth or as a

tool for architectural space programming (i.e., a diagram for the layout of building facilities).

In Prototype 4, the TUI makes transformation matrices more accessible in a modeling process as it establishes a visual connection between physical object transformations and their mathematical representation using a matrix. This can be useful to educate students about geometric transformations and the benefits of using such a mathematical tool for computational modeling. Prototype 5 allows for constructing parametric models using a TUI. The designer can set up modeling constraints using the artifact and NURBS curves, which can make the transition of a design concept (represented in a physical model) into a parametric model much smoother. These constraints can be later used for further form experimentation and analysis.

7.5. Summary

This chapter explains the progress of the work that involves developing several methods for automating parametric modeling procedures using tangible interaction and provides a theoretical framework of the method. The artifact, as shown in this research, provide more than a simple control system for analog data input. The designer's interaction with the artifact and data interpretation methods implemented using visual programs create a workflow for automating modeling procedures, thus extending the capabilities of TUIs to benefit parametric modeling. The TUI models in this work are simple prototypes developed for research purposes. Nonetheless, they do have potential applications across the fields of mathematics, computer science, HCI, and architecture.

Prototypes 1 to 3 have shown to provide an approach to address some of the challenges of parametric modeling by generating some of the essential and difficult information for setting up a digital model. However, they do require extensive preparation including designing, fabricating, and assembling the artifacts; and creating the computer algorithm, which can seem like two distinct tasks. The systems' set up is usually influenced by the type of transformations the designer is going to apply to the objects, e.g., if the designer is going to rotate an object, then a sensor for rotation must be used in the TUI, and a rotation parameter node must be used in the visual program. The TUIs in these examples are linked to visual programs that were created for conventional modeling (i.e., a defined sequence of nodes). In this case, the TUI can be replaced by a keyboard or a mouse. This approach of creating the prototypes' components has shown to limit the potential of tangible interaction for parametric modeling. Prototypes 4 and 5 demonstrate a different approach, which reduces extensively the task of defining parametric frameworks and mainly focus on analog data interpretation. These workflows take advantage of the physical and digital components in such a way to complement each other and to create a more generic tangible medium for creating and operating parametric models.

8. CONCLUSION & FUTURE WORK

In this research, workflows using tangible interaction have been developed for addressing the problems of defining parametric frameworks using mathematics and computer programming. Five haptic-based prototypes were constructed to demonstrate these workflows. Each example included a unique data interpretation method for analyzing analog data to automate the process of generating parametric modeling information (mathematical equations and algorithmic rules) and for performing some modeling tasks (applying geometric transformations and creating NURBS objects). These prototypes were later evaluated by comparing them to conventional parametric modeling approaches to provide more insight into their benefits and drawback. This research has shown the plausibility of tangible interaction for parametric modeling and the potential uses and applications of such a digital-physical workflow in both academia and practice.

8.1. Testing & Evaluation

Each workflow was implemented using an architectural case study. These examples demonstrate a design scenario where a type of digital information (mathematical or algorithmic) is required in a design process for creating a parametric model. The results of each test assisted in evaluating the workflow and discussing the implementation of further improvements to establish a more sophisticated system to address the problem of parametrizing and representing a design intent digitally. The

prototypes were internally tested, and the results of the tests have shown that the proposed workflows work as expected. More precisely, the information generated using the prototypes were representative of the design objects' physical state (design intent) and produced parametric models where the digital objects' state was consistent with their physical counterpart.

