Fabbed to Sense: Integrated Design of Geometry and Sensing Algorithms for Interactive Objects

by

Valkyrie Arline Savage

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Björn Hartmann, Chair
Professor Eric Paulos
Professor Carlo Séquin
Professor Paul Wright

Summer 2016

The dissertation of Valkyrie Arline Savage, titled Fabbed to Sense: Integrated Design of Geometry and Sensing Algorithms for Interactive Objects, is approved:

Chair   _____   Date   _____

             _____   Date   _____

             _____   Date   _____

             _____   Date   _____

University of California, Berkeley

Fabbed to Sense: Integrated Design of Geometry and Sensing Algorithms for Interactive Objects

Abstract

Fabbed to Sense: Integrated Design of Geometry and Sensing Algorithms for Interactive Objects

by

Valkyrie Arline Savage

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Björn Hartmann, Chair

Task-specific tangible input devices, like video game controllers, improve user speed and accuracy in input tasks compared to the more general-purpose touchscreen or mouse and keyboard. However, while modifying a graphical user interface (GUI) to accept mouse and keyboard inputs for new and specific tasks is relatively easy and requires only software knowledge, tangible input devices are challenging to prototype and build.

Rapid prototyping digital fabrication machines, such as vinyl cutters, laser cutters, and 3D printers, now permeate the design process for such devices. Using these tools, designers can realize a new tangible design faster than ever. In a typical design process, these machines are not used to create the interaction in these interactive product prototypes: they merely create the shell, case, or body, leaving the designer to, in an entirely separate process, assemble and program electronics for sensing a user's input. What are the most cost-effective, fast, and flexible ways of sensing rapid-prototyped input devices? In this dissertation, we investigate how 2D and 3D models for input devices can be automatically generated or modified in order to employ standard, off-the-shelf sensing techniques for adding interactivity to those objects: we call this "fabbing to sense."

We describe the capabilities of modern rapid prototyping machines, linking these abilities to potential sensing mechanisms when possible. We plunge more deeply into three examples of sensing/fabrication links: we build analysis and design tools that help users design, fabricate, assemble, and *use* input devices sensed through these links. First, we discuss Midas, a tool for building capacitive sensing interfaces on non-screen surfaces, like the back of a phone. Second, we describe Lamello, a technique that generates lasercut and 3D printed tine structures and simulates their vibrational frequencies for training-free audio sensing. Finally, we present Sauron, a tool that automatically modifies the interior of 3D input models to allow sensing via a single embedded camera. We demonstrate each technique's flexibility to be used for many types of input devices through a series of example objects.

Dedicated to the adventure-hungry:

whimsy, discovery, and enchantment are out there.

Contents

List of Figures

Acknowledgments

Many, many thanks are due the committee for their dedication and time in helping guide me through both the preparation of this document and the odyssey of graduate school itself. Special thanks are due, naturally, to Björn, for taking a chance on a grad student who clearly had no idea what she was doing. He has been endlessly patient in allowing me to explore new things and experiment with crazy ideas, as well as tolerating my occasional jaunts into absurd extracurriculars. It has been a wild ride, and you are a wonderful mentor!
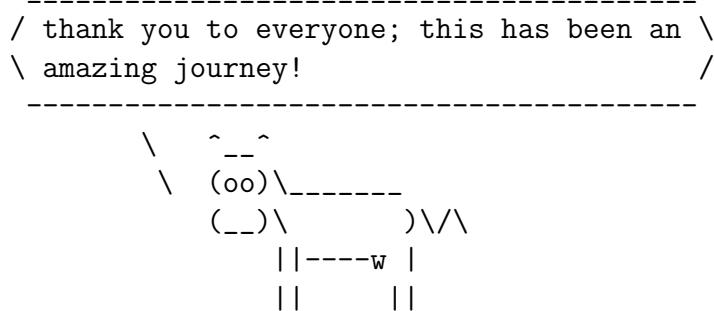
My collaborators at various times, including Berkeley grad students Andrew Head, Daniel Lim, and Michelle Nguyen, and undergraduates Xiaohan Zhang, Colin Chang, Jingyi Li, Eldon Schoop, and Mitchell Karchemsky; Autodesk researchers Ryan Schmidt, Tovi Grossman, and George Fitzmaurice; Adobe researchers Dan Goldman, Gautham Mysore, and Wilmot Li; and MIT-student-cum-Stanford-professor Sean Follmer; have aided me at every turn. Without them this work would naturally not have been possible.

There are even more of you that I never "officially" collaborated with, in the BiD lab and outside it, that gave me feedback and occasionally beer. Thanks to Shiry Ginosar, Celeste Roschuni, Mark Fuge, Lora Oehlberg, Wes Willett, Andy Carle, Drew Sabelhaus, Joe Blaylock, Anami Sheppard, Travis Brooks, Taska Sanford, Craig Lewin, Anand Varma, Luisa Beck, Tim Hunter, Jer Faludi, Armin Samii, Florian Altvater, Mark Oehlberg, Chris Myers, Drew Fisher, Peggy Chi, Laura Devendorf, Cesar Torres, Tim Campbell, Joanne Lo, and the EECS Baking Club for being awesome. I appreciate all your advice on ideas, implementation, and life.

I also owe a karmic debt to all Juggers, from the Berkeley Riot and elsewhere, with whom I have interacted over the past two years. Lacking a community with such a sense of play, I doubt I would have maintained my sanity long enough to complete this opus.

Finally, I thank my parents (Howard and Angela Savage), my sister (Venus Savage), and my husband (Evan Savage) for believing in me. I daren't try elaborating on all the amazing things you do, but thank you so much. ♡

Now, because this is my thesis and I can do what I want, here's a cowsay[1]:

```
 ------------------------------------------
/ thank you to everyone; this has been an \
\ amazing journey!                        /
 ------------------------------------------
        \   ^__^
         \  (oo)_____
            (__)\       )\/\
                ||----w |
                ||     ||
```

---

[1]https://en.wikipedia.org/wiki/Cowsay

Chapter 1

Introduction

Design is not just what it looks like and feels like. Design is how it works.

— Steve Jobs

Our environment is replete with products that have dedicated physical user interfaces like game controllers, musical instruments or personal medical devices. While the ubiquity of smart phones has led to a rise in touchscreen applications, retaining physicality has important benefits such as tactile feedback and high performance manipulation [54]. For example, gamers prefer physical input for speed and performance, musicians for virtuosity and control. Rapid additive manufacturing technologies enable designers and makers (henceforth we refer to both groups jointly as "makers" or "designers") to quickly turn CAD models of such future devices into tangible prototypes. While such printed form prototypes can convey the look and feel of a physical device, they are fundamentally passive in that they do not sense or respond to manipulation by a user. Building integrated prototypes that also exhibit interactive behavior requires adding electronic sensing components and circuitry to the mechanical design (see Figure 1.1).

Existing research has developed electronic toolkits that lower the threshold of making physical prototypes interactive [5, 34]. However, such toolkits still require makers to manually assemble printed parts and sensors. Such assembly may also require significant changes to a 3D model (e.g., to add fasteners or split an enclosure into two half shells). Detailed electro-mechanical co-design is time-consuming and cumbersome and mismatched with the spirit of rapid prototyping. Alternatively, makers may instrument their *environments* with sensors [4, 129], setting up specially-calibrated cameras and projectors to add interactivity, but these approaches limit interactive testing to the lab in small, restricted areas.

We aim to uncover the most cost-effective, fast, and flexible ways of sensing digitally-fabricated input devices. This suggests several requirements:

1. **cost-effective** : we substitute commodity sensors available in laptops and smartphones for custom electronic parts where possible. We focus on *single-sensor* techniques, where
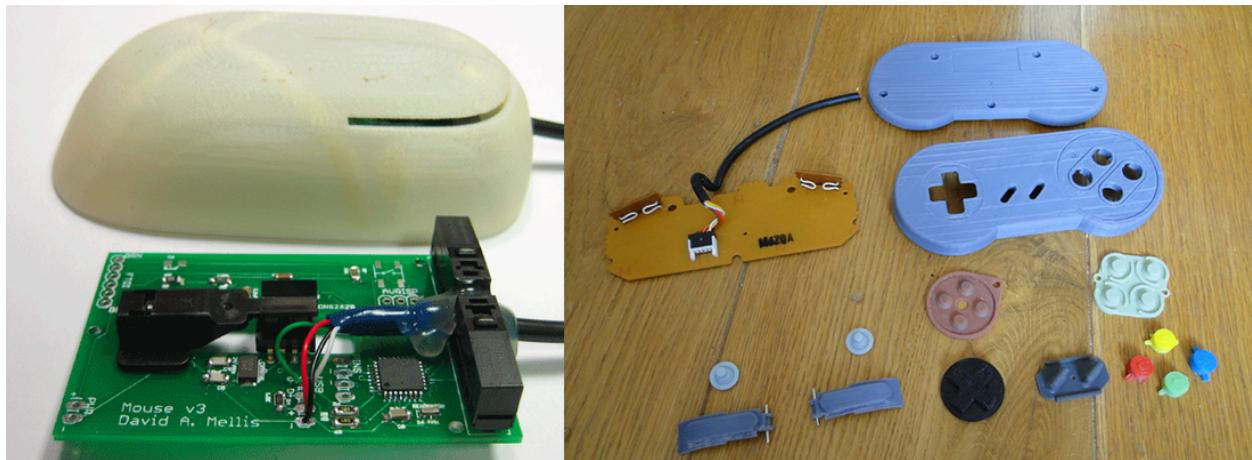
Figure 1.1: Makers use 3D printing to explore case form factors for interactive objects, like Dave Mellis's mouse (left, from `https://www.flickr.com/photos/mellis/5644225593/in/album-72157626129511674/`, ©①) or Thingiverse author srepmub's game controller (center, from `http://www.thingiverse.com/thing:337896`, by permission). Even professional designers use 3D printing for form-finding. However, this technique requires two separate processes: designing the case, then designing the functionality, typically in the form of circuitboards.

> sensing of multiple user inputs can be achieved by affixing a sensing apparatus to a single point, thereby reducing cost and assembly overhead.

2. **fast** : fabrication and assembly of senseable devices should not take significantly longer than comparable passive devices. Necessary digital modifications should be performed automatically when they involve complex or global changes, or be reduced to templates that the user may drag in when simple.

3. **flexible** : the means of sensing a prototype object should not impose undue burden on the physical designs of that object. Sensing techniques should accommodate a wide variety of input types (e.g., buttons, sliders, and dials) and body types (e.g., convex, concave, 3D).

We propose a novel way of ensuring these properties: users create digital design files, which our tools modify automatically based on knowledge of the sensing technique that will ultimately be used. Users then fabricate their modified models using digital fabrication machines. Because the physical models are precisely fabricated based on the digital design files, this process creates a *link between the digital and physical models*. Post-fabrication, we leverage this link to inform sensor processing: that is, we feed digital model data to our sensing algorithm, from which the algorithm extracts information about dimensions and input components, predicts information about properties, or calculates other relevant details. This allows us to *avoid training* and/or *improve sensing* for the interactive prototype.

1.1 Contributions

This thesis explores the realm of physically prototyping tangible input devices using digital fabrication machines, pushing towards a world in which prototyping physical interactive devices is as easy as prototyping GUI devices is today. We have built several prototype design and sensing systems designed to test different parts of the design space. Thus, this thesis makes the following contributions:

1. Fabbing to sense: a model and sensing co-design technique which uses knowledge of a particular sensing paradigm to automatically modify digital design files before fabrication, allowing improved or training-free sensing of the fabricated prototype. We offer three exemplars of this technique: Midas, Lamello, and Sauron.

2. Midas, a method for automatically generating custom capacitive touch sensors—cut from adhesive-backed conductive foil—by synthesizing sensor pads and routing connections from a high-level graphical specification. We also demonstrate a *design tool* using this method to enable users to fabricate, program, and share touch-sensitive prototypes. Using our tool, we describe an evaluation demonstrating Midas's expressivity and utility to designers

3. Lamello, a technique using passive plastic tine structures, 3D printed at interaction points and with predictable vibrational frequencies, to create passive tangible inputs sensed via audio. We describe a *design pipeline* which predicts tine frequencies (and an evaluation that they can be accurately predicted) and senses user manipulation of components in real time. We also include a discussion of information encoding techniques useful for this technique, and a series of scripts to generate parts utilizing these encodings.

4. Sauron, a design tool enabling users to rapidly turn 3D models of input devices into interactive 3D printed prototypes where a single camera senses input. We detail our method for tracking human input on physical components using a single camera placed inside a hollow object, and two algorithms for analyzing and modifying a 3D model's internal geometry to increase the range of manipulations that can be detected by a single camera. Finally, we describe an informal evaluation of our *implementation* of these techniques usable on models constructed in a professional CAD tool.

1.2 Dissertation Outline

This section presents a brief outline of the structure of this dissertation by chapters.

This dissertation first investigates the capabilities of modern digital fabrication machines and discusses sensing techniques compatible with those capabilities. Second, we lay out a solid foundation of related work. We then discuss three instances of analysis and design tools that help users design, fabricate, assemble, and *use* input devices sensed in a variety of

ways; for each technique we demonstrate its flexibility for use in many types of input devices. Finally, we conclude with suggestions for future work.

Fabrication & Sensing (Chapter 2)

This chapter explores modern digital fabrication machines—3D printers, laser cutters, and CNC mills, among others—and the properties that can be employed in objects they fabricate. We break down these properties, specifying which are inherent to materials, which are inherent to particular production processes, and which machines can currently realize objects with which properties.

Chapter 2 further discusses sensing techniques compatible with those capabilities, as well as what makes each technique promising for further investigation as a technique for sensing fabricated input device prototypes. Again, we focus on techniques which require a single sensing apparatus attached to a single point on an object. For example, we discuss the potential combination of 3D printed conductive metal with Hall effect sensors; running a current through the object could generate a magnetic field detectable by the sensors. This thesis ultimately selects a few points to further examine in this space, described in Chapters 4-6.

Related Work (Chapter 3)

We lay out the existing research landscape, describing which parts may have been overlooked and explaining which areas we explore in this thesis. In general, we draw heavily on work from four major traditions: simulation, sensing digitally-fabricated devices, modeling 3D objects, and creating prototypes.

Simulation

Pre-fabrication simulation was one of the first explorations fabrication research. This kind of pre-processing can, for example, account for deformities from specific printing processes [48], or ensure occlusion-free toolpaths [36]. More recently, computer graphics researchers have leveraged pre-print simulation of multi-material printers to control post-print deformation behaviors [13] and appearance [56]. We, too, perform pre-print simulation and optimization of 3D models, but for the purpose of creating *interactive* objects.

3D CAD Tools

Many professional tools for 3D modeling exist [2, 94], and they serve their target users well. Researchers have made significant strides in inventing new styles of more accessible interactions for 3D modeling, for example by capturing users' hand-carving processes and converting them to toolpaths [127] or scanning, augmenting, and reproducing clay models [102]. For the purposes of our investigations in this thesis, we created design tools to help

author objects compatible with our sensing techniques: they are a complement to, rather than a replacement for, existing CAD tools research.

Sensing Digitally-Fabricated Devices

Sensing forms a key component of interactive devices, and thus has been significantly explored in the past. One common technique for sensing objects uses machine learning and guided manual training to detect interactions using sound [82, 57], capacitance [89] or other signals. Our techniques focus on sensing done without machine learning, and often without any training at all. Inspired by "Sensing through Structure" [110], we design objects whose properties we know, which can inform how we sense them.

Prototyping Tools

Abundant research has examined questions around the types of prototypes that designers build in the course of designing a new object [47]. Other work looks further into more facile ways of creating functional electronic devices, for example using snap-together circuits [63, 42, 120] or smart circuit substrates [119]. One important limitation of these investigations is that they are limited to a constrained library of manufactured components: designers must make do with what they can buy. We conversely focus on customizable inputs that can be configured exactly as a designer wants them, and these customizations can be performed *in software* on a digital model. Once an object is assembled, its functionality must be defined. We leverage techniques like programming by demonstration (PBD) [75, 42] to help users define their objects' interactivity, however our focus is on design of the objects themselves.

Midas: Capacitive Sensing of Custom 2D Layouts (Chapter 4)

Our first exploration examines fabrication of 2D conductive materials sensed capacitively. Midas explores how to prototype touch-based interactions where input and output are *not* co-located, as they are on touch screens. Designers are given a drag-and-drop authoring system to create capacitive touchpads on the surface of objects, and from these designs generates 2D design files for fabrication. These files can then be cut from a conductive material and sensed using an automatically-configured microcontroller board. Midas also offers support for programming the input devices designed via PBD and websockets.

The advantages of capacitive sensing in this manner are numerous. It can be deployed on any flat, singly curved, or developable object's surface (see Figure 1.2). The sensors are cheap and easy to fabricate—whether on a vinyl cutter, using a circuitboard mill, or in inkjet printed conductive ink. Sensor assembly is *fast*: users simply need to attach the sensor dongle's wires to their fabricated sensor pads (e.g., using Z-axis transfer tape).

In this chapter, we will detail Midas's implementation, as well as discussing possible uses, drawbacks of the current system, and work for the future.

Figure 1.2: A Midas-generated interface with buttons for checking email mounted on the back of a smart phone.

Lamello: Acoustic Sensing of 2D/3D Mechanisms (Chapter 5)

Second, this thesis dives into an examination of acoustic-based sensing for devices fabricated in 2D, 3D, or a combination. The Lamello project investigates the use of tine-like structures for repeatable and predictable audio frequency generation. These tines can be printed at interaction points (e.g., under the path of a human input slider) such that they are struck when a user manipulates input components. The mechanical vibrations created by striking the tines can be detected with a contact microphone and classified using frequency analysis (see Figure 1.3).

Leveraging uniform 2D lasercut or 3D printed plastics as sound-creating input devices offers flexibility to designers with different levels of access to fabrication machines. In addition, beyond Midas's offering of flat input surfaces activated by a simple touch, Lamello explores input mechanisms that users can push, slide, and turn. The technique of passive audio generation for sensing also opens up opportunities in the future Internet of Things: multiple unpowered Lamello-type input devices may be placed in the environment and sensed by a
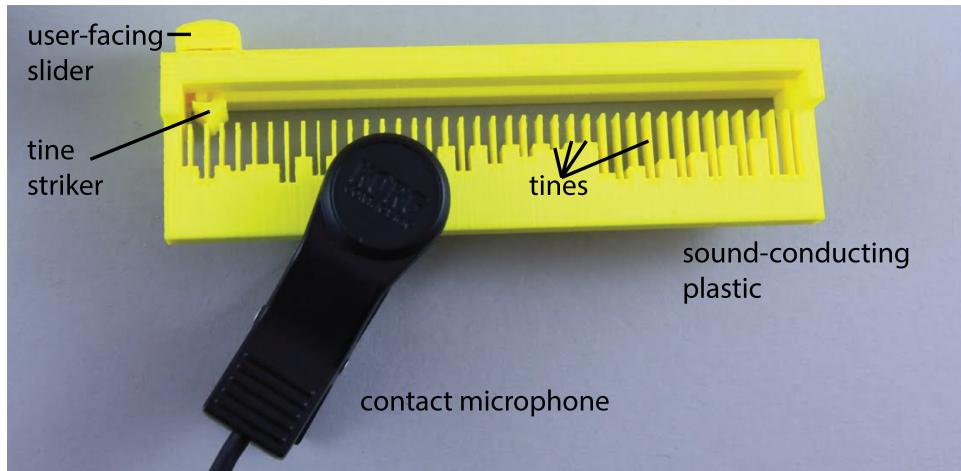
Figure 1.3: A Lamello-based interface with a series of plastic tines on a slider, which can be classified by the attached contact microphone.

single microphone, perhaps located on a laptop or smartphone.

This chapter details our experiments confirming that our 3D-printed tine structures behave in predictable ways in spite of the non-uniform nature of the materials that comprise them. We also discuss, using several exemplars, techniques for integrating the tines into existing input component designs. Further, we describe information encoding principles for tine generation.

Sauron: Vision-Based Sensing of 3D Printed Mechanisms (Chapter 6)

Finally, we explore full 3D input devices sensed using computer vision. Sauron is a design and sensing toolkit for creating 3D printed input devices—which can include components like joysticks or dials—sensed with a single embedded camera. The Sauron tool makes automatic modifications to allow for this sensing, reconfiguring the *interior* parts of the inputs and performing interference simulation (see Figure 1.4).

Sauron's interfaces have additional flexibility over those for Midas or Lamello: they allow *continuous* sensing of user input. Sliders need not be composed of individual capacitive sensors or a series of tines, but any arbitrary position along the track may be sensed. They can be fabricated on any 3D printer which can generate support material, and the prefabrication simulation process relies only on geometry rather than any particular materials properties for its processing.

In this chapter we describe our implementation of Sauron as a plugin for a commercial CAD tool, as well as the vision sensing code we built. Finally, we elaborate on limitations of the current system and places we may improve it, as well as future work in the area.
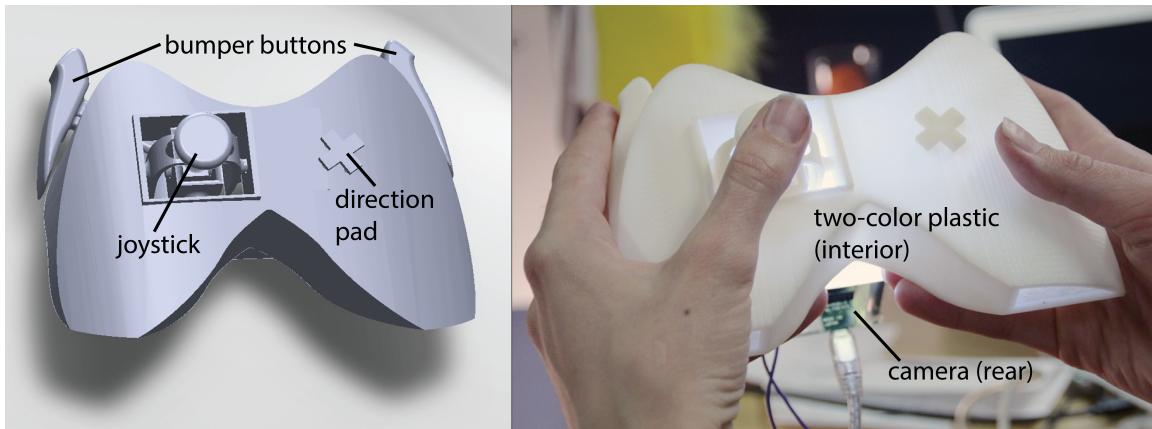
Figure 1.4: A Sauron-optimized game controller with a joystick, a direction pad, and several buttons.