Furthermore, a qualitative evaluation was also conducted comparing between the workflows developed in this research and the conventional modeling practice using text-based and/or graph-based programming methods. The evaluation focused on demonstrating the benefits and drawbacks of the workflows for addressing the modeling challenges of defining parametric frameworks. The analysis has shown that the TUI examples do require a level of programming knowledge and skills for constructing and integrating them in a design framework. Nevertheless, the TUIs can enhance the digital design process by capturing physical design intents and representing them as modeling information. For Algebraic Constraints (Prototype 1 and 2), a TUI can assist in representing a design intent mathematically and use the generated equations for setting up parametric constraints, which alleviates from the burden of manually constructing the equations and calculating their coefficients. For Algorithmic Rules (Prototype 3), the workflows provide a natural way of expressing algorithmic rules in a similar way to the written description of cell states (alive or dead) in Conway's Game of Life. The use of geometry provides a visual and tactile medium to create and manipulate the rules of the CA component in the visual program instead of providing numerical values representing the rules. For TUI Improvements (Prototype 4 and 5), this category demonstrates more

sophisticated workflows for data interpretation and adjustable artifacts. Prototype 4 allowed for applying compound and non-compound transformations to geometric objects (rotation and translation) using transformation matrices. The TUI was able to distinguish between both types of transformations when the physical object was manipulated. Applying these two types of transformations is a straightforward process using Graph-based programming. Nevertheless, the algorithm developed for this prototype can include other types of transformations and implement them in all three dimensions using the TUI. Prototype 5 was used for creating a NURBS curve. This type of curve requires several types of inputs to be constructed; this work focused on providing the number of control points and their location. The TUI's construction provided the flexibility needed to increase or decrease the accuracy of the NURBS curve by adding or removing more design objects in the artifact. Furthermore, boundaries were constructed for this curve for generating alternative configurations (i.e., design options). This work demonstrates a workflow for automating several modeling procedures using the TUI. Generally, such a process using a conventional workflow requires a level of understanding of programming procedures to create, parametrize, and generate interpolations using NURBS curves.

Testing results and the comparative evaluation demonstrate a proof-of-concept and the plausibility of the workflows for design practice and education. Further validation of the work through user studies is needed and is planned as future work. This research is in the early stages of development; establishing the workflows, experimenting with prototypes, and conducting the internal qualitative evaluation; and

for having user studies, there must be a well-defined framework of the system and thoroughly developed prototypes to be integrated into a design workflow. A description of a preliminary user study outline is provided in the Future Work section of this chapter.

8.2. Research Contribution

This work provides three categories of prototypes that are tested and evaluated for addressing the problems of defining parametric models using mathematics and computer programming. The result of developing the work has assisted in inducing a novel framework for using tangible interaction in the parametric modeling process, as shown in Figure 7.2 in the Discussion chapter. This framework provides a step-by-step description of the primary procedures for constructing the prototypes. In addition, the workflows (Figure 5.3, 5.8, 5.16, 5.19, 5.25, and 5.32) provide a detailed description of each of the steps in the framework for reproducing the work for each TUI example.

The Literature Review chapter has shown that TUIs are developed and used for a wide range of applications across the fields of art and science, as interactive instruments, educational tools, etc. TUIs are also increasingly integrated into digital workflows for design applications, and to name a few, as a platform for design collaboration, geometry manipulation, and environmental analysis and simulation. This research provides a framework for integrating tangible interaction and parametric modeling where physical design intents are captured into mathematical and algorithmic information and embedded in parametric models. This research explores tangible interaction and data

interpretation schemes to cover a range of digital modeling procedures and design tasks. To the knowledge of the author, this research contributes original work to the extensive body of research in the field of Computer-Aided Architectural Design or Design Computing.

8.3. Workflow Complexity & Generalizability

Figure 7.1 shows the progress of the workflows for performing modeling tasks, from single to multiple and simple to complex. Furthermore, the progress of the work also shows the development of the overall workflow from specific to generic. For example, prototypes 1 and 2 demonstrate a workflow for generating object relationships using linear and polynomial equations. Although these equations provide a range of possibilities for parametric modeling, the workflows are limited to these two types of relationships. Conversely, prototypes 3 to 5 demonstrate a more general approach for parametric modeling; Prototype 3 using a generative algorithm, which has been extensively researched as a design tool; Prototype 4 applying geometric transformations; and Prototype 5 creating NURBS objects, which are versatile modeling components.