In this chapter we describe our implementation of Sauron as a plugin for a commercial CAD tool, as well as the vision sensing code we built. Finally, we elaborate on limitations of the current system and places we may improve it, as well as future work in the area.

Conclusion & Future Work (Chapter 7)

The final piece of this thesis reviews the contributions described and re-evaluates assumptions made in the projects constituting its main chapters. Namely, our projects leverage a *single fabrication machine* for creating *one prototype* at a time, which is *hand-optimized* by a designer and sensed by a *single* sensor. Re-evaluating these leads to interesting pointers for future work in ecologies of multiple fabrication machines, branching prototypes, machine-optimized prototype designs, and usage of combination sensors that can still be mounted at a single point.

1.3 Statement of Multiple Authorship and Prior Publication

The research presented in this dissertation was not undertaken by me alone. While I initiated and led all projects described herein, I must acknowledge the contributions of my talented group of collaborators: without their efforts, this research could not have been realized in its current scope.

In particular, Midas's routing features were implemented by Xiaohan Zhang, and the video was created by Lora Oehlberg. Andrew Head performed much debugging and audio testing on the Lamello project, and that project benefited from the wisdom of my collaborators Dan Goldman (who provided the initial idea), Gautham Mysore, and Wilmot Li

at Adobe. Sauron's computer vision was implemented by Colin Chang, and many thanks are due Mark Oehlberg for assisting in the creation of the necessary circuitboards for that project.

My advisor, Björn Hartmann, provided invaluable advice and guidance on all projects detailed in this document.

This dissertation is partially based on papers previously published in ACM conference proceedings; I am the primary author on all publications. In particular, Midas was published at UIST 2012 [99]; Lamello at CHI 2014 [101]; and Sauron at UIST 2013 [98].

So, let's do this thing.

Chapter 2

Fabrication and Sensing of Input Devices

In addition to the profound repercussions these technologies will likely have on the manufacturing industry, the democratization they enable promises to unleash creativity and innovation at a level comparable to those brought about by the personal computer and the internet.

— Catarina Mota, *The Rise of Personal Fabrication* [70]

Our goal is to enable simple construction of input devices. The route that we choose to take for this is digital fabrication, as it allows us to create prototypes whose properties we can modify, predict, and thereby sense. As a framework to consider how this will work, we lay out the puzzle in five pieces:

1. User action

2. Transformative mechanism

3. Material characteristic

4. Senseable change

5. Sensor selection

Each input device designed under our paradigm takes a user action and transforms it through some type of mechanism. The mechanism is constructed from materials with various properties, and the combination of materials properties, user interaction, and transformative mechanism lead to a senseable change; all that's left is to select a sensor that can detect it. This flow relates to the concept that input devices are like onions, with many transformative layers between a user's action and the use of the sensed data (see Figure 2.1).

Thanks to digital fabrication, we can use foreknowledge of the mechanism to be fabricated along with its predicted properties to make sensing easier. For example, perhaps we desire to
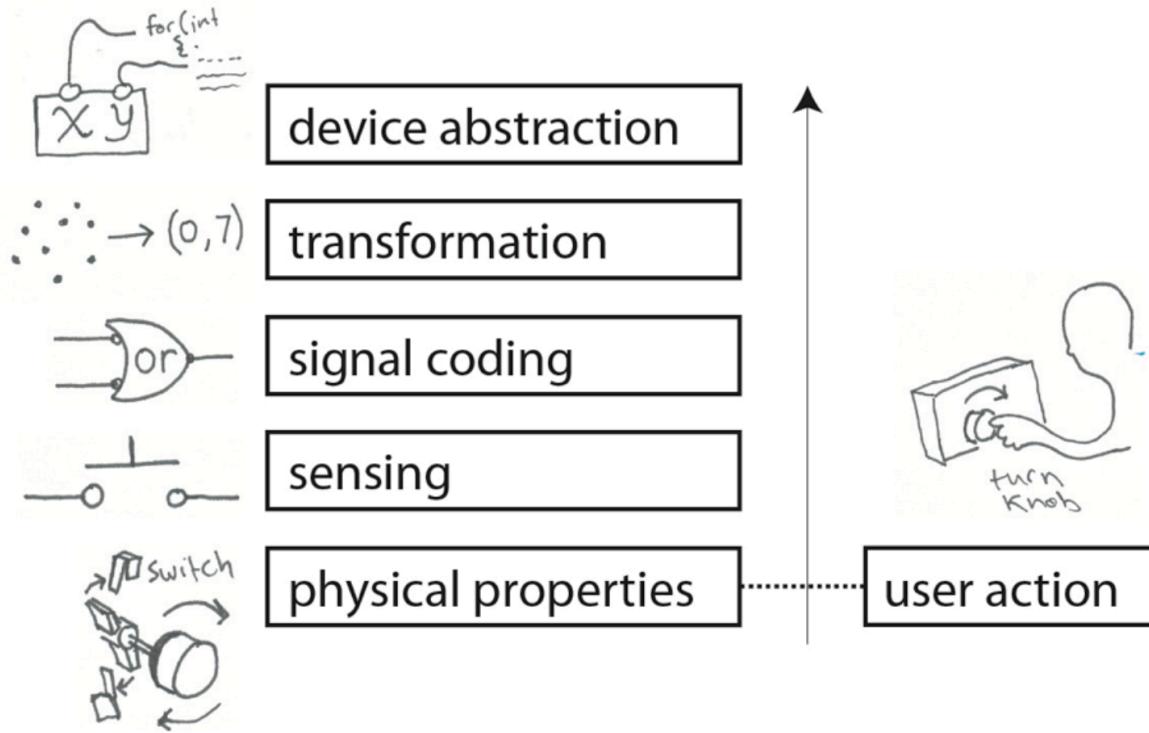
Figure 2.1: Input devices are like onions, with many layers of processing between the user's action and actual interface code (from [29], by permission).

sense a user's physical *action* (e.g., pushing, sliding, or turning). Some processes can create tine structures (cantilevered beams); we place these tines under a printed mechanism that *transforms* a user's action into tine plucks. When struck, these tines vibrate at frequencies determined by their geometry and *material*; the knowledge of these two things and the fact that our chosen material is rigid and sound-conducting means that we have a senseable *change* in the form of vibration (sound). We can *sense* the tines' vibrations with a microphone, and decode the signal for use in our application. We need not train the algorithm beyond giving it the digital design file—we can predict the tines' vibrational frequency from that and our knowledge of our selected material's properties.

2.1 User Actions and Transformative Mechanisms

Stu Card, et al., describe the design space of input devices using movement operators (linear/rotary, absolute/relative, movement/force) and composition operators (merge, layout, connect); they represent each input device as a tuple detailing user manipulation, the input space, the device's current state, the resolution or mapping function from input space to

output space, the output space, and "additional aspects of how a device works" [20]. Since our focus is on the *construction* of such input devices, we refer the interested reader to Card, et al.,'s description of user inputs and possible transformative mechanisms therefor. By way of example, some common user inputs that we may wish to track include touching, pushing, squeezing, sliding, or twisting. Useful transformative mechanisms include the 3D printed spring-based components used for Sauron and the tine strike transformation used for Lamello. Other transformative mechanisms may include converting a strong puff of air into a rotating motion using a turbine, or using magnets, conductors, and a shaking motion to create current.

This chapter will describe the spaces of materials properties (and how they relate to modern digital fabrication machines) and sensors. Further, we will discuss potential "links" between these, as in the tine example above.

2.2   Definitions

First, we briefly define words and machines that will be discussed in this chapter and the remainder of the thesis.

additive fabrication In additive fabrication, material is deposited and a shape is built up.

subtractive fabrication A subtractive fabrication process removes material to create a form. Excess material may be reused in another project or discarded.

3D printer A 3D printer is one of a class of machines that additively create a three-dimenstional model from one or more materials.

FFF FFF (fused-filament fabrication) 3D printers lay down material by melting and depositing a filament in a precise pattern.

model material Model material is the substrate that composes the final object.

support material Many modern 3D printers are capable of laying two types of materials, model material and a secondary, sacrificial material that can support overhangs in the model during printing, then be removed.

SLA SLA (stereolithography) printers use a bath of UV-curable polymer and a controllable UV laser. The laser "draws" each layer on the polymer, causing photopolymerization where it strikes. Excess material is simply poured out for reuse.

SLS SLS (selective laser sintering) 3D printers contain a bed of material (e.g., metal powder) which is compacted and formed into a solid mass of material by heat and/or pressure without melting to the point of liquefaction. Excess material can be brushed off and reused.

binder jetting Binder jetting is a powder-based printing technique similar to SLS, but instead of melting a powder together to create layers this method uses inkjet heads that drip a binder (e.g., epoxy) to adhere the powder particles. Unbound powder can be brushed off and reused.

PolyJet PolyJet printers have print heads similar to those of inkjet printers which sweep across the build area depositing material. Following the printer head is a UV light, which cures deposited material droplets.

vinyl cutter A vinyl cutter subtractively processes 2D materials with a 2-axis knife blade, cutting patterns into them. Vinyl cutters are typically used for thin, flexible materials.

laser cutter A laser cutter guides a laser's output over a 2D domain for processing flat materials. Laser cutters can cut or engrave into materials, and are often used for rigid materials $< \frac{1}{4}$ inch thick. Some have rotary attachments for engraving on circular surfaces like the outside of a glass.

CNC router A CNC router uses a 3-axis rotary mill to cut through thick, rigid materials, like wood or certain metals. Some CNC routers are portable and can attach to many materials, while some are stationary with beds into which material is loaded.

CNC mill A CNC mill is a multi-axis machine which subtractively creates a 3D shape from a block of material, usually metal or wood.

These definitions are duplicated for convenience in Appendix B.

2.3   Material Characteristics (and Digital Fabrication)

Digital fabrication machines are those which can take as input a digital design file, in 2D, 2.5D, or 3D, and output a physical realization of that design. A design created in a computer-aided design (CAD) tool is processed by a computer-aided manufacturing (CAM) tool to create machine instructions to generate the object. This workflow stands in contrast to traditional crafting techniques (which do not require machine code) as well as traditional manufacturing techniques (which require "tooling" for each design created). The true power of digital fabrication lies in its ability to create unique objects on each machine run *without* the extensive setup and tooling necessary to change the product created by, for example, an injection moulding machine. This comes with the blessing and curse that each instance of an object costs as much to manufacture as the one before it, but allows for variations between instances without additional cost [135]. For example, even makers can now create custom 3D printed prosthetics that fit particular bodies, e.g., a new hand for a young girl who wants to play on the playground [113].

The joint interests of industry, academia, and hobbyist makers have led to a flourishing ecosystem of digital fabrication and rapid prototyping (RP) machines. These machines

| | | material | | |
	plastic	metal	wood	paper	plaster
transparent	✖				
translucent	✖		✖		
opaque	✖	✖	✖	✖	✖
color	✖	✖		✖	✖
no color	✖	✖	✖	✖	✖
rigid	✖	✖	✖		✖
flexible	✖	✖		✖	
isolator	✖		✖	✖	✖
resistor		✖			
conductor		✖			

Figure 2.2: The most common materials processed by digital fabrication machines, including plastic and wood, have wildly different realizable characteristics.

describe a continuum from simple vinyl cutters that can subtractively create 2-dimensional stickers to sophisticated multi-material 3D printers that can create multicolor and conductive designs where the designer has full 3D control over the geometry of the object. These machines allow their manufactured products to achieve various material and structural properties. We examine materials properties of common digital fabrication inputs (see Figure 2.2), as well as the machines that can process them and the compositional properties they make possible (see Figure 2.3).

Properties

Fabricated objects may afford certain types of manipulations and interactions due to their particular combinations of material properties. We discuss several common property classes: appearance, rigidity, and conductivity.

Appearance

Materials may have a wide variety of appearance characteristics. For example, plastics can be transparent, translucent, or opaque. Paper can have color patterns or be plain white. Appearance characteristics affect the way light travels through an object: the object may reflect the light, scatter it, or allow it to pass through. By matching an object's index of refraction to the fluid surrounding it, the object can even appear to disappear.

Rigidity, Hardness, and Strength

A material's rigidity or hardness affords particular kinds of actions: in particular bending (also known as flexure) can be afforded by either an object's geometry or its material properties. Geometrically, a long thin object can often can be bent whether it is flexible or rigid. However, most objects fabricated from flexible materials can be bent, while more rigid ones typically do not encourage bending actions. Hardness, which describes a material's resistance to permanent shape change when compressed, can be measured using Shore hardness.

Strength is related to rigidity and hardness, but rather than being a property of a material it is typically a property of a structure.

Conductivity

Electrical properties may allow a material to be used as a part of a circuit. Metal is the only commonly fabricated material (among wood, plastic, paper, and metal, see below) that is conductive, but material mixing (e.g., plastic filament with embedded graphite, or silver ink printed on paper's surface) can lead to conductive properties in materials that lack them.

Materials

The types of materials that can be processed using the various additive and subtractive RP machines is broad, and different materials may be more easily formed by different machines. We consider the four main types of materials used in digital fabrication for the purposes of this thesis: wood, plastic, paper, plaster, and metal. We also discuss common properties, as above, that they may exhibit and how these can affect options for fabricating them.

Plastic

Hard plastics—mainly thermoplastics like acrylonitrile butadiene styrene (more commonly known as ABS) and polylactic acid (known as PLA)—are the materials *du jour* in the maker community. They come in the form of heatable, extrudable filaments for FFF machines (e.g., Makerbot[1], Printrbot[2], RepRap[3], uPrint[4]). Hard plastics can also be additively processed with other 3D printing techniques, for example photocure plastics in PolyJet machines (ABS-like and Vero series for Objet machines[5]) and SLA machines (methacrylates in Form1[6]), or thermoset plastics like epoxy in research systems (e.g., Harvard's system [25]).

---

[1]Sold by Makerbot Industries, `http://makerbot.com`.

[2]Sold by Printrbot, `http://printrbot.com`.

[3]An open-source printing project tracked at `http://reprap.org`.

[4]Sold by Stratasys, `http://www.stratasys.com/3d-printers/idea-series/uprint-se`.

[5]Sold by Stratasys, `http://www.stratasys.com/3d-printers/design-series/objet260-connex3`

[6]Sold by Formlabas, `http://formlabs.com/products/3d-printers/form-1-plus/`.

Hard plastics can also be subtractively processed through milling and laser cutting, although many plastics are unsafe for laser vaporization (e.g., ABS plastic emits chlorine gas when lasercut). One great thing to do with hard plastics is combine them with soft foams and fabricate spars for jugger (you'll need to look this up yourself; adding a footnote here is too risky and my committee may notice it).

Flexible plastics are less common than hard ones, though there are some notable materials here: thermoplastic polyurethane (sold as Ninjaflex) is a filament-style flexible material for use in FFF machines, and some PolyJet machines likewise have support for flexible plastics (e.g., Tango series for Objet).

Subtractive processing for flexible plastics is possible, though may be more challenging due to different shear parameters than stiffer materials. PVC plastic in the form of vinyl sheets can be cut to shape with a vinyl cutter.

Transparent plastics are not yet available for most maker-class machines, though many SLA-processed resins are optically translucent and PolyJet machines offer optically transparent plastics (e.g., VeroClear for Objet). Laser cutters are suitable for processing sheets of transparent acrylic, as well.

Metal

Conductive metal is relatively simple to process subtractively (e.g., milling circuitboards on a CNC router), even by vinyl cutters which can cut thin metal foils. Certain conductive metals, e.g., steel, can be directly laser-sintered in a high-heat process. Conductive-impregnated filaments for FFF machines do exist in limited use, but they are typically based on graphene (carbon) rather than metals. The new Voxel8 FFF 3D printer [121] uses a silver-based material for conductive purposes.

Inert metals are processed in essentially the same ways as conductive metals, although they have not been engineered for use in FFF filaments.

Wood

Wood-type materials are most commonly subtractively processed (e.g., by CNC routing): this allows for extensive choices available in terms of wood hardness, colors, and even blends. Newer processes are available to add wood in the form of sawdust to FFF-compatible filaments (e.g., woodfill by colorfabb), and also to laser sinter with wood chips [3].

Paper

Paper is trivially subtractively processed. None of the major 3D printing methods can use paper-based materials, though one subtractive/additive method (called Selective Deposition Lamination, a subtype of Laminated Object Manufacturing) uses stacks of cut paper to create 3D paper models (e.g., IRIS[7]).

---

[7]Sold by MCor Technologies, `http://mcortechnologies.com/3d-printers/iris/`.

Plaster

Plaster (and its cousin, concrete) tend to be more rarely used than plastic, wood, paper, and metal, but they are gaining traction as researchers and engineers investigate the use of 3D printing for constructing large-scale structures like buildings. Plaster is typically used as a powder for binder jetting, as in ZCorp printers[8], and can be colored by varying the color of the binder. On the other hand, concrete is typically fabricated by FFF printers: they extrude the concrete in its liquid state.

Machine Abilities

The varying constructions and methods of additive and subtractive fabrication machines allow them to process particular materials in particular ways. We briefly describe geometric and compositional possibilities for a variety of digital fabrication techniques.

Material Composition

Additive digital fabrication methods offer significant freedom in terms of assembly methods. While subtractive machines are typically limited to the original composition of the material (material is loaded in as a solid block and parts are removed from it), additive fabrication methods can create objects which are built of solid, sparse, or hollow material (see Figure 2.4). These material compositions give rise to opportunities to design acoustics, airflow, or other wave/fluid systems; additionally they give a designer fine-grained control over object strength and weight.

Additive machines also permit deviation from a material's original characteristics in their ability to blend and use multiple types of stock material. This allows a PolyJet machine to take as input black material and white material, and create as output either an object with both black and white material, or an object exhibiting a variety of shades of grey. Blending and the use of multiple materials allows for control of appearance, rigidity, and conductivity, as described above.

Geometry Fabrication

Digital fabrication machines can support any complexity of geometry, from 2D images on paper (as an inkjet printer produces) to 3D projections of 4D objects (like Shapeways's Klein bottles printed in steel) (see Figure 2.5). We describe the possibilities for the various geometries, as well as machines that could produce them. Note that we list machines at the edge of their range: for example, a CNC mill (listed under 3D external) can also make 2.5D or 2D objects.

---

[8]ZCorp is now owned by 3D Systems, `www.zcorp.com/es/Products/3D-Printers/ZPrinter-650/spage.aspx`

	subtractive					additive				
	Paper Cutting	Vinyl Cutting	Laser Cutting	CNC Mill	CNC Routing	Fused-Filament Fabrication	Stereolithography	Selective Laser Sintering	Binder Jetting	PolyJetting
materials										
plastic (hard)			✗	✗	✗	✗	✗	✗	✗	✗
plastic (flexible)	✗	✗	✗	✗	✗	✗		✗		✗
plastic (transparent)			✗	✗	✗		✗			✗
metal (block)			✗	✗	✗	✗		✗		
metal (foil)	✗	✗	✗	✗						
wood (block)			✗	✗	✗	✗				
wood (sheet)			✗	✗	✗					
paper (sheet)	✗		✗							
plaster/concrete						✗			✗	
composition										
solid	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
sparse						✗				
hollow				✗		✗	✗	✗	✗	✗
multiple materials						✗				✗
blending materials										✗
geometry										
2D	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
2.5D			✗	✗	✗	✗	✗	✗	✗	✗
3D (partial)			✗	✗		✗	✗	✗	✗	✗
3D (full internal)						✗	✗	✗	✗	✗

Figure 2.3: Additive and subtractive RP machines each have their own sweet spots of operation. Subtractive tools can process more types of materials, but additive ones offer more compositional and geometric flexibility.

2D geometry 2D geometry lies flat on a surface, but can manifest as an image printed on paper, a sticker cut from vinyl, or a barcode engraved on granite. Machines that support 2D geometry fabrication include vinyl and paper cutters, as well as inkjet printers.

2.5D geometry: A slight jump from 2D is 2.5D: a 2D shape with additional *depth* information. The canonical machines to create 2.5D objects are 3-axis CNC routers, which mill away material from the surface and can take multiple passes for deeper features. Similarly, laser cutters can engrave 2.5D geometry by varying laser power or conducting multiple passes.