8.4. Future Work

Future development of this research will focus on several areas including incorporating other types of Mixed Reality technologies, such as smart handheld devices and computer vision, in the workflows for creating a more practical user interface for interacting with parametric models. In addition, the current research in this document

suggests that the developed workflows can assist in design education for developing a level of algorithmic thinking. Such claims will be further validated by conducting user studies.

8.4.1. User Studies

Further development of the workflow will take place prior to performing user studies, such as:

- Developing a more flexible TUI, which will be used for Study 1 (below)
- Integrating more sophisticated algorithms for data interpretation such as machine learning (ML), which will be used for Study 2. Further description of ML is provided in section 8.2.2.

An initial outline of user studies is provided below, which will include two experiments, Study 1 and 2. Both studies will provide qualitative and quantitative data for validating the workflows. The studies will be conducted in an academic setting having architecture students as participants.

1. Study 1

This study focuses on students with no formal education in parametric design. The study will provide quantitative data for evaluating the workflow. The study will focus on the students' 1) speed, 2) efficiency, and 3) accuracy in completing design tasks using parametric modeling tools. The students will have a series of training sessions to familiarize them with digital tools and some of the basics in mathematics and computer

programming. Students are expected to learn how to operate digital tools using geometric operations to create simple parametric models.

This test will be conducted in two phases: Phase 1, creating a parametric model using algebraic constraints; and Phase 2, setting up generative algorithms. Each phase will have two sessions. For the first session, students will be asked to create a parametric model using computer input devices such as keyboards and mice; and for the second session, to repeat the same modeling process but with using TUIs. The test limits the students to a specific modeling scenario and using the corresponding prototype for completing the task.

2. Study 2

Like Study 1, students will be partaking in the experiment. This study will provide qualitative data as it mostly focuses on the participants' subjective experience working with GUIs and TUIs. They are given the freedom to create any geometric model of their choice, without having any restrictions, using their own acquired knowledge in parametric modeling. The experiment will be structured in a similar manner to Study 1, by having two sessions: one session using computer input devices and another using TUIs. Students are asked to provide their opinion regarding their modeling experience and which modeling tool did assist them in their creative process. Study 2 provides the opportunity to test more sophisticated workflows using ML.

8.4.2. Machine Learning

There is an increasing interest in the application of ML in the context of architecture. Future work will investigate the integration of tangible interaction with ML for capturing more complex types of design intents. Such an approach will provide designers with a sophisticated system that can automatically interpret physical interaction as modeling information without any extensive programming. Users interacting with the system can help build the knowledge base that would support the design process. In other words, a workflow combining tangible interaction and ML can learn designers' preferences through interaction and create an advanced system for human and machine collaboration. This ML component of the TUI will require big data that can be obtained from users of the TUI systems.

REFERENCES

Abell, J. H. (2013, April). *Hand Crafting computational design thinking in basic design studios*. Paper presented at the 1st aae conference, Nottingham, England. Retrieved from https://architecturaleducators.files.wordpress.com/2013/12/abell-2013-hand-crafting-computational-design-thinking-in-basic-design-studios.pdf

Aish, R. (2005). From intuition to precision. *Proceedings of the 23rd eCAADe conference*, 10-14. Retrieved from http://papers.cumincad.org/cgi-bin/works/paper/2005_010

Al-Qattan, E., Galanter, P., & Yan, W. (2016). Developing a tangible user interface for parametric and BIM applications using physical computing systems. *Proceedings of the 34th eCAADe conference, 2*, 621-630. Retrieved from http://papers.cumincad.org/cgi-bin/works/paper/ecaade2016_063

Al-Qattan, E., Yan, W., & Galanter, P. (2017a). Establishing parametric relationships for design objects through tangible interaction. *Proceedings of the 22nd CAADRIA conference,* 147-156. Retrieved from http://papers.cumincad.org/cgi-bin/works/paper/caadria2017_035

Al-Qattan, E., Yan, W., & Galanter, P. (2017b). Tangible computing for establishing generative algorithms – a case study with Cellular Automata. *Proceedings of the 35th eCAADe conference, 1*, 347-354. Retrieved from http://papers.cumincad.org/cgi-bin/works/paper/ecaade2017_057

Al-Qattan, E., & Yan, W. (2018, July). *Utilizing tangible computing for parametric modeling: Case studies for detecting types of geometric transformations and setting up constraints through tangible interaction*. Poster session presented at the 8th DCC conference, Lecco, Italy.