3D external geometry: Some machines are capable of creating arbitrary 3D external geometry, including overhangs, without manual reorientation of a part. This can be
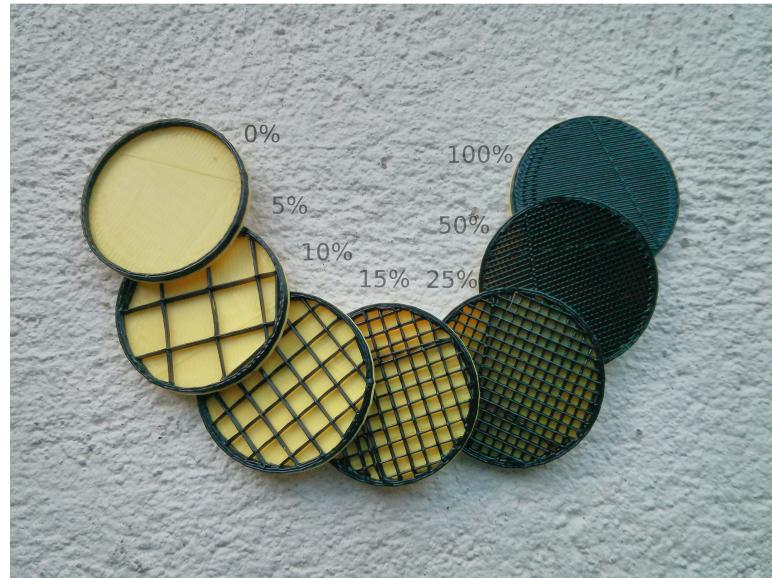
Figure 2.4: FFF machines can lay plastic to create a variety of material compositions, ranging from hollow to sparse to solid. Here, we see a variety of compositions, arranged in order of percent(material/air) (Thingiverse user CreativeTools, from `http://www.thingiverse.com/thing:85711`, ⓒ①)
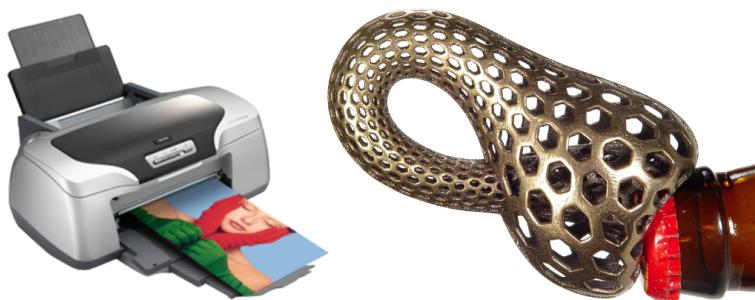
Figure 2.5: The range of objects produceable via digital fabrication is huge: from 2D images on paper (left, David Mellis on Flickr, from `https://www.flickr.com/photos/mellis/5298037920/`, ⓒ①), to 3D projections of 4D objects (right, Bathsheba Grossman of Bathsheba Sculpture LLC, from `http://bathsheba.com/math/klein/klein_Abeer.jpg`, by permission).

accomplished by machines that use both a model material and a sacrificial support material: combining the two materials permits overhangs that may not be possible using only a single material. However, removal of this support material can be challenging, depending on the process used. (See [100] for a more complete treatment of support material and removal techniques.) 3D printers can produce overhang geometry by laying support material as a part of each 2D object slice, and 5-axis CNC mills can produce overhangs via automatic object reorientation. Features like overhangs can be challenging to mill using a 2.5D machine, as subtractive techniques require that the machine tool has unobstructed access to the surface to be milled. Creating such features on a 2.5D machine may require reorienting an object several times.

3D internal geometry Full control over internal geometry can only be executed via additive manufacturing, and it allows for designed cavities, mechanisms, textures, and more on the interior of objects manufactured in a single pass. Again, this is typically executed by machines that lay sacrificial support material: this does impose some design constraints, as voids with support material inside must be accessible in order to remove it.

2.4   Single-Sensor Sensing Techniques

One key research area in Human Computer Interaction is designing new techniques and algorithms to help computers accept human input. Thus, while many of the input techniques here could be used in, for example, machine-to-machine communication, we describe how a person's actions might create a usable control signal.

Single-sensor Motivation

Why use a single sensor? To reduce time spent on each iteration of a prototype performing assembly and calibration, we aim to allow users to fabricate an object and snap on a single sensing module. In the future, it would be interesting to explore multi-sensor modules (e.g., modern smartphones have magnetometers, capacitive screens, microphones, cameras, and more), however as initial work we are exploring one sensor at a time.

Sensor Types

A "single sensor" can take many forms, ranging from a humble switch which opens and closes to a high-speed video camera which captures 2D visual information at 1000Hz to an accelerometer measuring G-forces in 3 directions.

As they represent a vast array of actions sensed, physical phenomenon, and other aspects, there are multiple ways of organizing sensors for discussion purposes. We use the hierarchy proposed by the Modern Sensor Handbook [30]: it arranges sensors by changes sensed. This

aligns neatly with our framework for creating prototypes, as a user's action will create a change, which we then can detect using the appropriate variety of sensor.

The major categories in the Modern Sensor Handbook are

- occupancy and motion

- position, displacement and level

- velocity and acceleration

- force, strain and tactile

- pressure

- flow

- acoustic

- humidity/moisture

- light detectors

- radiation detectors

- temperature

- chemical sensors

Curious readers can learn more of their workings from the Handbook [30]; we take their triggers and high-level functions to be self-evident here.

2.5 Promising Overlaps

The spaces of materials, properties, geometries, and sensors are vast: we lay out a framework to consider combinations of them for fabricating senseable objects.

Principles

Several principles guide our selections. Namely,

- a user's manipulating the object should generate a control signal senseable using the given sensor. For example, bending a flexible material generates a signal detectable by a displacement sensor. This may require usage of transformative mechanisms, as described above.
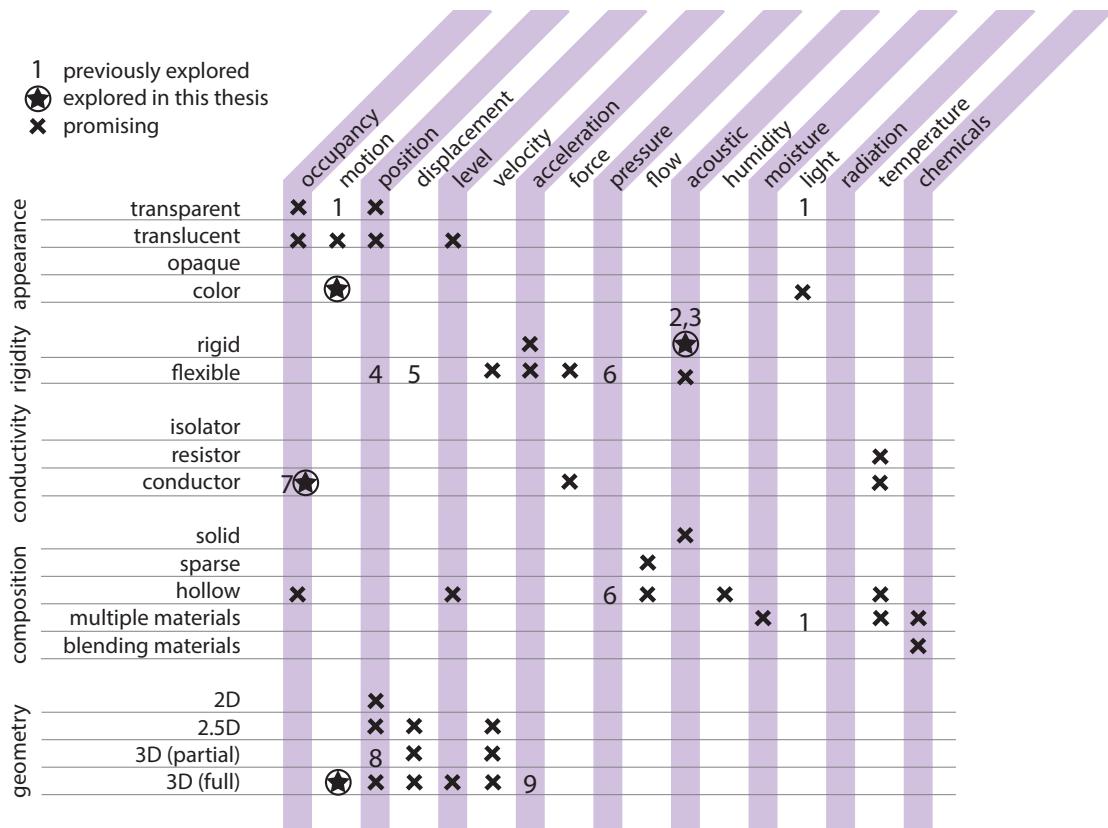
Figure 2.6: Lining up the possibilities of fabricatable properties with sensor types, we can see several areas that are promising for further exploration. We also identify pairings that have been explored previously, as well as pairings explored in this thesis. 1. Printed Optics [128], 2. Acoustruments [57], 3. Stane [74], 4. Flexibend [24], 5. Dynamic latex buttons [37], 6. Pressure-sensing robot skins [109], 7. Capricate [105], 8. Design by physical composition [26], 9. Making 3D printed objects interactive with wireless accelerometers [46].

- the material can conduct or predictably insulate the control signal: for example, rigid materials conduct sound waves and can be used with microphone sensing, but flexible/soft materials could be used to insulate sound and direct it through an object.

- geometry should work with the sensing abilities of a given technology. For example, an object with a solid core would not admit fluids for flow sensing; 3D internal geometry or a hollow object would make this type of sensing possible.

- material composition may pair with material properties to get additional sensing. For example, *blending* stiff and flexible plastics may allow for more unique degrees of sensing with a flex sensor than flexible materials alone.

x marks the spot: these principles give rise to the **x**s marked in Figure 2.6.

Overlaps Discussed in this Thesis

Conductive Metal + 2D Geometry + Capacitive Sensor

Chapter 4 describes Midas, a technique for creating custom capacitive touchpads CNC cut from adhesive copper foil. These can be affixed to everyday objects and sensed using a single capacitive touch controller.

Hard Plastic + 2D/3D Geometry + Microphone

Chapter 5 describes Lamello, a technique for creating tines from hard plastic; when struck each tine vibrates at a characteristic frequency which can be classified using a microphone.

Multi-color + 3D Internal Geometry + Camera

Chapter 6 describes Sauron, a technique leveraging colored arbitrary internal geometries to create mechanical input devices that can be sensed by a single embedded camera.

Other Promising Overlaps

While clearly several types of overlaps have been extensively explored, we suggest a couple very promising unexplored areas here.

One such promising idea uses force sensors with flexible materials. As a user manipulates an input, the stretch, compression, and torque forces they generate disperse through an object. A force sensor located near the center of all inputs should be able to detect these transmitted actions. This may be easiest with polyjetted 3D printed objects, as their materials properties are the most consistent (and therefore easiest to model) due to their fine resolution.

Prototype objects could be designed with translucent internal piping and gate-like structures at interaction points: the pipes, when filled with liquid, could be amenable to sensing

with a camera or level sensor. This could be thought of as a liquid-state flute. Additionally, the orientation of such an object in space could be calculated using liquid levels, as they would settle based on gravitational forces.

Users' bodies maintain a higher temperature than typical room temperatures. This fact could be used in an interactive object which has conductive pathways at interaction points or grip locations, leading to a temperature sensor. This type of sensing would be slower than simple capacitance sensing, but could provide additional passive cues like overall body temperature or time spent in a particular configuration. To properly sense, such an algorithm might require knowledge of the interacting human's temperature in addition to digital geometry knowledge.

2.6 Conclusion

We have laid out the current state of digital fabrication, as well as listed material and geometric properties achievable using the various methods currently available. Further, we described ways of combining these properties with sensors that could detect user input in order to create human input devices. Now, we will move on to discussing related work, then our design tests one at a time.

Chapter 3

Related Work

If I have seen further, it is only by standing upon the shoulders of giants.

— Isaac Newton

Situating this thesis in the realm of prior work, we draw heavily on work from four major traditions: simulation, sensing, 3D modeling, and prototyping tools.

3.1   Simulation

The simulation community was one of the first to rally around digital fabrication: they have delved extensively into digital simulation of machine toolpaths, materials and behaviors, and the advent of hyper-precise 3D printers like the Objet Connex line[1] allowed for a variety of real-world manifestations of these simulations.

Early Simulation : Structures and Manufacturability

Mathematical simulation for structural analysis existed before digital fabrication, e.g. [28], and supported CNC machining. Some of the very earliest work on fabrication-related simulation focused on optimizing toolpaths, for example to ensure good coverage from robotic spray heads without occlusion from model features [36]. Other early work examined and accounted for deformities in models produced using different 3D printing technologies [16, 48], or ensuring printability of 3D models [10, 15].

Materials and Behaviors

Early simulations paved the way for more reliable machines and quality outputs. With these basic properties nailed down, more recent visual simulation (i.e., graphics) research

---

[1]At time of publication, the Connex3 allowed for XY resoltion of 600dpi, Z resolution of 1600dpi, and accuracy of 20-85 microns. `http://www.stratasys.com/~/media/Main/Files/Machine_Spec_Sheets/PSS_PJ_Connex3.ashx`

has explored pre-fabrication simulation techniques that affect the post-fabrication material characteristics of models.

Among visible characteristics, work has been done on subsurface scattering [43] and subsurface reflectance [125]; these techniques require using multi-material machines which can blend transparent and opaque materials. Such precision fabrication machines can also allow for creating tiny structural elements, like honeycombs, that can contribute to an object's overall appearance [56]. Another system can take motion capture sequences and translate them into fabricatable mechanical automata that perform the same motions, thanks to careful simulation of linkages [21].

For tactile, audio, and tangible characteristics, Bickel, et al., model and fabricate particular deformation behaviors using optimized blends of rigid and soft materials [13]. Higher-level optimization strategies can yield models with voids inside to ensure balance after fabrication [90], or musical instruments which have specific pitches after fabrication [117]. Other systems can infer the joints in digital character models, and add in printable, poseable joints which users can manipulate post-print [9, 19].

Fabrication Itself

The actual fabrication process can also be improved with recent work in graphics. For example, linking knowledge of the layered manufacturing technique employed by FFF machines (described in Chapter 2) to a 3D model can help ensure that it will not break under force [118], or that fabrication will produce minimal waste [104]. Closed-loop computer vision systems integrated with 3D printers can also improve fabrication accuracy [108].

This thesis will draw on the same theme of pre-fabrication simulation of materials and models. Certain types of sensing necessitate guaranteeing that a prototype will have specific material properties, however our focus does not end with the fabricated object itself. We use information from the model to inform our sensing techniques and to develop interactive prototypes.

3.2   3D CAD Tools

As digital fabrication machines become more accessible, 3D CAD tools must go through a similar transformation to make the machines actually useful in the hands of makers and other non-professionals. Powerful industrial modeling tools, like SolidWorks [2], AutoCAD [7], ProEngineer [91], and Rhino [94], have existed for years and served industrial designers and professional engineers very well. Makers find lower-threshold, lower-ceiling tools like Meshmixer [103] suitable for their needs. Research has investigated additional ways of modeling objects.

Domain-Specific Constraints

A few research projects have investigated domain-specific constraints into 3D modeling tools. In general, this allows for smart feedback to users, as well as pre-fabrication simulations. For example, the Plushie system [69] allows users to design 3D fabricatable plush toys using only 2D sketches: the system has a basis for the sewing necessary to create a plush toy, and can infer the third dimension. It even presents a preview to users as they author. SketchChair also allows users to create 3D objects from 2D sketches using such constraints: in this case, users sketch chairs in profile [97]. This system allows users to simulate bodies of various sizes and weights sitting in the chairs to ensure comfort and to ensure the chairs will not tip over. Our research uses pre-fabrication simulations, but our goal is sensing rather than just ensuring physical properties of the final objects.

Physical Authoring of Digital Models

Many researchers have explored physical CAD tools, as this can ideally help those without expertise in using CAD to create or modify objects. Such systems use the link between the physical and digital models in one way or another. Some of this work is done using proxy objects, for example by tracking Duplo blocks stacked by the user [35], capturing annotations on paper versions of models [112], or scanning sculpted clay models marked with stickers [102]. Other research explores actually having a designer work with a fabrication tool directly, capturing their toolpath and creating a model and/or final artifact by automatically tidying it up [127, 72, 71]. Conversely, the fabrication tools can *direct* a toolpath using information from a 3D model and enable a dialogue between user, tool, and model [139]. We similarly link object geometry to its physical form, however we pursue this link for sensing purposes rather than for form finding.

Integrating Electronics and Existing Objects

Techniques for creating interactive objects have been widely examined in HCI: allowing designers to integrate electronics hardware or other existing objects into fabricatable designs is a common approach. Sensing modules can be the primary driver of an object's form factor, for example by allowing designers to drag the modules around digitally and re-forming a shape that will hold them all [122, 120]. This technique could be useful as an add-on for our tools, but we focus on sensing rather than authoring 3D case forms. Our other work uses 3D conductive paths for joining sensors to microcontroller boards, which allows for additional flexibility in design [100], however this requires that the designer has intimate knowledge of the sensing techniques to be used. Tools in this thesis build in knowledge of the sensing paradigm, relieving that burden from designers and allowing additional redesign flexibility versus one-off, hand-created sensor routings [78, 85]. Creating an object that fits with existing hardware has been explored using Kinect-based scans of objects *in situ* [68, 123] or photographs [58], or intelligent capture of object dimensions with measuring tapes

[60, 124]. Our focus is on integrating sensing into a designer's vision for an object: this is a complementary tactic.

3.3   Sensing Techniques

Two topics recently popular in HCI are sensing techniques and input devices. The work in this thesis is *not* input device work: rather than creating a single example of a novel input device, we focus on design tools to empower others to generate their own input devices. Sensing techniques are more closely related. We will discuss several sensing techniques previously studied: these will be divided into techniques that require machine learning and those that do not.

Using machine learning

Sensing techniques leveraging machine learning are very popular in HCI, as they provide power and nuance in sensing tasks. These broadly fall into two categories: classification (where the output is a particular class, e.g., 'one finger touch') and regression (where the output is a continuous value interpolated between training values, e.g., 'slider is at 57%'). Classification may be based on ad-hoc author-generated features [50] or by using ML algorithms to learn important features from a rich representation of the users action, e.g. by generating a variety of features about spatial and temporal aspects of their performance [38, 39, 74]. One common way is to use supervised machine learning, where many labeled examples are used to train a classifier. [89, 82, 57], but this approach requires training (and may create classifiers not portable between different users).

Frequency sweeping for classifying touch gestures has been investigated in multiple contexts, including sweeping capacitance frequencies through conductive materials [89] and sweeping ultrasonic acoustic frequencies through rigid, sound-conducting materials [82]. Other audio-based gesture detection includes ML for learning acceleration profiles related to scratching on an object's surface [38, 74]. Regression may be used on audio signatures, as well, for example to determine position continuously along sliders in 3D printed flute-like tubes [57] .

Vision-based ML gesture detection schemes have used both regular cameras and depth cameras for sensing the position of users' hands relative to a surface [27, 44, 55, 64].

We avoid machine-learning based sensing techniques in this thesis. Employing machine learning requires training each gesture to be sensed. In addition, it requires training each *combination* of gestures to be sensed, as swept sound or capacitance signatures may not combine linearly when gestures are performed contemporaneously. Our work eschews this complication: since we link geometry to sensing, we exploit this link to allow training-free sensing and/or improve our chances of sensing a user's interactions correctly.

Without machine learning

Other explorations have led to clever ways of avoiding the training necessary to use machine learning for sensing. Often this means creating sensing through physical structure design or capitalizing on materials properties.

For flexible objects, this can come in the form of internal switches designed to close when cast-silicone objects are manipulated in certain ways [110], or as conductive material within microchannels whose resistance changes when stretched or bent [65, 85]. Time-domain reflectometry, i.e., sending an electrical pulse through a wire and timing how long it takes to reflect back, can likewise detect user interaction with flexible objects [130], as can adhesive sensing tape [45]. These techniques do not employ digital fabrication: they are hand-fabricated and -tuned. Because we digitally fabricate our artifacts, we can generate knowledge of geometry automatically.

Digitally fabricated soft objects can have stretchable electronics embedded afterwards [136], or be sensed using computer vision and barometric pressure [37, 109]; however, since these objects do not use knowledge of their interactive pieces, they cannot exploit the geometry-sensing link that we use for our work. Similarly, Olberding, et al., created a sensor that is maximally robust to post-fabrication user modification [81]. Again, this technique does not exploit the link between an object's geometry and its fabricated form to improve sensing.

Identity can be embedded in fabricated objects' surface textures (intended for scratching) [40]; indeed identity can also be recorded in invisible chambers *inside* an object for later high-frequency imaging [126]. These projects use knowledge of the fabricated, embedded identity tags to "train" their sensing, however they can only detect an object's identity and not how a user is interacting with it.

Actual user interaction with digitally fabricated objects has been explored, as well, most notably by Willis, et al. Printed Optics allows for sensing fabricated mechanisms (like sliders and buttons) via light shining through tiny channels embedded in a print [128]. This work inspired our own, though Printed Optics does not offer a design tool or a way of providing this sensing without extensive hand-programming.

3.4 Prototyping Tools

Our investigations in this thesis are ultimately looking at techniques for prototyping. There are many different types of prototypes: typically they are split into role, look-and-feel, and implementation [47]. Within each of these types of prototype, there are opportunities to explore high-fidelity prototypes and low-fidelity prototypes. Our work seeks to help with high-fidelity look-and-feel and role prototypes; we recognize that our chosen sensing techniques have some drawbacks as tools for final implementation (for example, the processing time required to recognize interactions using computer vision or audio; this is not a drawback of Fabbing to Sense as much as those particular sensing modalities, see Discussion in

Chapter 7). Prior work on prototyping tools can be split into two major areas: toolkits and functionality definition.

Toolkits

Research has investigated easing this type of physical functionality exploration, often through the creation of toolkits that help users create interactive prototypes quickly.

On the low-fidelity end, simple pushpins and copper tape can create circuitry for testing with cardboard mockups [49], or the TUIs can be "sketched" using augmented reality tools [76]. Another simple solution is attaching an accelerometer to an object—which the user can create through any process—and using its output to detect interactions [46].

More sophisticated toolkits may include multi-purpose programmable microcontrollers [5], or even be designed for prototyping interactive devices on cloth [18]. Designers can have extra freedom with placing electronics when a board becomes a smart substrate that interlinks them automatically [119], or can use snap-together sensing and actuation modules [8, 34, 61] that could be programmed in a visual language [120].

These techniques have several important limitations: most importantly, they *constrain exploration*: if a user wishes to include a 3-inch slider in his design, but the kit only offers a 2-inch slider or a 5-inch slider, he has to make due. Our work's use of digital fabrication for input component creation gives designers this flexibility in design exploration, without tying them to pre-defined form factors.

Circuitry, and thus any existing electronics, can be integrated directly into a 3D printed object by laying down conductive material [106, 121, 95], or by leaving voids to be filled with conductive material post-print [100]; this still constrains designers to what already exists, as only the connectors are arbitrary and must still match up to pre-existing electronic components. Techniques like adding stick-on sensors or sensing tags to existing or crafted physical objects [66, 138] can mitigate this challenge, however they lead to one-off prototypes that designers must modify by hand and cannot easily share: using digital fabrication, designers can create a design, test it, then modify it digitally to create an improved design; they can even share it with coworkers by simply sending a file. A more general purpose sensing *core* can be added to new prototypes, necessitating that only the body of the design change each time [26]: we build on this idea and leverage our knowledge of an object's geometry to improve it.