Appleby, D., & VandeKopple, J. J. (1997). *Programming languages: Paradigms and practices*, Mass: McGraw-Hill.

Araghi, S. K., & Stouffs, R. (2015). Exploring cellular automata for high density residential building form generation. *Automation in Construction, 49*,152-162. https://doi.org/10.1016/j.autcon.2014.10.007

Arduino [Microcontroller]. (2005). Retrieved from https://www.arduino.cc/

Aşut, S., & Meijer, W. (2016). FlexiMold: Teaching numeric control through a hybrid device. *Proceedings of the 34th eCAADe conference, 1,* 321-328. Retrieved from http://papers.cumincad.org/cgi-bin/works/paper/ecaade2016_077

Austin, M., & Qattan, W. (2016). 'I'm a visual thinker': Rethinking algorithmic education for architectural design. *Proceedings of the 21st CAADRIA conference,* 829-838. Retrieved from http://papers.cumincad.org/cgi-bin/works/paper/caadria2016_829

Bare Conductive. (2009). *Electric paint*. Retrieved from https://www.bareconductive.com

Beesley, P., Williamson, S., & Woodbury, R. (2006). Parametric modeling as a design representation in architecture: A process account. *Proceedings of CDEN*, 158-165. https://doi.org/10.24908/pceea.v0i0.3827

Burry, M. (2007). Innovative aspects of the Colònia Güell chapel project. In M. Burry (Ed.), *Gaudí Unseen: Completing the Sagrada Famíla* (pp. 59-61). Berlin, Germany: Jovis.

Burry, M. (1999). Paramorph: Anti-accident methodologies. In S. Perrella (Ed.), *Hypersurface Architecture II*. Chichester, England: Wiley.

Burry, M. (2011). *Scripting cultures: Architectural design and programming*. Chichester, England: Wiley.

Chen, X., & Hoffmann, C. M. (1995, December). Design compilation of feature-based and constraint-based CAD. *Proceedings of the SMA symposium,* 13-19. http://dx.doi.org/10.1145/218013.218021

Cuff, D. (1991). *Architecture: The story of practice*. Cambridge, Mass: MIT Press.

Cruz, C., Karakiewicz, J., & Kirley, M. (2016). Towards the implementation of a composite cellular automata model for the exploration of design space. *Proceedings of the 21st CAADRIA conference*, 187-196. Retrieved from http://papers.cumincad.org/cgi-bin/works/paper/caadria2016_187

Dassault Systèmes. (1995). SolidWorks [Computer software]. Retrieved from https://www.solidworks.com/

Davis, D. (2013). *Modelled on software engineering: Flexible parametric models in the practice of architecture* (Doctoral thesis, RMIT University, Melbourne, Australia). Retrieved from https://researchbank.rmit.edu.au/view/rmit:161769

Dourish, P. (2001). *Where the action Is: the foundations of embodied interaction*. Cambridge, Mass: MIT Press.

Dünser, A., Looser, J., Grasset, R., Seichter, H., & Billinghurst, M. (2010). Evaluation of tangible user interfaces for desktop AR. *Proceedings of ISUVR symposium*, 36-39. http://dx.doi.org/10.1109/ISUVR.2010.19

Eckerdal, A. (2009). *Novice programming students' learning of concepts and practise* (Doctoral thesis, Acta Universitatis Upsaliensis, Uppsala, Sweden).