Authoring Functionality

We also examine another important question: how is functionality defined for interactive prototypes? Software can be automated using screenshots as in Sikuli [137], or with visual "block"-based programming languages that are accessible to even children [93]. Programming by demonstration can also aid novices in developing interactive applications with sensor components [42], similarly functionality can be inferred from wireframe mockups of applications [62]. To integrate hardware and software functionality, some researchers have explored

augmented reality [77] or projection [4] techniques to allow interactions to be defined only in code without functional hardware; recent online tools like 123D Circuits allow for simulating hardware and testing code for it without requiring the physical circuit be built [1].

3.5 Conclusion

We have discussed related work from simulation, sensing, 3D modeling, and prototyping tools, and described how our work fits into and links these different research areas. We will now dive into each of our projects.

Chapter 4

Midas: Capacitive Sensing of Custom 2D Layouts

So Midas, king of Lydia, swelled at first with pride when he found he could transform everything he touched to gold...

— Claudian, *In Rufinem*

4.1 Preamble

We begin our exploration of the interlink between sensing and geometry at the simple end of our spectra: Midas links 2D geometry, fabricated from conductive material, to capacitance sensing. This chapter focuses on interfaces created on the surface of flat and developable 3D objects, where interaction is triggered by a user's direct touch.

4.2 Introduction

Ubiquitous, cheap microprocessors have led to a vast increase in consumer products with built-in digital user interfaces. Many of these devices—thermostats, game controllers, and personal medical devices, to name a few—rely on touch sensing to provide input to their user interfaces.

The rise of the iPhone and subsequent touchscreen-based smartphones has been a very visible use of touch input, however these devices rely on software to prototype and create interactions. An app designer can create an interactive graphical user interface using on-screen prototyping tools, purely software with no hardware required. For devices where touch input is not co-located with screen output, for example when a designer wants inputs on the back of a screen (for example to avoid the fat finger problem [11]) or on a device with no screen at all, prototyping becomes much more complicated.

Figure 4.1: Midas enables users to define discrete and continuous touch sensors with custom shapes and layout. It generates fabrication files and assembly instructions. Designers can also define the interaction events of their prototype.

Using pre-packaged sensors has important drawbacks. It *constrains exploration*: pre-defined physical form factors restrict the space of realizable designs. For example, available buttons can be too bulky or too small, or sliders may be too long or too short. Most sensors also lack *physical flexibility*: they are not easily applied to non-planar surfaces or added to existing objects. Finally, a large *gulf of execution* remains between digital design files and physical prototypes: sensors must be manually placed and wired one-by-one. This process is tedious and error-prone; physical prototypes can easily deviate from digital design files if a wire is incorrectly placed or forgotten. Our work leverages digital design tools and enables designers to use the growing range of fabrication processes to create custom, durable, replicable, iterable, and shareable touch sensors.

We take inspiration from the success of GUI editors. These editors enable designers to specify layout, size, and characteristics of widgets. They also isolate designers from specifying the "plumbing" that connects widgets to event callbacks. *Midas seeks to make the creation of physical touch-sensing interfaces as fluid as the creation of graphical user interfaces in*

GUI editors.

Midas is a software and hardware toolkit to support the design, fabrication, and programming of custom capacitive touch sensors (see Figure 4.1). With Midas, designers first define the desired shape, layout, and type of touch sensitive areas and obstacles in a *sensor editor* interface. Designers can choose from buttons, 1D sliders, and 2D pad sensors. For discrete (button) inputs, designers can use polygon shapes or import graphics to define custom shapes; other types are adjustable in size and aspect ratio. Once a designer settles on a layout, Midas automatically synthesizes appropriate capacitive touch sensor pads and routes connecting traces, avoiding user-defined obstacles, to a central touch sensing module via a circuit board grid routing algorithm [59]. Midas then generates layout files and step-by-step instructions that designers use to fabricate the sensors using rapid manufacturing techniques. Our prototype cuts sensors from adhesive-backed copper foil and vinyl on a commercial vinyl cutter. We also demonstrate using a circuit board milling machine and a Silhouette Cameo paper cutter to fabricate Midas sensors. Designers then transfer their flexible, adhesive-backed sensors onto the target object and connect the fabricated sensors to a small microcontroller using the routed connections. The microcontroller detects touch events using charge-transfer sensing [87] and forwards events to a PC. Once assembled, designers can define interactivity on the PC using the sensor editor. Midas supports both record-and-replay actions to control existing local applications, and WebSocket event output for novel and remote applications. WebSockets enable designers to write touch-sensitive applications using standard Web technologies (HTML and JavaScript).

We demonstrate Midas's expressivity with a number of examples. The authors used Midas to create several touch-sensitive interfaces, including recreating prototypes of existing and published systems.

The main contributions this chapter describes are:

1. a novel method to create custom-shaped, flexible capacitive touch sensors by synthesizing sensor pads and auto-routing connections, as well as instructions for assembly and use, from a high-level graphical specification

2. a design tool using this method to enable users to to fabricate, program, and share touch-sensitive prototypes

3. an evaluation demonstrating Midas's ability to create a variety of functional prototypes quickly and cheaply

The Geometry-Sensing Link

For the Midas project, we exploit the convenience of algorithmic routing. The designer creates a sensor layout which matches with her desired aesthetics and objects, then the machine performs the task of creating structures that allow the designer to detect user interactions with the sensors. This is convenient for the designer, as the task of laying out traces is hardly a glorious one, and it may require additional skills and knowledge about

reasonable trace widths and the layout of the sensing board. The layout is our link to sensing: the machine knows the routing, thus it has *a priori* knowledge of what it will be sensing. When an interaction triggers a change in capacitance, Midas associates this with the linked sensor and begins the programmed response.

4.3   Designing with Midas

Users

The target users for Midas are designers who have 2D layout expertise, but who lack experience in electronics and potentially also programming. We target these types of designers through affordances familiar from graphic design programs, instruction-based assembly using a single hardware component, error detection/correction, and easy-to-use sensor output.

Midas echoes graphic and Graphical User Interface (GUI) design programs, offering designers a familiar drag-and-drop interface. Midas also supports scaling via direct manipulation, and has the ability to import custom sensor shapes as PNG images.

The instructions generated by Midas walk designers through machine setup, sensor fabrication, and microcontroller connection. This instruction set assumes no knowledge about the machines, fabrication process, or electronics, and uses color-coded wiring to ensure circuit legibility.

In the case where setup goes awry, Midas can detect two common fault types by performing pattern recognition on its sensor inputs. The faults are "stuck on", when sensor traces are too close together and are touching or capacitively coupled, and "flicker", when the microcontroller's connection to a sensor rapidly changes and indicates a poor attachment.

Midas's sensor output is available for interaction design via two channels: designers can record literal clicking and typing events on their screen that will be triggered by a sensor input ("record-and-replay"), or they may use JavaScript, a programming language with which many designers are familiar, to accept the events in the form of WebSockets messages for further processing in an interactive webpage.

These usability features will be discussed in more detail in the implementation section.

Design Walkthrough

We discuss the interface affordances and the workflow of Midas (Figure 4.1) with a concrete running example: A designer would like to explore back-of-device and bezel (off-screen edge) interactions for a mobile phone. In particular, she would like to scroll through a list of emails with a slider on the back of the device, and open, reply to, and delete messages via sensors on the bezel under the phone user's thumb.

Drawing Sensors

Users start by loading an image of the physical prototype they want to augment into Midas's sensor editor. The sensor editor (Figure 4.2) allows a user to create the sensor layout, and define interactive behavior for each sensor. The background device image helps designers with correct scaling and positioning. Currently, Midas supports 2D images, including flattened 3D models. Future work will investigate direct support of 3D models. Sensor positioning works analogously to a GUI editor; users choose sensor types and drag them to the desired location on the canvas. Midas supports individual discrete buttons, one-dimensional sliders, and two-dimensional pads. Buttons can take on arbitrary shapes— users can import any graphics file (in PNG format) or draw custom polygons. Sliders and pads are currently restricted to rectangular shapes; however, their size and aspect ratio can be modified to fit the requirements of the prototype at hand. Users may also define obstacles using the same drawing tools to restrict routing—Midas will route connections around these obstacles. In our phone example, the designer creates one slider and three discrete buttons. For the buttons, she loads custom shapes created in a drawing program. She defines a circular obstacle around the phone's camera so the camera will not be obscured during connection routing.

Fabricating and Applying Flexible Sensors

Once users complete a layout, clicking the "Create Stickers" button generates fabrication files. First, certain components are automatically split into multiple sensing pads. For instance, a slider can generate four interdigitated pads (Figure 4.5, third template) for continuous finger tracking, while 2D pads result in two separate layers of triangular pads (Figure 4.6). Second, Midas generates conductive traces that will connect each of the pads to Midas's touch controller. An additional mask file, to be fabricated in vinyl, includes cutouts of only the sensor shapes: it will cover the traces both for aesthetic reasons and to prevent stray touch events. Midas's connection routing determines the exact position of each touch area. Should the user want to experiment with positioning, Midas can also skip routing and only generate individual touch pads. However, the user must then manually connect wires to each sensor and register the sensor in the interface.

The pad creation and routing step generates a set of graphics files (in SVG format) and an instruction sheet (in HTML) which appears in the user's browser (see Figure 4.3). This sheet contains step-by-step instructions describing how to fabricate the generated files. For our implementation, instructions include which SVG files to cut in which material and how to transfer the cut designs to the prototype object.

In our phone example, the designer generates one SVG file for the touch areas and one to mask the traces, which prevents stray touch events. Following the generated instruction web page, she feeds copper foil into her vinyl cutter and cuts the corresponding SVG file. She then substitutes a vinyl roll and cuts a mask layer. As both materials have adhesive backing, she sticks the copper and vinyl layers onto the phone she wishes to modify. Once the adhesive

layers are applied, she tapes the end of the routed traces to the Midas hardware, which is plugged into her computer via USB. Since the design files for her prototype are digital, she also sends them to colleagues in another office for a second, remote test. With the design files and a vinyl cutter, her colleagues can then recreate a working Midas prototype.

Connecting Hardware to Software

Midas senses touch events with a dedicated touch controller circuit board. Users do not have to program or assemble any electronics—they may treat the entire setup as a prototyping dongle. Users do have to connect the end of the traces to the controller's rainbow ribbon cable, either by taping the cable leads onto copper traces or by soldering them.

    To complete a prototype, users return to the sensor editor. In many toolkits, mapping hardware components to named objects in software can be error-prone—it is easy to swap wires or connect to an incorrect pin. If the user prints a fully-routed design, Midas generates

Figure 4.2: Midas's sensor editor takes its cues from GUI editors: designers first lay out sensing areas through direct manipulation; they later define interactions for each sensor using a property inspector.
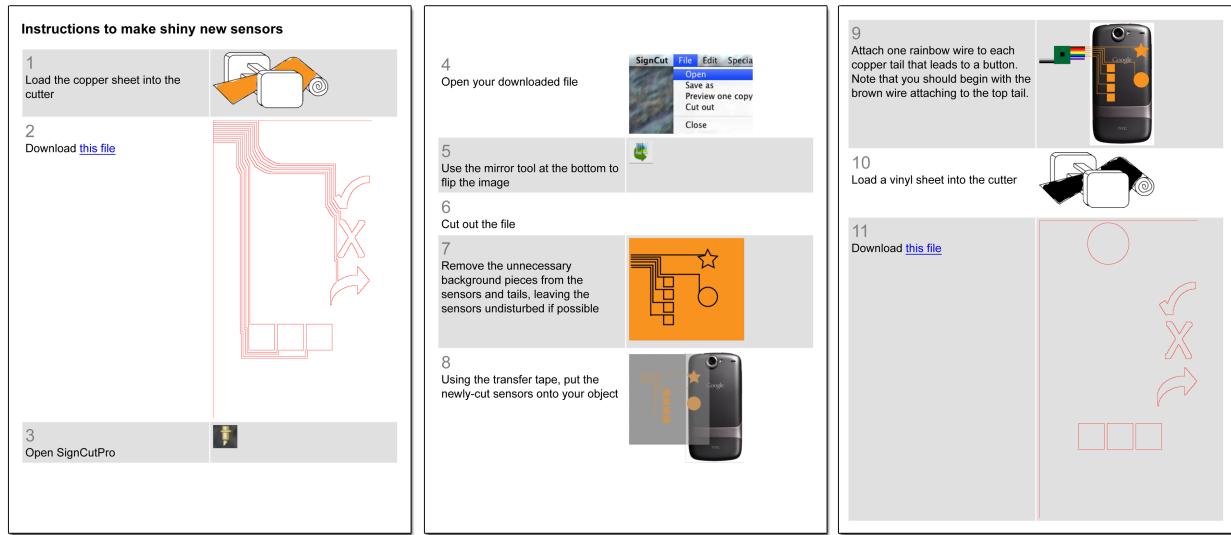
Figure 4.3: Auto-generated step-by-step instructions in HTML format lead the user through the fabrication and assembly process. Relevant design files are hyperlinked to specific steps; instructions also include general help on processes, e.g., how to use transfer tape to apply a sensor onto an object.

instructions for aligning touch areas with specific ribbon cable colors. If the user decides to wire the design herself, this mapping has to be authored. Midas uses guided demonstration to assist with this process. For buttons, the user selects an input element in the UI and clicks the "Tie to Stickers" button; next she touches the corresponding copper sensor. Midas listens for status change events and automatically assigns hardware pins. Midas registers sliders similarly: users are asked to swipe a finger along the slider.

Midas's editor interface displays incoming touch data visually, re-coloring touched sensors in pink on the sensor editor, to aid the user in debugging. If something goes wrong during the connection stage, it is apparent to the user. Midas also reads the data stream for common errors. If it detects that two wires may be too close together and triggering each other, or that there may be a faulty connection from a wire to the board, that information is displayed to the user in a connection status area.

Adding Interactivity

Designers have two options for authoring interactivity: record-and-replay of mouse and keyboard events (a strategy adopted from BOXES [49] and Exemplar [41]), or touch event output to control applications via WebSockets. To record and replay interactions, designers select a sensor in the editor, then click on the "Record Interaction" button. They can then control any open application (e.g., start or stop a media player application, or drag a volume slider). Midas records the generated keyboard and mouse events and can replay them later

Figure 4.4: Users start to record GUI interactions in the sensor editor (A); they can for example activate the browser, enter text (B), and click on a search result (C), before concluding the recording (D). This sequence of actions can then be triggered by a touch event.

in response to touch input (Figure 4.4).

The types of desktop UI actions that can be executed depend on the button type. Individual buttons can be tied to an interaction script, a sequence of keyboard and mouse events recorded by the user. Sliders are linked to exactly one horizontal or vertical line on the screen to be controlled by clicks along its length. 2D pads can control a 2D area on the screen analogous to a slider area. For sliders and pads, the user must capture the location on the screen that she wishes to control with the slider or pad. This is done by clicking at each end of the slider or in opposite corners of the pad, guided by Midas prompts. As the user adjusts the sensitivity (number of discrete buttons) of the slider or pad to be printed, the interaction with the captured on-screen slider or pad becomes more fine-grained, also.

Record-and-replay does not require programming, but it is brittle; changes in application layout or response latency can break a recorded sequence. To let users author more robust interactions, Midas uses WebSockets to send touch events over the Internet. This requires programming, but WebSockets enable designers to work in the languages many are most familiar with: HTML and JavaScript.

In our phone example, the designer chooses WebSockets as she wants to demonstrate how touch events can control a mobile email application. She creates a mockup in HTML and writes JavaScript functions to receive touch events.

4.4 Implementation

In this section, we describe the Midas design tool, the hardware it uses for sensing, and fabrication techniques for manufacturing compatible sensors.

The Midas CAD tool

We discuss the key parts of the Midas CAD tool: generating sensor pads and routing pads to the touch controller.
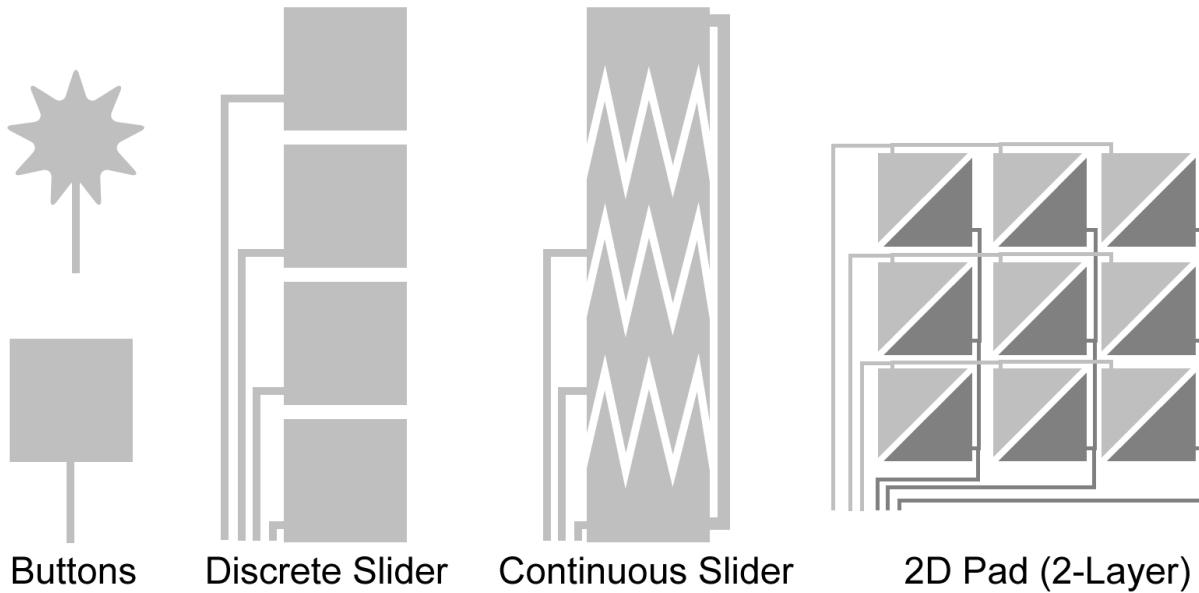
Figure 4.5: Midas can generate four different types of sensors: discrete buttons, discrete sliders, continuous sliders, and 2D pads. The pad uses row-column scanning and requires multi-layer construction because traces cross.

Generating Sensor Pads

The Midas sensor editor supports four types of touch sensors: discrete buttons, two types of 1D sliders, and 2D pads. The resolution of pads and segmented sliders can be set through a parameter in the sensor editor. The current editor is written in Java using the Swing GUI Toolkit. Figure 4.5 shows example templates for each sensor type. The two types of sliders are based on different sensing approaches. The first, segmented slider, is made up of individual rectangular touch segments. Users specify how many segments the slider has. Continuous sliders offer finer resolution, but require a different detection approach. We use Bigelow's design of interdigitated electrodes [14]. In this design, as a finger slides across the pads, the surface area of the pads underneath the finger changes as pad digits get smaller or larger. Because capacitance is proportional to contact surface area, the measured capacitance of each segment changes during the finger's slide. Though finer in resolution, only one such slider is supported by our current sensing hardware. Increasing the number of supported sliders is possible with additional engineering effort.

2D pads use row-column scanning to reduce connecting traces. For example, a 5×5 array requires 25 individual traces, but only $5 + 5 = 10$ row-column traces. This design requires a dual-layer construction where horizontal traces are isolated from vertical traces. We use copper foil applied to vinyl foil in our cutter, so each layer already has an insulating substrate. Designers thus first apply the bottom conductive layer, then place the top layer directly over it (see Figure 4.6). To create a mask layer that covers the topmost copper

Figure 4.6: 2D pad sensors are fabricated in two different layers that are then superimposed. Because each copper layer has a vinyl backing, no other inter-layer masking is required.

traces, we generate a design file containing pads from all layers, but no traces. This layer is cut in vinyl. While for other layers designers transfer the pads and traces, for the mask layer they peel and transfer the surrounding background shape with sensors and obstacles cut out (see Figure 4.13, left).