Eden, H., Scharff, E., & Hornecker, E. (2002, June). Multilevel design and role play: Experiences in assessing support for neighborhood participation in design. *Proceedings of the 4th DIS conference,* 387-392. Retrieved from https://dl.acm.org/citation.cfm?doid=778712.778768

Eng, M., Camarata, K., Do, E. Y. L., & Gross, M. D. (2006). FlexM: designing a physical construction kit for 3d modeling. *International Journal of Architectural Computing, 4*(2), 27-47. https://doi.org/10.1260/1478-0771.4.2.27

Fischer, T. (2005). Teaching programming for and with microcontroller-enhanced physical models. *International Journal of Architectural Computing, 3*(1), 57-74. https://doi.org/10.1260/1478077053739603

Fitzmaurice, G. W. (1996). *Graspable user interfaces* (Doctoral thesis, University of Toronto, Toronto, Ontario, Canada). Retrieved from https://autodeskresearch.com/publications/georgephd

Fitzmaurice, G. W., Ishii, H., & Buxton, W. (1995). Bricks: Laying the foundations for graspable user interfaces. *Proceedings of the SIGCHI conference*, 442-449. http://dx.doi.org/10.1145/223904.223964

Frazer, J. (1995). *An evolutionary architecture.* London, England: Architectural Association.

Garber, R. (2014). *BIM design: Realising the creative potential of building information modeling*. Somerset, England: Wiley.

Gardner, M. (1970) Mathematical games: The fantastic combination of John Conway's new Solitaire game "life". *Scientific American, 223*(4), 120-123. Retrieved from http://www.jstor.org/stable/24927642

Gehry Technologies (n.d.). Digital Project [Computer software]. Retrieved from https://www.digitalproject3d.com/

GeoGebra (2018). GeoGebra math app. Retrieved from https://www.geogebra.org/

Gerber, D. J. (2007) *Parametric practices: Models for design exploration in architecture* (Doctoral dissertation). Available from WorldCat. Retrieved from https://tamulibraries.on.worldcat.org/oclc/145075459

Hearn, D., & Baker, M. P. (1997). *Computer graphics C version*. Upper Saddle, NJ: Prentice Hall.

Hernandez, C. R. B. (2006). Thinking parametric design: introducing parametric Gaudi. *Design Studies*, *27*(3), 309-324. https://doi.org/10.1016/j.destud.2005.11.006

Herr, C. M. (2008). *From form generators to automated diagrams: using Cellular automata to support architectural design* (Doctoral thesis, University of Hong Kong, Pokfulam, Hong Kong SAR, China). Retrieved from http://hdl.handle.net/10722/50272

Hoffmann, C. M. & Joan-Arinyo, R. (2002). Parametric Modeling. In G. Farin, J. Hoschek & M. Kim (Eds.), *Handbook of Computer aided geometric design* (pp. 519-541). North Holland, Netherlands: Elsevier.

Horn, M. S., & Jacob, R. J. (2007). Designing tangible programming languages for classroom use. *Proceedings of the 1st international TEI conference*, 159-162. http://dx.doi.org/10.1145/1226969.1227003

Hornecker, E. (2005). A design theme for tangible interaction: embodied facilitation. *Proceedings of the 9th ECSCW*, 23-43. https://doi.org/10.1007/1-4020-4023-7_2

Hornecker, E., & Buur, J. (2006). Getting a grip on tangible interaction: a framework on physical space and social interaction. *Proceedings of the CHI'6 conference*, 437-446. http://dx.doi.org/10.1145/1124772.1124838

Ishii, H. (2008). Tangible Bits: beyond pixels. *Proceedings of the 2nd international TEI conference*, xv-xxv. http://dx.doi.org/10.1145/1347390.1347392

Ishii, H. & Ullmer, B. (1997). *Tangible bits: towards seamless interfaces between people, bits and atoms*. *Proceedings of the SIGCHI conference*, 234-241. http://dx.doi.org/10.1145/258549.258715

Issa, R. (2013). *Essential mathematics for computational design*. Retrieved from https://www.rhino3d.com/download/rhino/5.0/essentialmathematicsthirdedition/

Jabi, W. (2013). *Parametric design for architecture*. London, England: Laurence King Publishing.