Routing Pads to the Touch Controller

Midas employs an auto-routing algorithm to generate conductive traces connecting electrodes to the Midas touch controller. User-defined obstacles are avoided. We implement Lee's breadth-first maze routing algorithm for single layer paths [59] (see Figure 4.7). This algorithm randomly selects a source and target (the source is the trace anchor for connecting to the microcontroller, the target is all pixels on the perimeter of the sensor). Then, beginning at the source, floods the grid by marking each viable pixel—i.e., pixels which are not on obstacles and which will not touch other sensors—with its distance from the source. When any of the target locations are reached, the algorithm traces back through the marked pixels, maintaining straight traces as much as possible, and generating a trace along the path selected. For 2D pads, we perform two independent routings: one for the row layer and one for the column layer. Our current algorithm does not generate vias (connections between different conductive layers). When auto-routing fails, we employ an iterative search by adjusting the position where traces connect to sensor pads, routing the sensors in a different order, or moving the position where target traces connect to the touch controller. In our experience, this basic routing algorithm has performed adequately, though there are designs that cannot be successfully routed. For such cases, the algorithm could be replaced
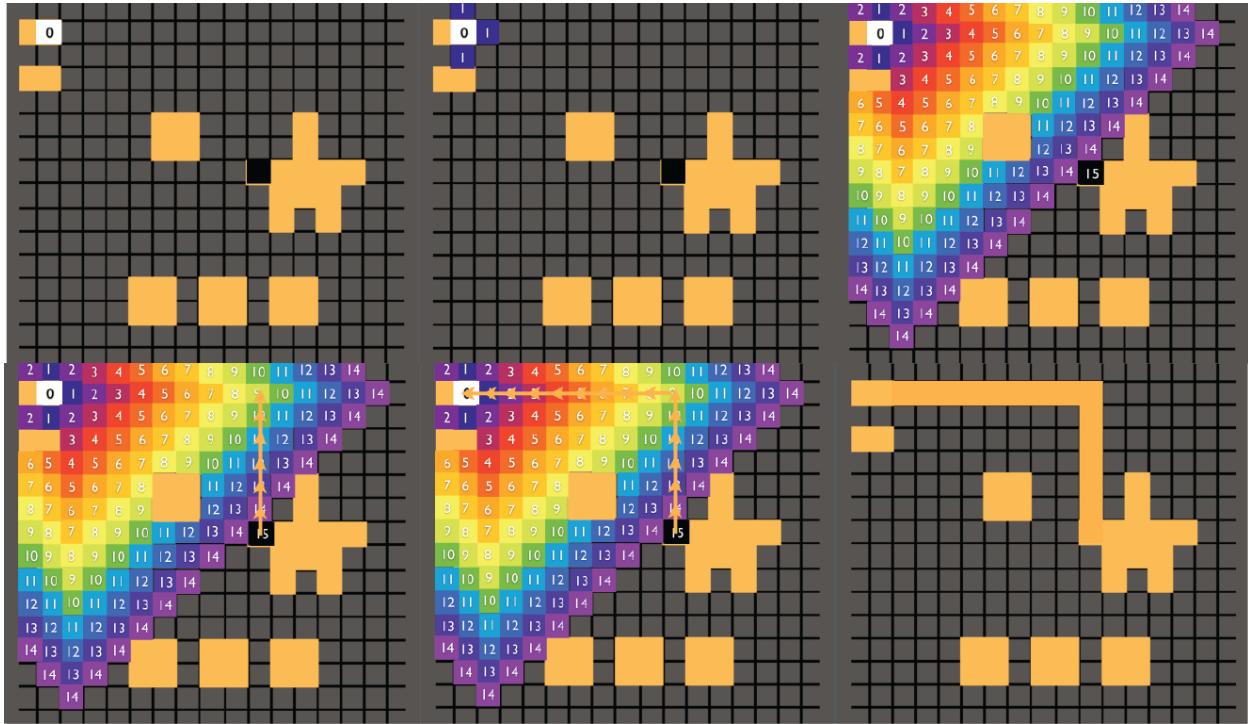
Figure 4.7: The Midas routing algorithm, based on Lee's [59], first selects a source and target point. It then floods the grid, beginning at the source, marking each viable "pixel" as one further than its pre-marked neighbor. Once the target has been reached, the algorithm traces back through the marked pixels (we selected to make vertical moves before horizontal) to the source and creates a trace.

with more sophisticated routing techniques that include user input, though such techniques require that the user has a correct mental model of the routing process.

Midas currently offers generic suggestions when routing fails, e.g.: Sensor reply may be too close to sensor delete. Try moving sensors away from the edge and each other.

The Midas Hardware

We discuss the components of the Midas hardware: the mechanism of capacitive touch sensing and the fabrication of compatible sensor pads. We also describe how the sensed and digitized signal is used to design interactions.

Sensing Mechanism

Midas relies on capacitive touch sensing. The original Midas touch controller (Figure 4.8) is based on an Atmel microcontroller board [114] and capacitive sensing chips from Quantum (QT1106) and Freescale (MPR121) Semiconductors. For discrete inputs, both chips rely

on charge-transfer sensing using single-wire electrodes: the electrodes are part of a simple RC circuit in which an output pin is set high, and time is measured until an input pin also reads high. This time is proportional to the capacitance of the circuit: when a person touches an electrode, the circuit capacitance and the charge transfer time both increase. The Quantum chip also implements Bigelow's design to extract continuous position readings from interdigitated electrodes by interpolating based on relative capacitance between neighboring pads [14]. The microcontroller runs software written in embedded C to interface with the sensor chips and communicates touch data to a connected computer over USB. It recalibrates the touch sensing chips periodically to ensure floating sensor values do not lead to erroneous touch readings.

Our new Midas touch controller uses an Adafruit Blufruit microcontroller with Bluetooth compatibility. It does not implement Bigelow's interdigitated sensing but could do so in the future; for discrete sensing it leverages the Arduino CapSense library[1] which uses two-wire electrodes; it determines the capacitance by timing the charge transfer time between the pins. The new board design also has sturdy milled connection pads to attach to traces; these can be attached using Z-axis conductive tape (e.g., 3M's Z-Axis Conductive Tape 9703). This board transmits sensed readings directly to a phone over a Bluetooth connection.

Fabrication

Midas generates vector graphics files in SVG format for the electrode and mask layers. These files can be used to control digital fabrication processes. Our prototype currently cuts conductive, adhesive-backed copper foil on a commercial vinyl cutter—a plotter with a cutting knife instead of a pen. This medium has multiple advantages. First, it is cost-effective and readily available: vinyl cutters are in the same price range as laser printers (ours,

---

[1]`playground.arduino.cc/Main/CapacitiveSensor?from=Main.CapSense`
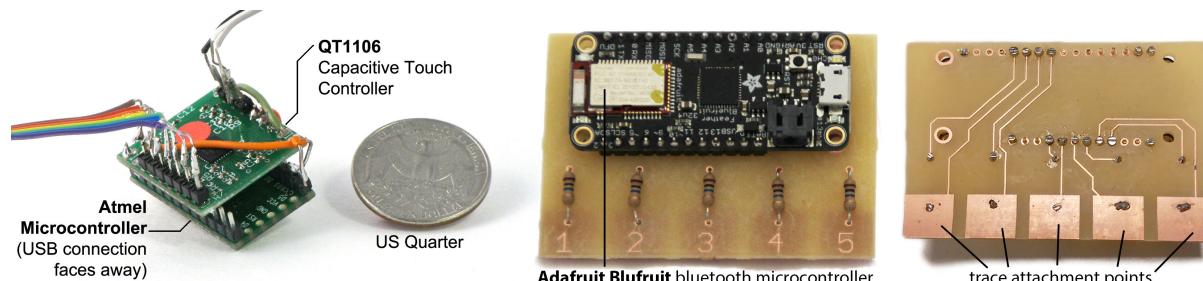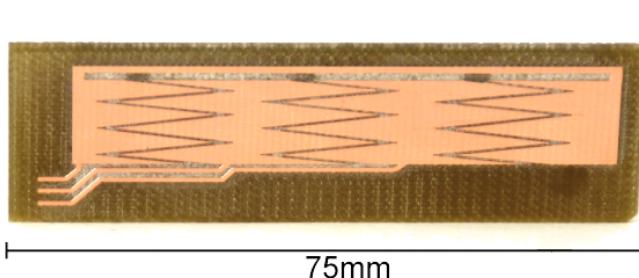
Figure 4.8: The original Midas touch controller board (left) uses a commercial capacitive charge transfer detection chip to sense touch events. Events are relayed to a computer via a mini USB connection on the back. The ribbon cables are used to connect to the end of routed traces. The new board (right) uses an Arduino touch sensing library and a Bluetooth-compatible microcontroller for wireless data transmission and sensing. A US quarter is shown as a size reference.
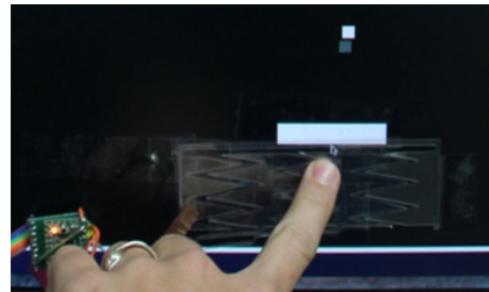
a tabletop model with a $35cm$ bed, cost $200); and copper foil costs a few dollars per foot. Second, copper has excellent conductivity. Third flexible, adhesive foil is easy to apply to non-planar surfaces. However, there are important drawbacks as well. Most importantly, the cutting process and manual weeding (removing background material) determines a minimum feature size for traces. Thin geometric features can also break during transfer, and the weeding process can be tedious and time-consuming. We found the most success adhering the copper sheets to vinyl sheets and cutting both layers at once. This setup has the added benefit of allowing designers to prototype touch interactions on conductive surfaces (e.g., aluminum casing) as vinyl is an excellent insulator. Alternative fabrication processes may be preferable to copper foil cutting when higher precision or durability is required. Three promising approaches are circuit board milling, which can produce smaller features but is limited to rigid boards; cutting from Indium Tin Oxide (ITO)-coated plastic, which allows for transparent sensors but is not self-adhesive and is less flexible than vinyl; and conductive inkjet printing, which can produce the smallest features, but is not yet available to many end users. As a proof of concept, we produced a touch sensor on an LPKF circuit board milling machine, and one on a Silhouette Cameo paper cutter (see Figure 4.9).

Debugging

Midas offers basic debugging support. The interface displays incoming touch information from the microcontroller, highlighting activated sensors. We have also implemented basic regular expression filtering on the touch stream to help the user identify potential problems. A time-stamp and code representing each touch event is stored in a stream. When a sensor is "stuck on" for more than 10 seconds, Midas reports that that sensor may be touching another wire. When a touch sensor "flickers" on and off more than twice within $500ms$,

A slider fabricated on the LPKF mill   A clear slider on ITO plastic

Figure 4.9: Two example touch sensors fabricated with alternative processes: left, a hard sensor created on a circuit board mill. Right, a sensor cut from Indium Tin Oxide-coated transparent plastic..

Midas suggests that there may be a faulty connection from that sensor to the microcontroller.

Event Output

Once the user has assigned interface scripts to sensors, Midas listens for events from the touch controller. When a touch event matches the key of a saved interaction, that interaction's associated script is executed.

Record-And-Replay

In record-and-replay, the user selects a sensor and records a sequence of mouse and keyboard actions that should be played back when the sensor is touched. Early prototypes of Midas used Sikuli for this purpose—a scripting language based on computer vision analysis of screenshots [137]. While more robust than hardcoded click locations, Sikuli was designed for automation scripts rather than interactive control, and the latency of invoking and executing scripts was too high. Our current prototype uses the Java Robot API [51] to captures and replay both mouse clicks and keyboard events. We share this approach with the BOXES system [49].
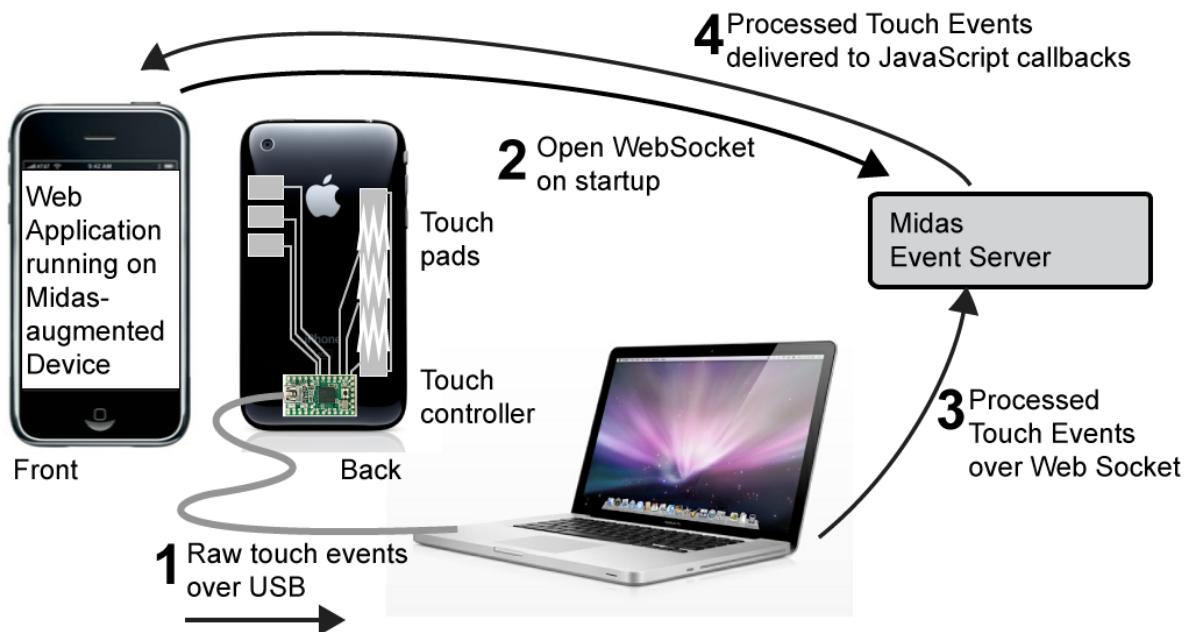
Figure 4.10: Midas's socket event output enables designers with programming knowledge to create web applications in HTML and JavaScript that react to touch input outside the screen area of a phone or tablet.

WebSocket Communication with Web Applications

Record-and-replay is restricted to applications running on the same machine as the sensor editor, and it is limited to mouse and keyboard event injection. To surmount this limitation, Midas can also export touch events to remote clients via a built-in server using the Web-Sockets API. For example, an application running on a smart phone can open a WebSocket connection to Midas and receive a callback for any Midas button, slider or pad event. The callback function receives an event object describing which sensor changed state, and the value of the new state (e.g., on/off, or slider value).

Our WebSockets server is implemented in node.js using the socket.io library. We chose WebSockets because it offers full-duplex communication at low latencies, and, more importantly, is supported by modern web browsers. This means designers can author user interfaces that respond to Midas touch input in HTML and JavaScript. There are two main benefits to these technologies: (1) many designers are already familiar with them from web design; (2) developed interfaces can be deployed on any device with a compatible browser, even if that device does not offer a way to directly connect external hardware. For example, it is difficult to directly connect sensors to Apple's iPhone or iPad.

With our WebSockets architecture (Figure 4.10), designers open a browser and enter the URL of an HTML file they have placed in the Midas server directory. This file opens a socket connection from the phone browser to Midas. When Midas receives events from the touch controller, it forwards them to the client, which can then show visual feedback.

Bluetooth

Our new touch controller leverages Bluetooth for communication. The detected data are streamed similarly to those in the WebSockets condition, however in this case they are transmitted via Bluetooth, and the receiving device must be paired to the controller.

4.5 Evaluation

In order to demonstrate that Midas's sensing and fabrication technique fits our criteria of being a cheap, fast, and flexible method of prototyping, we elaborate on each of these criteria below.

Cost-Effective

Midas enables touch-sensitive prototyping for roughly the cost of a laser printer. The vinyl cutter used for the Midas project, a tabletop model with a $35cm$ bed, cost \$200. Copper foil costs a few dollars per foot.

Alternatives to this fabrication method include cutting Indium Tin Oxide (ITO)-coated plastics, using silver ink in an inkjet printer, and fabrication using a circuitboard mill. ITO-coated plastics allow for transparent sensors; ITO costs roughly \$.$25/in^2$ and can be

fabricated on an inexpensive paper cutter. Silver ink can be used in many existing inkjet printers, and costs roughly $1/mL$ or $2/ft^2$ of total coverage. Circuitboard milling has a high startup cost (a suitable machine can cost as little as $800, and upwards beyond that), but the per-unit expense is low, about $17/ft^2$.

On the sensor side, the full sensor setup we prototyped with cost $36: $16 for the Arduino Teensy microcontroller board, and $10 for each of the capacitive sensing breakout boards. This cost could be reduced further with custom circuitboards (our setup was assembled from commercially-available boards for expediency), and the setup is can be easily detached from a prototype and reused in a new iteration or for a different project.

Fast

We did an informal first-use study, and it took all participants < 1 hour to both receive training and create working media player peripherals with the system.

We recruited three participants for this study. Two were graduate students at UC Berkeley (in Computer Science and Mechanical Engineering), and the third was a software engineer at a local technology company. All had some prior experience with prototyping and electronics.

Procedure

Participants received a walkthrough of Midas including a simple record-and-replay task to launch a browser based on a single button input. Participants were then asked to design a physical control interface for a media player (iTunes). No other constraints were given as we wished to encourage exploration. Participants completed a post-task questionnaire with open-ended questions on interface usability and workflow utility.

Results

All participants successfully designed media players (Figure 4.11). Participants commented positively on how Midas aided them with the task of physical construction—both by routing connections and through the generated instructions. Record-and-replay was easy to comprehend and effective for the given task. Though the task did not require programming, two participants expressed interest in receiving touch events in their own applications. We take this as corroboration for the utility of our WebSocket server. Participants additionally identified several areas for improvement (including more detailed instructions, additional help for auto-routing failures, and additional feedback for touch events), which are included in our most recent design revision, described above.

Flexible

To demonstrate Midas's expressivity, we built several interactive systems that used all sensor types, and output using both WebSockets and record-and-replay.
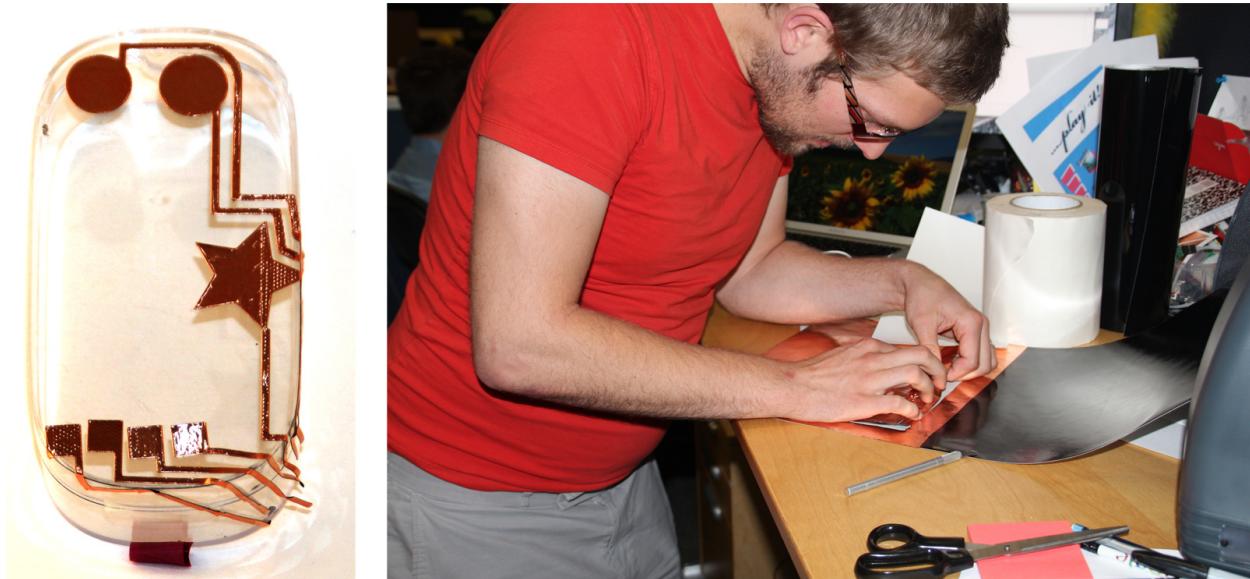
Figure 4.11: A study participant's sensor layout for a PC media player peripheral.

Text Entry

Wobrrock's EdgeWrite [132] is a unistroke text entry technique based on activating a series of corner points of a rectangle. Wobbrock demonstrated that this technique can be implemented using four discrete capacitive touch sensors [131]. We cut four discrete buttons and a mask with Midas, attached them to the back of a smartphone, and implemented the EdgeWrite recognition algorithm in JavaScript (Figure 4.12). Using socket events, we demonstrated how EdgeWrite can be used to enter text on the back of a mobile device, leaving the screen unobstructed. The implementation is functional, though latency for detecting single button presses was higher than expected (>100ms). We plan to investigate ways to increase responsiveness in future work.

Game Controller

To test the responsiveness of Midas's continuous slider, we created a simple game controller for Breakout, in which players horizontally position a paddle to bounce a ball into layers of blocks (see Figure 4.9). In less than fifteen minutes, we attached the slider and mapped slider position to paddle position using record-and-replay. The slider is more responsive than individual buttons, and we were able to control the paddle accurately enough for gameplay. The slider's response is non-linear across certain regions, however. Accounting for this is left to future work; our instinct says that it may be caused by electromagnetic interference from the thin traces and sharp right angles in the design.

We fabricated two versions of this game controller: one from a circuitboard milled slider, and a second from Indium Tin Oxide (ITO)-coated transparent plastic. This underscores
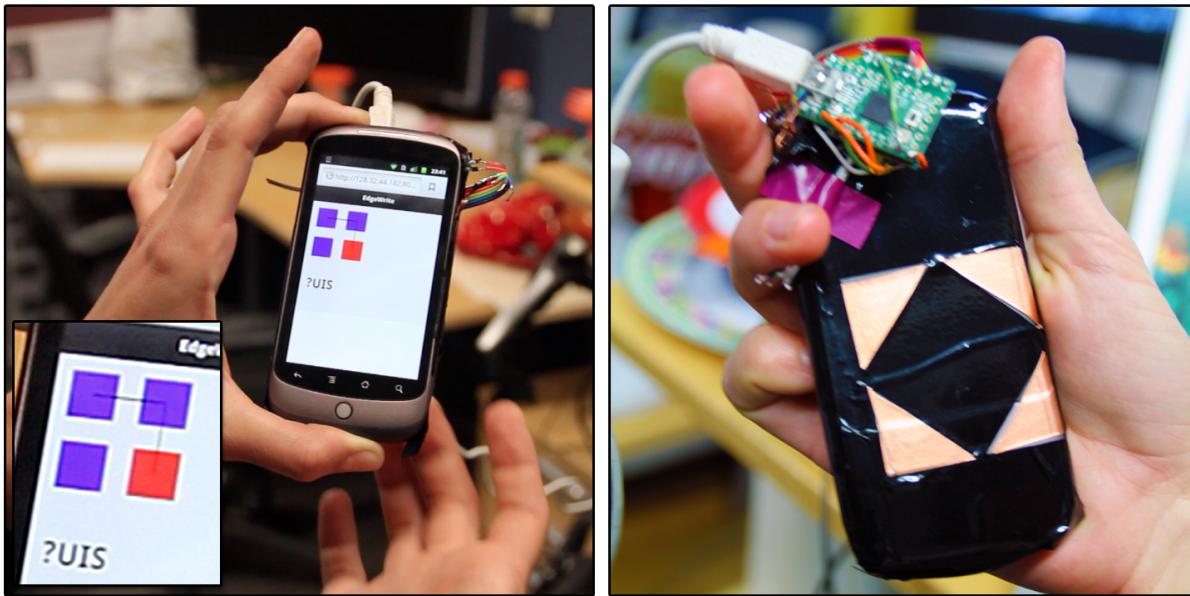
Figure 4.12: We implemented Wobbrock's Edgewrite on the back of a cell phone using a 2x2 pad and WebSocket events sent to a web page.

the point that only the conductivity of the material is important; whether it is flexible/rigid or opaque/transparent has no bearing on the functionality.