Jones, E., Oliphant, E., & Peterson, P. (2001). SciPy [Open source scientific tools for Python]. Retrieved from https://www.scipy.org/

Karle, D., & Kelly, B. M. (2011). Parametric thinking. *Proceedings of the ACADIA Regional conference,* 109-114. Retrieved from http://papers.cumincad.org/cgi-bin/works/paper/acadiaregional2011_012

Kensek, K. M. (2014). Integration of environmental sensors with BIM: Case studies using Arduino, Dynamo, and the Revit API. *Informes de la Construcción*, *66*(536). Rederived from http://informesdelaconstruccion.revistas.csic.es/index.php/informesdelaconstruccion/article/view/3575/4027

Kępczyńska-Walczak, A. (2014). The act of design – beyond the digital. *Architecturae et Artibus*, *6*(1), 24-28. Retrieved from http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-9d10115e-afa4-42ef-a7f9-a610782db4ef

Kim, M. J. & Maher, M. L. (2008a). The impact of tangible user interfaces on designers' spatial cognition. *Human-Computer Interaction, 23*(2), 101-137. https://doi.org/10.1080/07370020802016415

Kim, M. J. & Maher, M. L. (2008b). The impact of tangible user interfaces on spatial cognition during collaborative design. *Design Studies*, *29*(3), 222-253. https://doi.org/10.1016/j.destud.2007.12.006

Klemmer, S. R., Hartmann, B., & Takayama, L. (2006). How bodies matter: Themes for interaction design. *Proceedings of the 6th DIS conference*, 140-149. https://doi.org/10.1145/1142405.1142429

Kolarevic, B. (2000). Digital architectures. *Proceedings of the 22nd ACADIA conference*, 251-256. Retrieved from http://papers.cumincad.org/cgi-bin/works/paper/dcb9

Krawczyk, R. J. (2002). *Architectural interpretation of Cellular Automata*. Proceedings of GA, 7.1-7.8. Retrieved from https://www.generativeart.com/on/cic/papersGA2002/7.pdf

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin, 37*(3), 14-18. https://dl.acm.org/citation.cfm?doid=1151954.1067453

Lynxmotion. (n.d.). *Lynx B – pan and tilt kit*. Retrieved from http://www.lynxmotion.com/p-287-lynx-b-pan-and-tilt-kit-black-anodized.aspx

Ma, Y., & Harmon, S. W. (2009). A case study of design-based research for creating a vision prototype of a technology-based innovative learning environment. *Journal of Interactive Learning Research*, *20*(1), 75-93.

Maher, A. (2011). *Designing the design: establishing boundary conditions for designing parametrically – lessons from architectural practice* (Doctoral thesis, RMIT University, Melbourne, Australia). Retrieved from https://researchbank.rmit.edu.au/view/rmit:160202

Maher, M. L., & Kim, M. J. (2005). Do tangible user interfaces impact spatial cognition in collaborative design? *Proceedings of the international CDVE conference*, 30-41. https://doi.org/10.1007/11555223_4

Martin II, D. R. (2017). *Design intent in Cero parametric* [Kindle DX version]. Retrieved from https://www.amazon.com

McNerney, T. S. (2004). From turtles to tangible programming bricks: Explorations in physical language design. *Personal and Ubiquitous Computing*, *8*(1), 326-337. http://dx.doi.org/10.1007/s00779-004-0295-6

Milgram, P., & Kishino, F. (1994). A taxonomy of Mixed Reality visual displays. *IEICE Transactions on Information and Systems, 77*(12), 1321-1329. https://search.ieice.org/bin/summary.php?id=e77-d_12_1321

Mitchell, W. J. (1990). *The logic of architecture: Design, computation, and cognition*. Cambridge, Mass: MIT Press.