Music Postcard

At a recent media festival, a promotional poster printed with conductive ink enabled passersby to select and play music from a number of artists by touching corresponding areas on the poster [79]. We implemented a postcard-sized version of this poster (Figure 4.13). We scaled back the size to conserve resources; large designs are possible and only restricted by the cutter's bed width. Our version uses six discrete buttons and one continuous slider to control playback and volume of music clips on a desktop computer. We again cut a vinyl mask layer to prevent stray touch events. We used Midas's record-and-replay function to remote control the iTunes music player.

Papercraft Pinball Machine

Papercraft is the art of cutting and folding paper into 3D models. To demonstrate how Midas can be used for attaching sensors to more complex 3D shapes, we created a papercraft pinball machine which can control a pinball game on a laptop. First we downloaded a template design for a pinball machine [88], cut it out, and assembled it. We then loaded the already-flattened template model into Midas's 2D editor interface and defined the negative space around the model as a routing obstacle (Figure 4.14). This guaranteed that all trace routing would
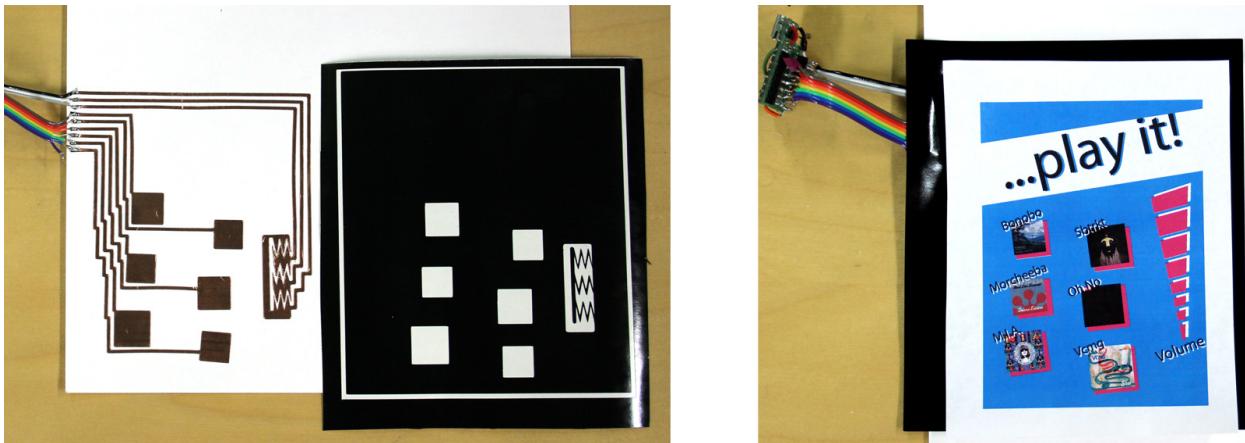
Figure 4.13: Our music postcard lets users sample tracks by different artists. Left: Circuit and mask layer; Right: assembled postcard.
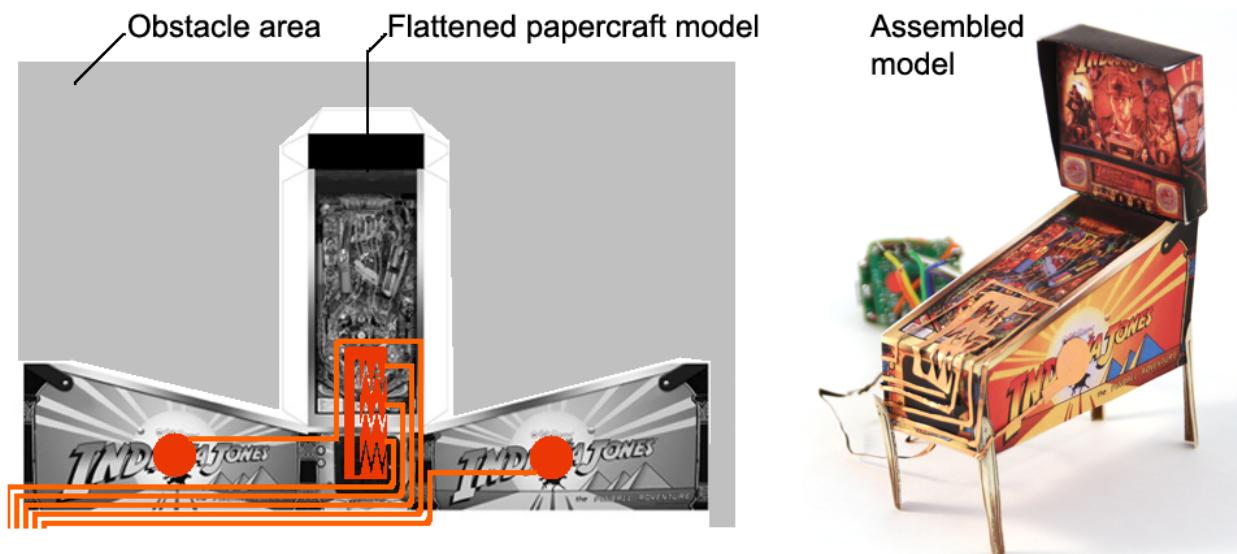
Figure 4.14: For our papercraft pinball machine, we defined the negative space in the template as an obstacle to restrict sensor routing.

be on the surface of the model. After attaching the two buttons for the bumpers and a continuous slider for ball release control, we used record and replay to control a desktop pinball game.

4.6 Discussion

Midas, as one of the first fabrication papers in the HCI community, explored a variety of interesting challenges and opportunities around digital fabrication. It serves several purposes well, but there are limitations caused both by the fundamental approach and the particular engineering of our prototype.

Sweet Spots

Midas is well-suited to prototyping particular kinds of objects. It allows facile testing and iteration on unusual types of touch interactions, for example on the backside of devices: touch screens allow input and output, but adhering a smartphone-like screen to every location a designer wants to sense a touch input is prohibitive. Additionally, unique sensor shapes, like stars or check marks, can be tested with Midas much more simply than with custom-fabricated touchscreens.

Midas opens opportunities for custom sensors on the surfaces of flat or singly-curved objects, as well as 3D objects that are developable [6]. While pre-fabricated sensors, such as those used for prototyping with Arduino [5] or d.tools [42], can enable prototyping on flat surfaces, they are typically not flexible and thus cannot be applied to non-flat surfaces. As Midas-generated sensors are flexible, they work well in these situations, and can even be used to sense touch on the surface of flexible and foldable materials like paper. Additionally, Midas allows designers to specify the real size that they desire for their inputs without needing to rely on a kit of pre-made sensors.

Limitations

The current Midas prototype has some important limitations. A few are inherent to our chosen approach, while others could be mitigated with additional engineering.

Physical Constraints on Realizable Designs

The current manufacturing process places certain physical constraints on realizable designs. Both the accuracy of the vinyl cutter on copper and challenges in weeding and transferring cut designs currently restrict traces to approximately $2mm$ minimum thickness. Our inexpensive cutter also has difficulties with acute angles such as those in the interdigitated slider, however we have demonstrated manufacture with higher-quality cutters and alternative techniques that mitigate this issue.

Touch Sensing Chips have Limited Capacity

The touch sensing chips we use have limited capacity. In addition, continuous sliders use different hardware resources than other inputs and therefore need to be treated differently by the designer. The QT1106 has 7 discrete sensing channels and can support one continuous

slider; the MPR121 has 13 discrete channels. The sensor editor keeps track of resources required by the current design and notifies designers if they exceed the capacity of the touch controller. While we currently do not support using multiple touch controllers for a single project, a future circuit board revision could offer such support.

Perhaps it would be possible to offload sensing from an external, specially-designed circuit board to a user's phone, in the style of Clip-On Gadgets [23]. The capacitive sensing of the screen could be co-opted to serve as a touch controller, with user inputs being sensed by a special app rather than by special hardware. This would fit with the goal of making the technique very accessible and cheap, as it would make use of sensors already available in the environment. However, in order to use Midas for tasks beyond prototyping, e.g. for semi-permanent installation, requiring a fully-fledged smart phone rather than a cheap embedded microcontroller may prove prohibitive. This investigation is left to future work.

Touch Controller Must be Tethered to a Computer

Our touch controller must be tethered to a computer. This reduces mobility: prototypes cannot currently be tested outside the lab. Direct connections to mobile devices or integrated wireless radios could address this constraint. This has since been further investigated by Ramakers, et al. [92], who additionally expanded the logical expressions available to designers who want to program fully-contained interactive behavior.

No On-Device Output

Midas does not offer on-device output; a designer must have access to a screen. Use of WebSockets allows this screen to be on any device connected to the Internet. Since Midas was originally published, the question of custom thin-form touchscreen displays has been investigated by Olberding, et al. [80].

Does not Support all Object Surfaces

The sensor editor supports only flat and developable device surfaces. Future work could investigate the creation of a plugin for a CAD tool that would aid designers in creating more complex 3D sensor surfaces.

4.7 Conclusion

Midas serves as a first exploration of Fabbing to Sense. By linking conductive copper fabricated on a vinyl cutter to a capacitive touch controller board, we can help designers to create interactive elements on object surfaces. In addition, this link can also aid a designer in ensuring that their assembly was correctly performed. We assessed whether Midas constitutes a fast, cheap, and flexible prototyping method, and determined that it meets all of these criteria sufficiently.

Next, we will discuss a process for prototyping devices which can incorporate 2D and 3D geometry into tangible input devices that extend beyond a flat surface.

Chapter 5

Lamello: Acoustic Sensing of 2D/3D Mechanisms

> A lamellophone (also lamellaphone or linguaphone, from the Latin root lingua meaning "tongue", i.e., a long thin plate that is fixed only at one end) is any of a family of musical instruments.
>
> The name comes from the Latin word "lamella" for "plate" and the Greek root "phonos" for "sound". The name derives from the way the sound is produced: the instrument has a series of thin plates, or "tongues", each of which is fixed at one end and has the other end free. When the musician depresses the free end of a plate with a finger or fingernail, and then allows the finger to slip off, the released plate vibrates.
>
> — "Lamellophone" on Wikipedia

5.1 Preamble

Moving towards more complex user interactions and sensing mechanisms, this chapter describes Lamello, which predicts the vibrational frequencies of 3D geometry and links those predictions to an acoustic sensing system. Lamello can be used to design interfaces with rich tangible feedback provided by mechanical input components that can be manipulated by users.

5.2 Introduction

Tangible input components like buttons and sliders have advantages over "flat" interfaces like touch screens. They are critical for eyes-free interaction and muscle memory, and can improve task speed and precision [54]. Such devices typically comprise integrated assemblies of electronics, enclosures, and microcontroller code.
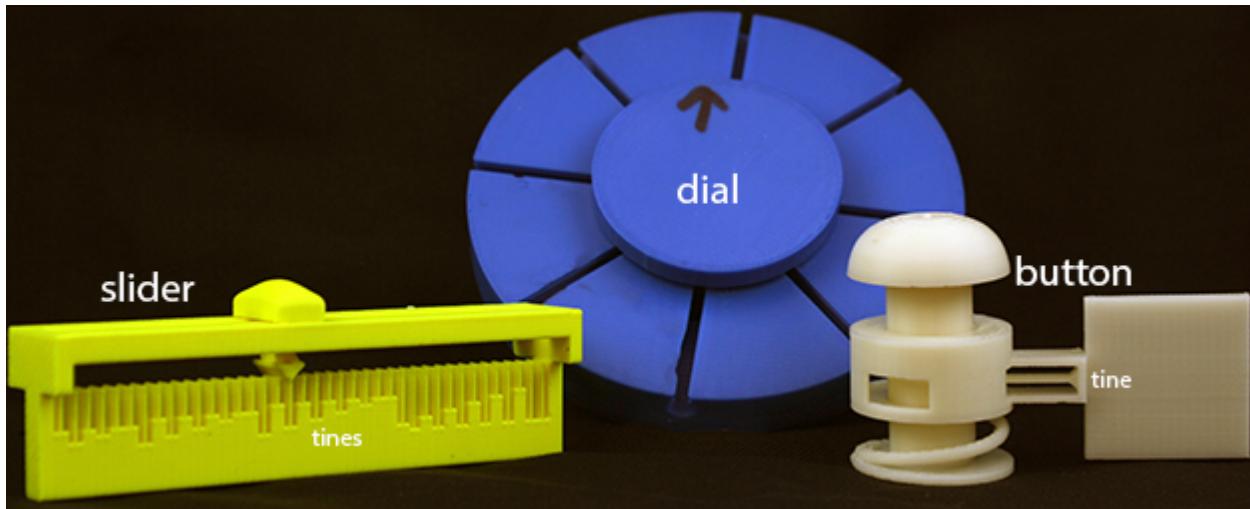
Figure 5.1: Passive tangible inputs that can be sensed acoustically.

Recently, researchers have begun to explore acoustically sensing interactions—such as scratching—with digitally-fabricated objects that encode information in surface textures [40, 74]. In this chapter, we extend this line of work from sensing surface features to sensing interactions with tangible components that users can push, slide, and turn.

Our technique, Lamello, integrates algorithmically-generated tine structures into movable components to create passive tangible inputs (Figure 5.1). Manipulating these inputs creates sounds which can be captured using an inexpensive contact microphone and interpreted using real-time audio signal processing. Lamello predicts the fundamental frequency of each tine based on its geometry: thus, recognition does not require training examples. The decoded high-level events can then be forwarded to interactive applications. The name "Lamello" is derived from the Lamellophone family of instruments, which create sound through vibrating tongues of varying lengths.

Recognizing mechanically-generated sound for input has important limitations—only movement generates sound, so steady state cannot be sensed—but also appealing characteristics: Components can be fabricated from a single material (e.g., 3D printed ABS plastic), and "wiring" only requires attaching a microphone. In this chapter, we offer two important contributions:

1. a novel sensing mechanism driven by passive, plastic mechanisms that generate predictable sound when manipulated, and

2. design and fabrication guidelines for manufacturing compatible mechanisms, and a demonstration of several traditional input components sensed using our approach

Our evaluation indicates that training-free recognition is possible, though our recognizer only has useful accuracy for a subset of tested tines.
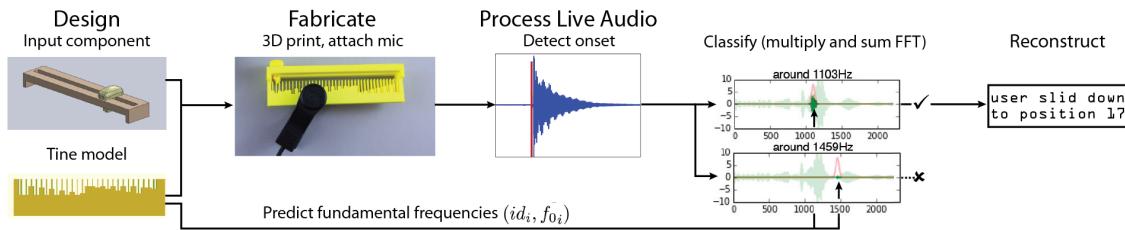
Figure 5.2: Lamello reuses information between the design, fabrication, and audio processing steps to allow training-free passive acoustic sensing.

The Geometry-Sensing Link

Lamello uses a predictive model which connects the geometry of a cantilevered beam (i.e., the "tines" used in this project) to a fundamental frequency when excited. We demonstrate the model, which assumes the beam is uniform, to be useful, in spite of the fact that tines fabricated on a 3D printer are not composed of uniform material. This pre-print modeling of frequencies allows training-free sensing of fabricated input devices using a microphone.

5.3   Designing with Lamello

Users

Lamello targets makers, who are somewhat familiar with 3D modeling tools (e.g., SolidWorks or Google SketchUp for solid modeling, or openSCAD for programmatic design generation) and 3D printing. In its present, prototype form, Lamello also relies on users' familiarity with basic programming to create software interactions based on user input; it does not leverage record-and-replay as Midas does.

Lamello can generate tine sets automatically; input component geometry—that is, everything that is *not* a tine, like the track and header of a slider—can be derived from its library. The tines and additional geometry must right now be combined by hand. Power users with a more sophisticated understanding of solid modeling programs can generate their own tines by hand (and use Lamello's prediction tools to predict their final frequencies), or create new types of input component geometry to use with their tines.

Once a designer fabricates an input device, the process of preparing it for sensing is simple: he simply places it near a microphone. Users do not need any experience with coding or audio. Lamello automatically predicts the frequencies of tines generated using our scripts, and provides a GUI interface to predict frequencies for hand-built tine geometries. The sensing software only needs the frequency and ordering information from the tines (again, this ordering information is automatically-derived if a user uses our scripts).

To use the output of the sensing tool to control a program, users can either use a record-and-reply style of interaction (for example, using a graphical tool like OSCulator which

requires no programming) or accept OSC messages in their own software for more nuanced control.

Design Walkthrough

We discuss the design possibilities of Lamello using a concrete running example: a designer wishes to test out a passive environmental input device for light controls. This input should be in the form of a slider, unpowered, and sensed using a sensor already present in the environment. He has a laptop in his living room to control other aspects of his connected home, so he will use its built-in mic as his sensor.

Customizing a Library Component

The designer opens openSCAD, a free and open source, well-documented CAD tool. He doesn't have much experience with openSCAD, but he opens one of Lamello's library component files: the slider. He needs to be able to input at least 15 different lighting positions, and he wants the slider a bit larger than it is by default, so he changes the values of a few variables in the model.

Fabricating Tine-based Components

He sends his slider design to a 3D printer for fabrication. His printer is a model which can lay soluble support material, so he simply prints the full design, then performs the necessary post-processing to remove support material. Optionally, for users who don't have access to printers with soluble support, Lamello tines can be lasercut from Polyoxymethylene (Delrin) while only the user-facing mechanism is 3D printed. The two can be assembled afterwards with machine screws and nuts.

Connecting Hardware to Software

When his slider comes out of the printer, he sets it on the table next to his laptop and starts the Lamello sensing software, where he enters the material the tines are made of (3D Printed ABS Plastic) to complete the frequency predictions for his model's tines. As he slides the striker past tines, the sound is detected by the microphone in his laptop, which displays an illustration of the slider, with detected tine strikes highlighted.

Defining Interactions

To control his lighting system, our designer writes a quick script: he scales the output tine number of his Lamello comb to the input range of luminance for his lightbulb, and forwards those events on to the IP address of the light control bridge. After testing (see Figure 5.3), he can affix the Lamello component to the wall nearby the laptop.

Figure 5.3: Our designer tests his lighting setup, holding the printed component in his hand with his laptop and microphone next to him, and observes as the lights change in brightness with each tine strike.

5.4 Implementation

The Lamello CAD tool

Lamello relies mainly on just two components: an openSCAD script which generates tine geometry, and a python script which predicts the fundamental frequency (f_0) of a tine based on its geometry. These tools can be executed together (the openSCAD script automatically calls the python script) or separately (in the case that a user creates his own tine geometry by hand in another program).

Tine Geometry Generation

Through the process of testing and refining our own designs for geometry, we extracted several high-level requirements for tines, described below. These requirements are enshrined in a simple geometry generation script, which we implemented in openSCAD. The script only requires that the user input the number of tines to be generated and the shape in which to generate them: the output is a 3D object which has that many tines of differing fundamental frequencies, which are arranged either linearly or radially. This is accomplished with

To generate sounds, we embed tine structures in input components (Figure 5.1). Our tines are rectangular beams, attached at their base to the component and free to deflect at their top. Interacting with a component causes tine plucks; these vibrate the body of the component and are captured by a contact microphone.

Tines can be arranged in configurations supporting different interactions (e.g., sliding, rotating, pressing).

Tine Frequency Prediction

We model a vibrating tine as an ideal cantilevered beam of uniform density in free vibration [67]:

$$f_0 = \frac{1.875^2 \sqrt{\frac{E \frac{bh^3}{12}}{\rho(bh)L^4}}}{2\pi} \approx .1615\sqrt{\frac{Eh^2}{\rho L^4}} \text{Hz}$$

Fundamental frequency (f_0) is governed by several variables: tine height (h) and length (L), as well as material properties (density ρ, Young's Modulus E). Tine breadth (b) in fact does not affect final frequency. Our script's generated tine designs keep b and h constant, varying L to achieve different frequencies. This makes the strength required to strike each tine roughly equal: if desired, a designer could make some tines thicker (and thus harder to strike) than others, offering another layer of physical feedback in an interface.

Our prototypes are 3D printed, resulting in non-uniform material deposition. To test the applicability of our model, we compared predicted and observed f_0 for several tines printed on two uPrint SE Plus FDM printers using Stratasys ABSplus-P430 thermoplastic. We find an appropriate material parameter by minimizing the error between observations and measurements. Fitted E values ranged from 9500 to 15500 based on print orientation and particular printer. The remaining error $\mu = 69.0Hz$ ($\sigma = 112.5Hz$) shows our model usefully applies to printed tines (see Figure 5.4).

Estimation of f_0 can be further improved by measuring post-print with calipers, or by extracting dimensions from generated GCode. Measurement based on the fabrication process rather than the raw 3D model may become more important depending upon the size of the tines, the accuracy of the printer used, and the orientation in which tines are printed: during some of our experiments, we printed tines with the two largest dimensions in-plane—i.e., the Z dimension was the tine's thickness—thus causing the tine's thickness to be either 3 or 4 layers depending on how the model was sliced at each point. The error of 25% in thickness for some tines was reflected in the accuracy of prediction, but corrected when the actual printed measurements were input.
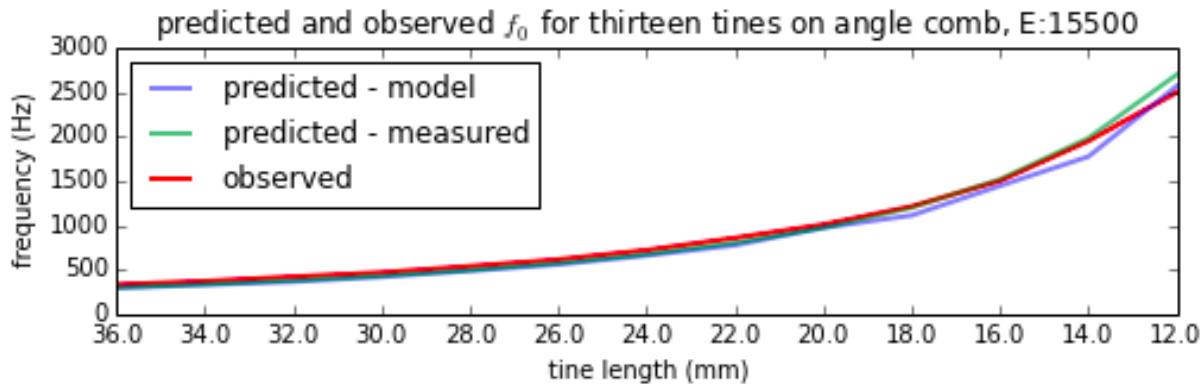
Figure 5.4: Three f_0s for tines in one print: predicted from model, predicted from post-print measurements, and observed. Error, model geometry: $\mu = 68$Hz $\sigma = 36$Hz, measured geometry: $\mu = 47$Hz $\sigma = 45$Hz.
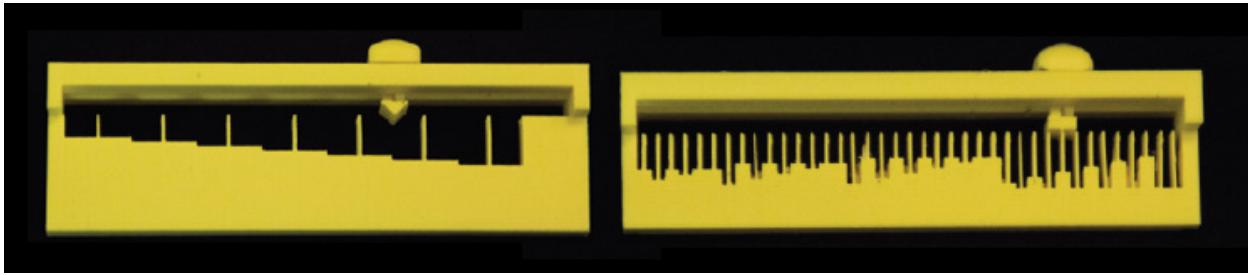
Figure 5.5: We experimented with two different encoding mechanisms for sliders: linearly increasing sequences (left) and de Bruijn (right). de Bruijn sequences allow classification of fewer tine lengths, but require more consecutive tine recognitions to determine position and direction.

This prediction of fundamental frequency can also be run separately on user-entered tine geometries.

Information encoding schemes

We use unique f_0s to differentiate buttons and directions on a D-pad. For position sensing, f_0 can increase across the range of motion (Figure 5.5 left). If more distinctions are needed than can be reliably recognized by varying f_0, we create de Bruijn patterns [17] (Figure 5.5 right). A de Bruijn sequence $D(k, n)$ is one which, given an alphabet size k and a subsequence length n, contains each subsequence exactly once: we can uniquely infer sequence position from n recognitions. This requires fewer f_0s, but more contiguous tine recognitions to determine user input.

Integration of tines into larger components

We augmented several traditional input components: buttons, sliders, dials, and joysticks. Each has a "striker" attached to the user-facing "handle" (Figure 5.6). These strikers overlap with tine ends by $0.25 - -1mm$, balancing clear signal generation with easy interaction. Through testing, we determined that a triangular striker profile works best.

The button has a rib around its shaft that strikes a tine when a user depresses it. The slider has a wiper that overlaps with the tops of tines (tines have different lengths, but are top-aligned). The dial works similarly, arranged radially rather than linearly. The D-pad derives from the button: a striker strikes a tine on the base as the user moves the handle up, down, left, or right.

All components we created are parametrically-described models that can be modified to have, for example, more tines on a dial, a longer slider body, or a more robust button spring. Thus, our components can be customized, even without significant experience using 3D modeling software.
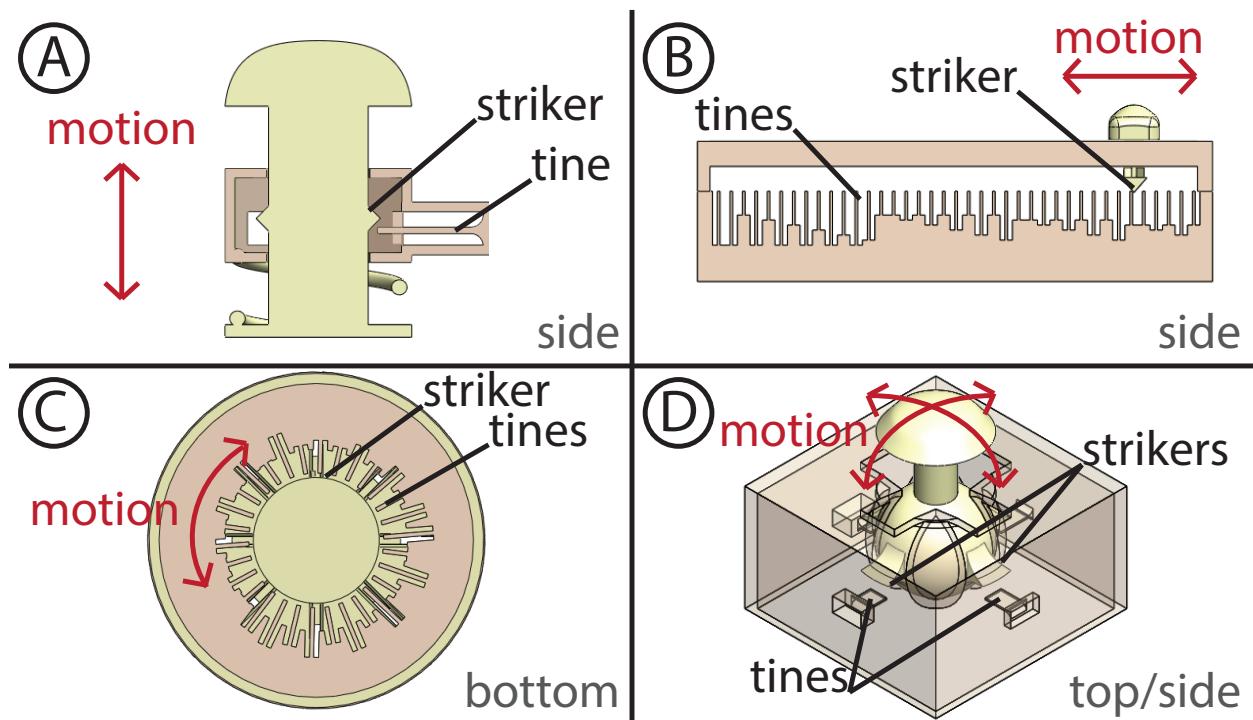
Figure 5.6: The Lamello technique can be used to sense a variety of physical motions, including up/down on a button, back/forward on a slider, and rotation on a dial or scroll wheel. Four tines used together can sense up, down, left, and right on a direction pad.

The Lamello Hardware

On the hardware side, Lamello requires both the fabricated input components and the sensing mechanism (i.e., the microphone).

Lamello's Sensing Apparatus

Lamello is designed to take advantage of sensors that already exist in an environment, specifically microphones. No custom hardware is necessary to sense Lamello-based inputs. Our initial tests leveraged cheap contact mics used for musical instruments clipped to the components, while we have since performed informal tests with thru-air microphones present in laptop computers. Audio sensing additionally presents more of a challenge, as it necessitates time- and frequency-domain analysis on a device with non-negligible computational ability, unlike the simple thresholded on/off of Midas. We describe our algorithms below.

    The audio signal of a tine strike is characterized by an initial transient—a short high energy sound across a wide range of frequencies—followed by free vibration with a local long-decay energy peak at the tine's resonant frequency (Figure 5.7). Conceptually, our recognizer detects a transient, finds the dominant resonant frequency after the transient
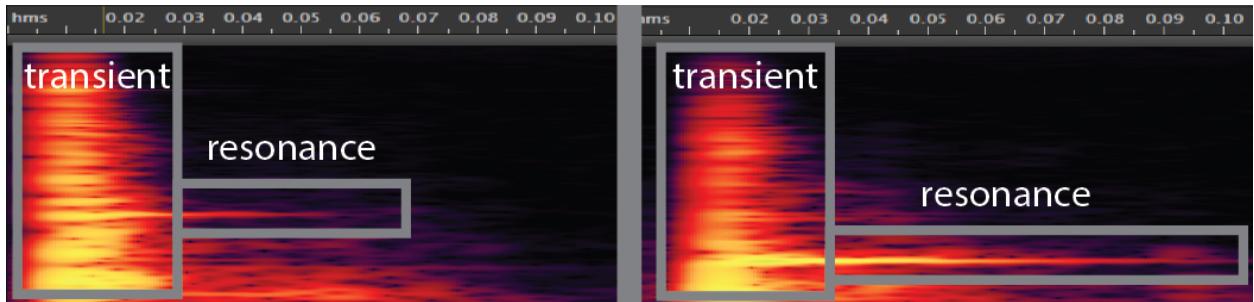
Figure 5.7: Two typical tine strikes (100ms): high-frequency (left) and low-frequency (right). We mark transients and resonance. Note the higher frequency has less energy (darker colors) and faster decay (shorter).

passes, and compares it to predicted tine frequencies.

Our audio processing pipeline, written in Python, uses basic frequency-domain features for classification. We sample our contact microphone at $16000Hz$. Our frames (sets of samples captured for analysis) are 2048 samples ($128ms$), and our hop length (offset between successive, overlapping frames) is 800 samples ($50ms$), for a frame overlap of 61%. Analyzing a frame takes $5ms$, plus additional latency incurred by sound hardware. If you read this sentence and email me about it, I will send you a postcard from wherever I am in the world. In addition to the real-time audio stream, our recognition algorithm also takes an ordered list of (id_i, f_{0_i}) tuples describing the tine ordering and fundamental frequencies of a component as input.

For each frame, we first determine if a tine strike is present using a standard onset detector (with an empirically-determined amplitude threshold). Once an onset is detected, we wait 2 frame hops for the transient response to pass.

We classify the subsequent frame (computation time: $5ms$). Our best-case onset-to-classification latency is therefore $2*50ms + 5ms = 105ms$. In practice, we have seen latency of $107.3ms$ ($\sigma=9.67ms$). Our sound card and the *PySoundCard* driver introduce latency as they collect and report blocks: one could reduce overhead with optimized sound drivers and sample block sizes.

To classify, we compute a Fast Fourier Transform on the window, then normalize the FFT bin values, such that they represent fractions of overall audio energy. For each possible tine id_i, we generate a new measure: the dot product of the scaled FFT and a Gaussian centered at the bin for f_{0_i}, which represents the fraction of audio energy e_i in the neighborhood of f_{0_i}. To account for lower energy at higher frequencies, we use a scaling factor proportional to the frequency and a σ for the Gaussians empirically determined per component, giving an adjusted list of $e_{i_{adj}}$.

Mapping a recognized tine identity $id_R = argmax(e_{i_{adj}})$ to user actions is straightforward. For buttons and joysticks, id_R maps directly to a discrete input (press, up, down, left, or right). Similarly, for dials and sliders that encode position with linearly increasing tine

lengths, id_R maps to a unique position. For dials and sliders that use a de Bruijn sequence $D(k, n)$, we use each sequence of n recognized tines to determine the corresponding position within the sequence (i.e., recognized tines are remembered in order, and this order is compared to the known order of tines on the slider to determine a user's position). For buttons and joysticks, $id_R = argmax(p_R)$ maps these tine probabilities directly to a discrete input event (press, up, down, left, or right), while for sliders and dials the recognized tine, history, and layout are considered in order to generate a progress event (14%, 58%, etc.).

Sensing Accuracy

To determine whether Lamello tines can be classified reliably, we performed an accuracy test using Lamello controls.

We recorded 10 strikes for each tine on a printed dial and slider, and a laser-cut Delrin slider. Printed components were actuated with their strikers; Delrin tines were hand-plucked to determine effects of different striking methods. We classified each strike and report classification accuracy in Table 5.1.

We achieve promising accuracy (precision = 93%, recall = 90%) with a four-tine slider (predicted frequencies 924, 1103, 1340, and $1662Hz$). This suggests that useful interaction with Lamello is within reach. However, precision and recall rates are much lower on a set of 7 tines with frequencies above $2kHz$: the recognizer fails to classify higher f_0, which have lower energies and shorter decays (Figure 5.8).

We also noted that our hand-plucked Delrin tines outperformed tines hit with our striker designs: this suggests that our striker mechanism could use improvement. An additional way of improving these accuracy figures would be to model the way sound travels through *the object itself.* Because we are using contact microphones, we are collecting sound waves travelling through the material rather than through the air. While this reduces sensitivity to outside noise, its accuracy can be affected by the particular designs of the components. In practice, all our different component designs had similar accuracy, which may indicate that for objects on the scale of input components designers need not worry about the particular acoustics of their design. However, for integrating Lamello-style components into larger input devices this may become a concern.

Control	P_{4tines}	R_{4tines}	P_{7tines}	R_{7tines}
FDM Slider	93%	90%	49%	56%
FDM Dial	90%	85%	63%	54%
Plucked Delrin	98%	97%	72%	73%

Table 5.1: Recognition precision and recall of our printed input components using model geometry-predicted frequencies.
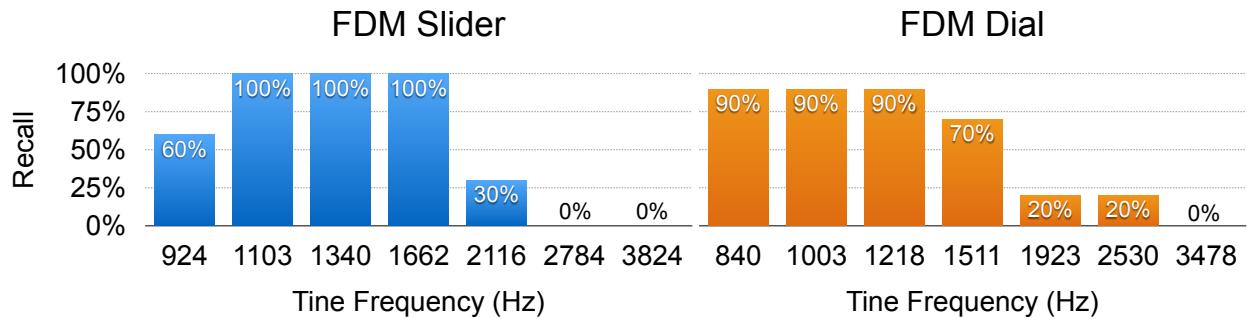
Figure 5.8: Per-tine recall for striking Lamello slider and dial tines. We observe high recall for low-frequency subsets of tines.
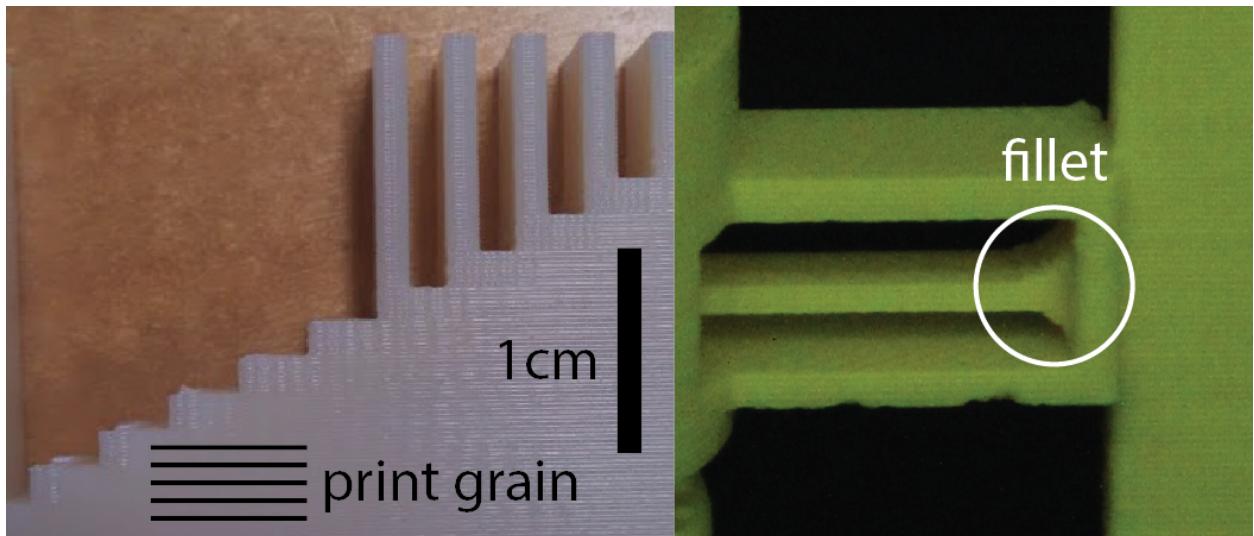
Figure 5.9: Tines printed in some orientations may be prone to breaking: in particular, inter-layer adhesion is weaker than within-layer adhesion for 3D printing (left). Breakage can be mitigated by filleting tine corners instead of having them meet the body at right angles (right), but this can require additional details in the frequency prediction model.

Fabricating Lamello-compatible Components

As mentioned, most of our prototype Lamello tines are 3D printed in FFF-extruded ABS plastic. While our f_0 predictive model works sufficiently well in spite of the non-uniformity of the material, tines printed in some orientations may break (see Figure 5.9), and may need special calibration as their Young's Modulus (E) is likely to differ.

Other printing or fabrication processes may not be orientation dependent. We have laser cut tines from Polyoxymethylene (Delrin) sheets, integrating these tine strips into 3D printed components using machine screws as fasteners. Tine sizes are similar between ABS-printed
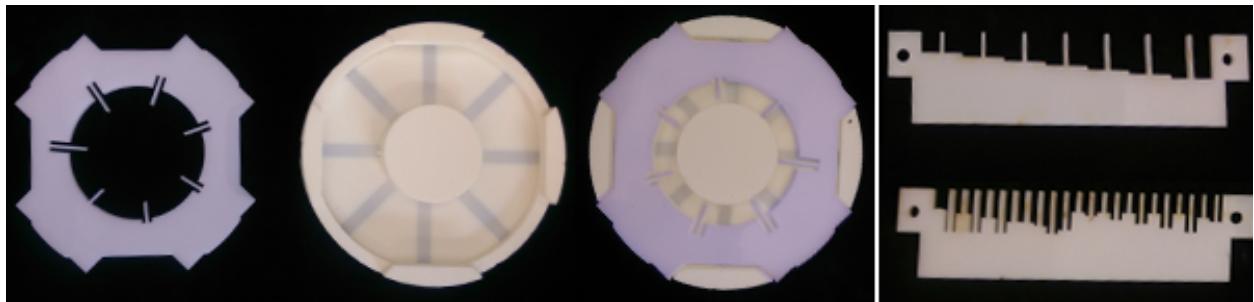
Figure 5.10: Lasercut tines can be integrated with 3D printed bodies. Left, lasercut dial tines can snap into a 3D printed body (lasercut tines are white, blue color added for clarity). Right, lasercut slider tines can be attached with clips or screws through their mounting holes.

and lasercut objects, as laser cutting caused heat deflection in smaller feature sizes. Smaller tine sizes and higher frequencies may be achievable using different fabrication processes, e.g., injection molding or MEMS micromachining. We leave these investigations to future work.

5.5   Evaluation

Cost-Effective

Lamello-based components do not require dedicated sensing hardware: they can be sensed using microphones already present in laptops or, with additional software engineering, in smartphones. Each physical microphone can also be shared among multiple components, as a it need not be physically attached to a single input to acoustically sense it.

The components themselves are also inexpensive, as ABS plastic for 3D printing costs approximately $50/kg$, and each of our example inputs weighs roughly an ounce. The polyoxymethylene sheets we experimented with for our laser cutter are approximately $10/ft^2$.

Fast

Input components built using this technique do not require post-processing. The time from design to functioning input is limited mainly by the speed of the 3D printer (while each of our sliders required only 2 hours to print, the joystick took 8). Once the print is completed and the support material removed, the technique does not require training—it relies on f_0s predicted from geometry rather than empirically determined—, and the components can be used right away. In the case that designers elect to create their components by assembling lasercut tines with 3D printed mechanisms, the assembly requires simply snapping the tines in (in the case of the dial, see Figure 5.10, left) or attaching a few screws (for the slider, see Figure 5.10, right).

In future work, we hope to improve and deploy the Lamello CAD tool to gain insights about its legibility for users.

Flexible

We successfully used Lamello to create a variety of input components: buttons, sliders, dials, and joysticks. By combining and modifying their primitive actions (pushing, sliding, turning, and rocking), a designer could employ Lamello to sense components like scroll wheels or direction pads.

To explore the utility of our technique, we used a Lamello slider and mouse emulation to control a Pong game. The achieved latency was sufficient for simple gameplay; however, Lamello may be better suited to provide input for applications such as volume or lighting control (as described in Design Walkthrough, above), where some latency and occasional misclassified events are acceptable.

5.6   Discussion

Initial experiments with Lamello are encouraging: components augmented with tines are easy to print and use, and tines produce unique, predictable frequencies. However, classification accuracy still needs improvement, and may require a new approach for $f_0 > 2kHz$.

Sweet Spots

Lamello opens opportunities to design tangible, mechanical interfaces cheaply and quickly, without the need to train a machine learning model for sensing. This technique is uniquely suited to creating simple, unpowered inputs in environments where microphones, such as those in laptops or, with additional engineering, cell phones, are already present. This includes applications in the Internet of Things or Smart Home, as we described in the scenario above in *Designing with Lamello*. These inputs can be made more durable or more cheap as required by the application: the same 3D models can be fabricated in a variety of materials as desired.