Monedero, J. (2000). Parametric design: A review and some experiences. *Automation in Construction*, *9*(4), 369-377. https://doi.org/10.1016/S0926-5805(99)00020-5

Morphocode (n.d.). Rabbit [Computer software]. Retrieved from https://morphocode.com/rabbit/

Myers, B. (1990). Taxonomies of visual programming and program visualization. *Journal of Visual Languages & Computing, 1*(1), 97-123.

Novak, M. (1998). Transarchitectures and hypersurfaces: operations of transmodernity. *Architectural Design*, (133), 84-94.

O'Sullivan, D., & Igoe, T. (2004). *Physical computing: sensing and controlling the physical world with computers*. Boston, Mass: Course Technology Press.

O'Sullivan, D., & Torrens, P. M. (2001) Cellular models of urban systems. In S., Bandini, T., Worsch (Eds.), *Theory and Practical Issues on Cellular Automata* (pp. 108-116). London, England: Springer. https://doi.org/10.1007/978-1-4471-0709-5_13

Otey, J., Company, P., Contero, M., & Camba, J. D. (2018). Revisiting design intent concept in the context of mechanical CAD education. *Computer-Aided Design and Applications*, *15*(1), 47-60. doi:10.1080/16864360.2017.1353733

Otto, F., & Rasch, B. (1996). *Finding form: Towards an architecture of the minimal*. Stuttgart, Germany: Edition Axel Menges.

Oxman, R., & Gu, N. (2015). *Theories and models of parametric design thinking*. *Proceedings of the 33rd eCAADe conference, 2,* 477-482. Retrieved from http://papers.cumincad.org/cgi-bin/works/paper/ecaade2015_33

Ozcan, O., & Akarum, L. (2001). Mathematics and design education. *Design Issues, 17*(3), 26-34.

Parallax Inc. (n.d.). PLX-DAQ [Computer software]. Retrieved from https://www.parallax.com/downloads/plx-daq

Payne, A., & Issa, R. (2014) *Grasshopper primer*. Retrieved from https://www.modelab.is/grasshopper-primer/

Payne, A., & Johnson, J. K. (2012). Firefly [Computer software]. Retrieved from http://www.fireflyexperiments.com/

Pottmann, H., Asperl, A., Hofer, M., & Kilian, A. (2013). *Architectural geometry*. Exton, PA: Bentley Institute Press.

PTC. (2011). Cero [Computer software]. Retrieved from https://www.ptc.com/en/products/cad/creo/

Razzaq, M. A., Qureshi, M. A., Memon, K. H., & Ullah, S. (2017). A survey on user interfaces with human and machines. *International Journal of Advanced Computer Science and Applications*, *8*(7), 462-467. https://pdfs.semanticscholar.org/a5ab/e99363013237cbc0cc278aa3724fe2924c9d.pdf

Reas, C., & Fry, B. (2001). Processing [Computer software]. Retrieved from https://processing.org/

Reeves, T. C. (2000). *Enhancing the worth of instructional technology research through "design experiments" and other development research strategies*. *International Perspectives on Instructional Technology Research for the 21st Century, 27*, 1-15. Retrieved from http://treeves.coe.uga.edu/AERA2000Reeves.pdf

Ruegg, C., Cuda, M., & Gael, J. V. (2002). Math.Net Numerics (V3.20.0) [algorithms for numerical computation]. Retrieved from https://numerics.mathdotnet.com

Rutten, D. (2015, December 17). Re: How to find mathematical functions of curves [Web log comment]. Retrieved from https://www.grasshopper3d.com/forum/topics/find-mathematical-function-of-curves

Rynne, A., & Gaughran, W. (2007). *Cognitive modeling strategies for optimum design intent in parametric modeling (PM)*. *Proceedings of the ASEE conference and exposition*, AC 2007-2132. Retrieved from http://icee.usm.edu/icee/conferences/asee2007/sessions/Wed.html

Salim, F. D., Mulder, H. M., & Burry, J. R. (2011). Form fostering: A novel design application for interacting with parametric models in the embodied virtuality. *Journal of Information Technology in Construction*, *16*(9), 135-150.

Salim, F. D., Mulder, H., Jaworski, P., Ransom, J. W., Cutellic, P., Iseki, T., … & Köm, Y. (2010). Collaborative design and live interaction with parametric models using UbiMash. *International Journal of Architectural Computing*, *3*(8), 377-398.

Sass, L. (2009). Parametric constructionist kits: physical design and delivery system for rapid prototyping devices. *International Journal of Architectural Computing*, *7*(4), 623-642. https://doi.org/10.1260/1478-0771.7.4.623

Schnabel, M. A. (2007). Parametric design in architecture. *Proceedings of CAADFutures,* 237-250. https://doi.org/10.1007/978-1-4020-6528-6_18

Sears, A. & Jacko, J. A. (2007). *The Human-Computer Interaction Handbook: Fundamentals, evolving technologies and emerging applications*. NY and London, England: Lawrence Erlbaum Associates.

Shaer, O., & Hornecker, E. (2010). Tangible user interfaces: past, present, and future directions. *Foundations and Trends in Human-Computer Interaction*, *3*(1-2), 4-137. http://dx.doi.org/10.1561/1100000026

Shaer, O., Leland, N., Calvillo-Gamez, E. H., & Jacob, R. J. K. (2004). The TAC paradigm: specifying tangible user interfaces. *Personal and Ubiquitous Computing, 8*(5), 359-369. Retrieved from https://dl.acm.org/citation.cfm?id=1023820

Sharp, J. A. (1992). *Data flow computing: theory and practice*. Norwood, NJ: Ablex Publishing Corporation.

Shelden, D. R. (2002). *Digital surface representation and the constructibility of Gehry's architecture* (Doctoral dissertation). Retrieved from https://dspace.mit.edu/handle/1721.1/16899

Smith, A. (2004). *Architectural model as machine: A new view of models from antiquity to the present day*. Amsterdam, Netherlands: Architectural Press.

Soler-Adillon, J. (n.d.). The Game of Life [computer code]. Retrieved from https://processing.org/examples/gameoflife.html

Stavric, M., & Marina, O. (2011). Parametric modeling for advanced architecture. *International journal of applied mathematics and informatics, 5*(1), 9-16.

Sutherland, I. E. (1963). *Sketchpad, a man-machine graphical communication system* (Doctoral dissertation). Retrieved from http://hdl.handle.net/1721.1/14979

Terzidis, K. (2006). *Algorithmic architecture*. Oxford, England: Architectural Press.

The Mathematics Behind NURBS. (2016). Retrieved 2018, from http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-84AD4364-96CA-48B5-BBBE-D0BF082CAB2F

Ullmer, B., & Ishii, H. (2000). Emerging frameworks for tangible user interfaces. *IBM Systems Journal*, *39*(3.4), 915-931.

Van Roy, P., & Haridi, S. (2004). *Concepts, techniques, and models of computer programming*. Cambridge, Mass: MIT Press.

Vermillion, J. (2014). Physical computing without the computing: small responsive prototypes. *Proceedings of the 18th SIGRADI conference, 1*, 643-646. Retrieved from http://papers.cumincad.org/data/works/att/sigradi2014_201.content.pdf

Weisberg, D. (2008). *The engineering design revolution: The people, companies and computer systems that changed forever the practice of engineering*. Retrieved from https://www.cadhistory.net/

Weiser, M. (2002) The computer of the 21st century. *IEEE pervasive computing*, *1*(1), 19-25.

Woodbury, R. (2010). *Elements of parametric design*. Routledge.

Yan, X., & Su, X. (2009). *Linear regression analysis: Theory and computing*. Hackensack, NJ: World Scientific.

Zaghloul, M. (2010). Mantis [Computer software]. Retrieved from http://zaghloul4d.blogspot.com/

Zuckerman, O., Arida, S., & Renick, M. (2005). Extending tangible interfaces for education: digital Montessori-inspired manipulatives. *Proceedings of the SIGCHI conference,* 859-868. https://doi.org/10.1145/1054972.1055093