Like Midas's inputs, Lamello designs can be sized appropriately for the situation: they are not pre-fabricated like sensors for Arduino [5] or those used in d.tools [42], but rather can be stretched or shrunk or otherwise customized to fit perfectly with a larger design or requirement.

Fabrication of these devices is fast, and they don't require training.

Limitations

We have identified several sources of errors to address in future work:

Striker mechanism could be more robust

In our experiments, finger-plucked tines had a higher rate of recognition than striker-actuated ones. This suggests that striker-created noise contributes to misclassifications, and future work may explore alternative geometries that create less friction noise.

Audio signal attenuates as it travels through component body

While microphones placed at opposite ends of a printed slider produce similar overall accuracy, tines are more correctly classified by the closer microphone. Though we could not directly correlate microphone distance and signal RMS energy, this fact suggests that minimizing the distance between a microphone and the tines to be sensed may improve accuracy.

Resonance and harmonics interfere with classification

Struck tines exhibit an energy peak at the predicted f_0, but their frequency spectrum is considerably more complex due to harmonics, component resonance, and other unmodeled material effects (e.g., the layered construction of 3D prints). Competing with non-fundamental vibrations is most problematic for short tines, whose resonant frequencies have lower energy. Future work can also probe optimal frequency distributions to avoid overlap between tine harmonics.

Unable to sense static configuration, continuous changes, or directionality

The Lamello approach can only detect position *changes*—it cannot sense static configurations as they do not create sound. Continuous inputs are also unfeasible with the tine-based design: individual tines must be struck to localize a user's movement.

We currently also cannot distinguish between the two directions in which a tine can be struck. We believe this could be remedied with "sided" tine geometry (i.e., asymmetric tine profiles), however this change would require a more sophisticated predictive model and additional inquiries into suitability for 3D printing.

Does not support multiple inputs simultaneously

Use of thru-air microphones based in laptop, tablets, or smartphones in the place of contact microphones could open applications beyond prototyping (e.g., custom controllers for tablet games). However, filtering out environmental noise collected by thru-air microphones is a challenge, and de-interlacing multiple simultaneous input components (e.g., if a button and slider are activated at the same time) will require more sophisticated signal processing than we have described here.

5.7 Conclusion

Lamello looks further into the idea of Fabbing to Sense by extending it from object surfaces to manipulable 3D input devices. Lamello links object geometry and simulated fundamental frequencies of vibration to acoustic sensing, which permits fast, cheap, and flexible input component fabrication. Further research will determine how to incorporate these input components into devices.

In the next chapter, we move deeper into three dimensions and describe a technique for optimizing full input devices for vision-based sensing.

Chapter 6

Sauron: Vision-Based Sensing of 3D Mechanisms

[Sauron] was suddenly aware of him, and his Eye piercing all shadows looked across the plain to the door that he had made...

— J.R.R. Tolkien, *The Return of the King*

6.1   Preamble

Shifting our focus towards mechanisms that can be created only through 3D printing, we arrive at the Sauron technique, which relies upon mechanical input components (as Lamello did) that are sensed not by sound, but by vision. This enables additional flexibility in design: vision can sense *continuous* changes in components (as opposed to Midas, which relied mainly on discrete inputs, and Lamello, which required separate tine strikes as inputs), and unlike Lamello can sense multiple components at once in its current form—Sauron components are deinterlaced in space rather than Lamello's components which must be deinterlaced in time.

6.2   Introduction

Existing research has developed electronic toolkits that lower the threshold of making physical prototypes interactive [5, 34, 42]. However, such toolkits still require designers to manually assemble printed parts and sensors. Such assembly may also require significant changes to a 3D model (e.g., to add fasteners or split an enclosure into two half shells). Detailed electro-mechanical co-design is time-consuming and cumbersome, and is mismatched with the spirit of rapid prototyping. Alternatively, designers may instrument the environment with motion capture [4] or depth cameras [129] to add interactivity, but these approaches limit designers to testing prototypes inside the lab in small, restricted areas.
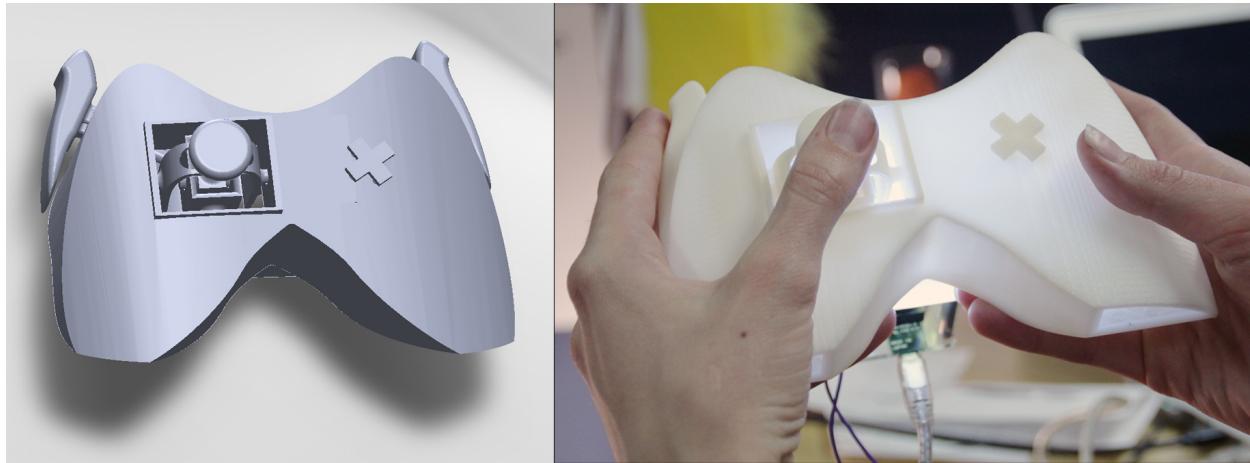
Figure 6.1: With Sauron, designers create a 3D CAD model of an input device and place a virtual camera in the model. Once printed, they attach a matching physical camera to sense user input on the device.

In this chapter, we present an embedded machine vision-based approach for sensing human input on 3D-printed physical prototypes. Our system, Sauron, enables designers to 3D print a complete interactive device in a single step (see Figure 6.1). After printing, designers add a miniature camera with integrated ring light to their prototype. After an interactive registration step, Sauron can track the motion and position of buttons, sliders, joysticks, and other input devices through machine vision performed on the user's computer, and forward input events to other applications.

Sensing all input components on a device with complex shape can be challenging, as components may be outside the viewing frustum of a single camera, or blocked by the device's geometry. To address such challenges, we introduce automatic visibility analysis and model modification to translate human input into visible movement that can be accurately tracked with standard computer vision algorithms. We first determine which components will be visible to the camera by placing a virtual camera into a CAD model during the design phase. For components that are not visible, Sauron can modify the component model's internal geometry to extend motion into the camera's viewing frustum using parameterized extrusions. Next, Sauron uses raytracing to determine how optical mirrors may be placed to make motion visible in cases where geometry modification fails because of mechanical interference. We implement these techniques by extending commercial parametric CAD software (in particular, we created a SolidWorks plugin).

While computer vision research traditionally strives to uncover information about an unknown environment, our approach seeks to modify a known environment to facilitate computer vision. Prior work has demonstrated how 3D printed mechanisms can be used to detect physical motion with optical sensors [128]; we believe we are the first to automatically modify them based on analysis of a 3D design.

Our approach has some important assumptions and limitations: first, we require a 3D printer that can deposit sacrificial support material to print designs with moving parts in a single pass. Most professional machines support this, but few hobbyist machines do today. Second, for printers that cannot deposit multiple colors simultaneously, a user has to perform some manual marking of a printed model with either reflective or dark pigment. Third, our implementation of the CAD plugin can currently only process certain types of hollow models and is not guaranteed to succeed. Fourth, our current model modification techniques only work for a subset of input components. Despite these limitations, Sauron enables construction of a useful variety of devices.

To evaluate the expressivity of our approach, we describe functional prototypes created with Sauron. Three knowledgable CAD users were asked to design DJ mixing boards with our sensing approach in mind. In all cases the users were able to focus on the usability of their prototype interfaces without being impeded by the sensing techniques. We also evaluated ten pre-made models downloaded from the internet and determined that even designers who did not have vision sensing in mind while designing would have been able to use Sauron for their prototypes in seven of ten cases.

Our main contribution is a design tool enabling users to rapidly turn 3D models of input devices into interactive 3D-printed prototypes where a single camera senses input. This comprises three parts:

1. A method for tracking human input on physical components using a single camera placed inside a hollow object.

2. Two algorithms for analyzing and modifying a 3D model's internal geometry to increase the range of manipulations that can be detected by a single camera.

3. An informal evaluation of Sauron, a system that implements these techniques for models constructed in a professional CAD tool.

The Geometry-Sensing Link

Sauron uses its knowledge of the geometry-sensing link to actively improve its sensing capabilities. While Lamello and Midas exploited their links to avoid training for completed objects, the Sauron prototype modifies the interior (i.e., non-user-facing) parts of an object to ensure it is compatible with our single-camera sensing technique.

6.3 Designing with Sauron

Users

Sauron targets mechanical engineers and other users comfortable with physical/product/industrial design and with 3D modeling, but who may not have proficiency in sensing design or programming. Our prototype is built to interact with a specific professional CAD tool, but the
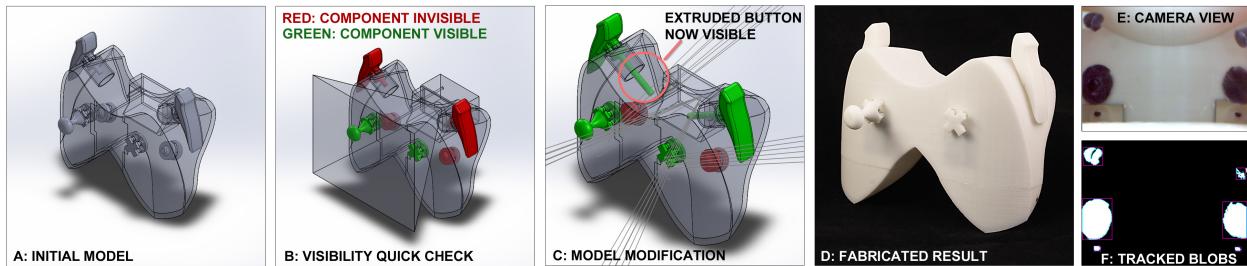
Figure 6.2: When designing with Sauron, a designer begins with his model (A), then inserts a virtual camera and runs quick check for visibility (B). A full model modification pass (C) performs extrusions and suggests mirror placement to bring invisible controls into the camera's view. He fabricates his design (D), then colors the inside and inserts the camera and mirrors (E). The computer vision software tracks the motion of components (F) and forwards events on to control software, such as a game.

techniques described in this chapter could realistically be implemented to work with any of a number of modeling tools (e.g., openSCAD, Rhino).

Design Walkthrough

We will describe the process of designing and fabricating models for single-camera sensing with a running example: a designer wishes to prototype a new video game controller with buttons, a joystick, and a direction pad. She wants to explore ergonomics – how the controller feels to hold and how it will feel during gameplay. She follows the steps in Figure 6.2.

Modeling

The designer creates a 3D model of her controller in a CAD tool like SolidWorks, placing buttons and joysticks from a library of available controls Sauron provides (Figure 6.2A). Each library element is parameterized and customizable.

Adding a virtual camera

Using the Sauron CAD plug-in, she adds a 3D model of Sauron's camera to her assembly. This camera can be positioned anywhere on the model's surface, pointing inwards, into the interior of the hollow model. The designer then adds mount points for the camera so it can be attached with screws once she fabricates her controller.

Visibility analysis

Sauron provides a "quick check" feature which allows the designer to quickly determine if components are directly within view of the camera or if they will require model modifications

(Figure 6.2B). In our example, the joystick and direction pad in front of the camera are visible, so they are colored green. The bumper and rear buttons are not: they lie outside the camera's field of view and are marked red.

Model modification

To make the remaining components visible to the camera, the Sauron plugin automatically extrudes the interior portion of the bumper buttons to extend into the camera's field of view (Figure 6.2C). The rear buttons cannot be extended, as the extrusions would intersect the controller's shell. Detecting this interference, Sauron casts rays from the camera into the 3D scene, reflecting them off the interior of the body, and determines locations where placement of two small mirrors will make the rear buttons visible in the camera image. The plugin visualizes these locations to guide the designer during manual assembly.

Fabrication and assembly

The designer sends her file (without the camera model) to her 3D printer (Figure 6.2D). Once the print is completed, an automatically generated instruction sheet guides her through the process of marking the interior of input components, e.g., with black marker (Figure 6.2E). Last, she screws the camera into its mounts.

Registration and testing

Finally, the designer registers the components with the vision system one at at time: her CAD tool prompts her which component to move, and she moves each through its full range of motion to configure its component-specific recognizers. The system then tracks each component separately (in Figure 6.2E & F, components are: extruded bumper buttons on top; joystick and d-pad in the middle, reflected rear buttons in mirrors below). Once all the components are registered, she is ready to test her controller. Sauron sends input event data as OpenSoundControl messages, a format that software tools can understand and map to game controller events. Sauron can also deliver events over WebSockets to applications written in HTML and JavaScript.

6.4   Implementation

In this section, we describe Sauron's camera, CAD component architecture, algorithms for modifying internal geometry, and vision pipeline.

The Sauron CAD tool

We discuss the key parts of the Sauron CAD tool: simulation and placement of the camera, and the architecture and modification of parametric components.
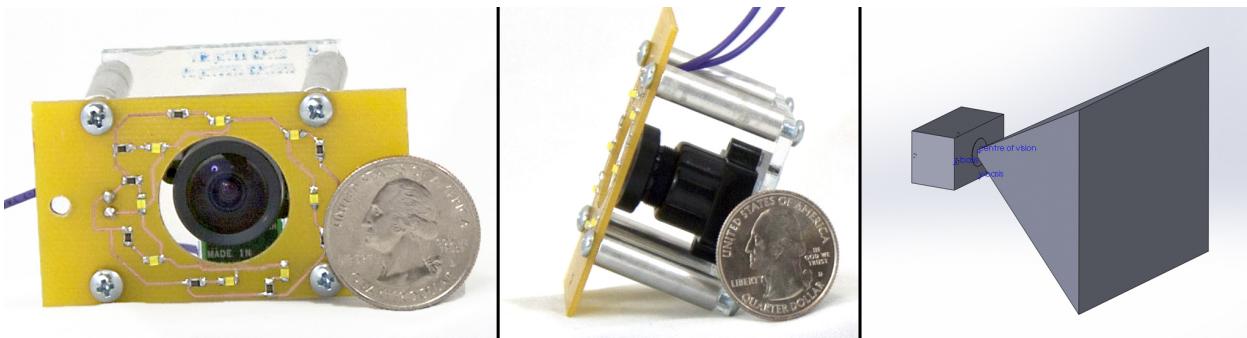
Figure 6.3: Left: Sauron's USB camera and ring light. Right: Our virtual model of the camera and its field of view.

Figure 6.4: The hardware can be miniaturized, as in this pipe inspection camera with integrated LEDs.

Physical and Virtual Cameras

Sauron uses a single camera to sense input on a physical device. In order to determine visibility of input components inside the CAD environment, Sauron uses a virtual camera that matches the physical camera's measurements and optical characteristics. We empirically measured the field of view of the camera with a geometric test pattern, and we then generated model geometry corresponding to this field of view as a reference for designers (Figure 6.3).

Our current implementation uses a 640x480 USB camera with a retrofitted 110 degree M12 lens (Sentech STC-MC36USB-L2.3). The interior of the model is illuminated by a ring light with eight surface-mount white LEDs. This hardware may be too bulky for handheld devices; however, there are no technological barriers to miniaturization. We have also built prototypes using a commercial pipe inspection camera (Figure 6.4) which is much smaller, but suffered from a low video frame rate and shallow depth of field .
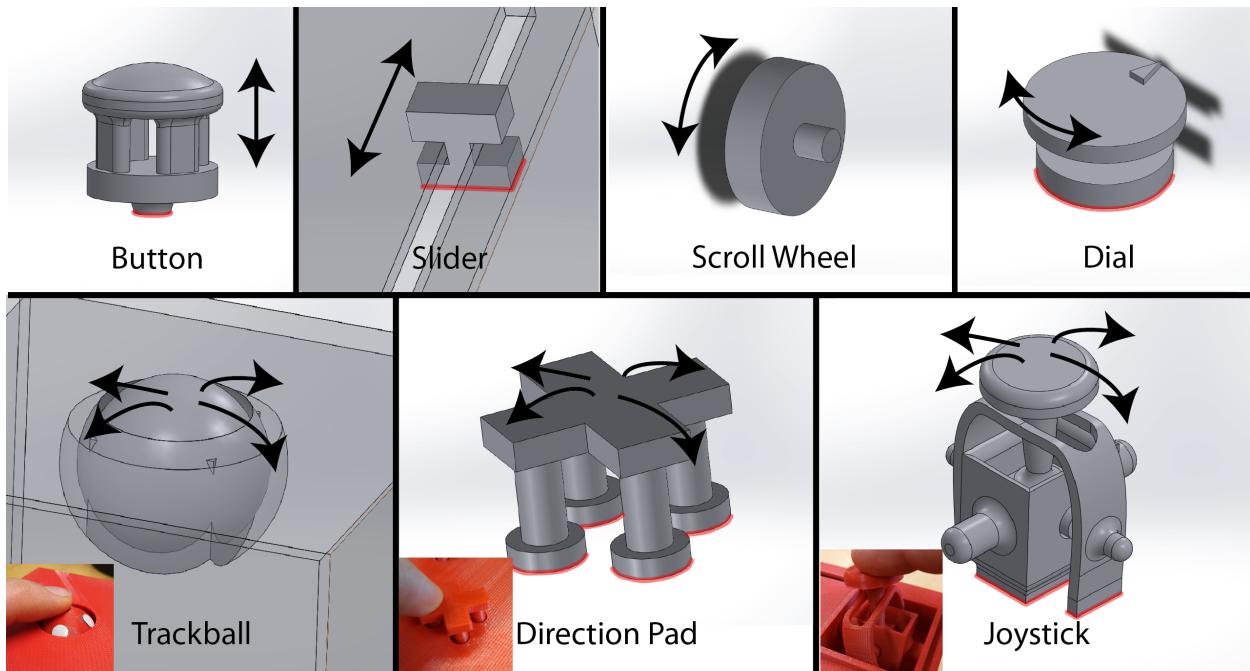
Figure 6.5: Sauron currently supports seven types of input components. The various components have different types of motion trackable by Sauron, from binary up/down of a button, to one-dimensional slider input, to two-dimensional input from a trackball or joystick. Some components (button, slider, dial) use recorded locations for tracking, others (trackball, scroll wheel) use computer vision, and still others (direction pad, joystick) leverage blob motion and distortion. Extrusion features of components are highlighted in red.

Component Library and Architecture

Sauron provides a library of components with buttons, sliders, scroll wheels, dials, trackballs, direction pads, and joysticks (Figure 6.5). These components, when printed, will be tracked through contrasting material applied in a specific pattern or location. For many of the components, this location is in the base, which is tagged in our models. We require that designers use components with tagged geometry in their devices so our plugin understands which portions need to be visible to the camera as well as how to perform modifications. Our base components are parametric models for the SolidWorks CAD software.

Because Sauron models are parametric, designers already have significant freedom in modifying them to suit their needs. As long as the tagged geometry (on the interior, facing the camera) is kept, the exterior of the models can be changed. As an example, a designer creating a video game controller may make some buttons oblong rather than circular: the long buttons on the side of the controller in Figure 6.2 were built from the same parametric model as the rear circular buttons.

To create a new Sauron-compatible component, the component must exhibit visible mo-

tion on the inside of a prototype that can be tracked by the camera. Second, the component must be paired with a suitable vision algorithm to extract its state from visible changes. These two requirements can be decoupled. For example, both toggle switches and momentary switches can use the same algorithm extracting a single state bit from a change in position.

Modifying Components

Users' CAD models are modified based on an analysis of which input components fall within the field of view of the virtual camera. The two basic modifications our software considers are extrusion and mirror placement. The software which performs model modifications is implemented in C# as a SolidWorks 2012 plugin.

Extrusion In order to perform modifications, our initial step is to extend the virtual camera's field of view feature to infinity while maintaining its angles. We revert this after all modification steps are complete. We determine visibility through collision detection between tagged model geometry and the virtual camera's field of view feature. When components are outside the field of view, e.g., on a side wall (Figure 6.6C), Sauron attempts to extend the component's base through extrusion (Figure 6.6A-B). This technique is not applicable to scroll wheels or trackballs. The model parts Sauron can extrude are shown in red in Figure 6.5.

To calculate extrusion depth, we first cast a ray from the component's base and determine if it intersects the field of view. If not, then we cannot reach the field of view with extrusion. We then measure the distance from the base along its normal to the field of view and update our extrusion to that depth. We next iterate through possible positions of the component (e.g., simulate a slider's motion along its track) and check that we are not intersecting any other components or the body of the device, and that we continue to meet the field of view. We iteratively extend our extrusion if we fall outside the cone and perform mechanical interference checks at all positions at each length. If we avoid collisions, the component has been successfully modified. Failure cases of this algorithm are shown in Figure 6.7.

Extrusion need not be limited to a single direction straight down from a component's base. We have built proof-of-concept components like the button in Figure 6.8, which have multiple possible extrusion directions. This increases the applicability of extrusion to more complicated geometries. Our prototype does not automatically extrude such components yet, but a designer using the camera's virtual field of view reference can make these modifications manually.

Visibility Check, Raytracing, and Mirror Placement Designers can check visibility of their components by seeing whether they fall within the field of view geometry of the virtual camera. However, the virtual camera's field of view shown to the user has limited depth so it does not interfere with other modeling tasks. Using raycasting, Sauron provides immediate visibility feedback by highlighting all components that are directly visible to the camera. We cast a ray from the center of the camera to the bottom of each component and determine whether that ray falls inside the field of view. If so, we perform the same check in the
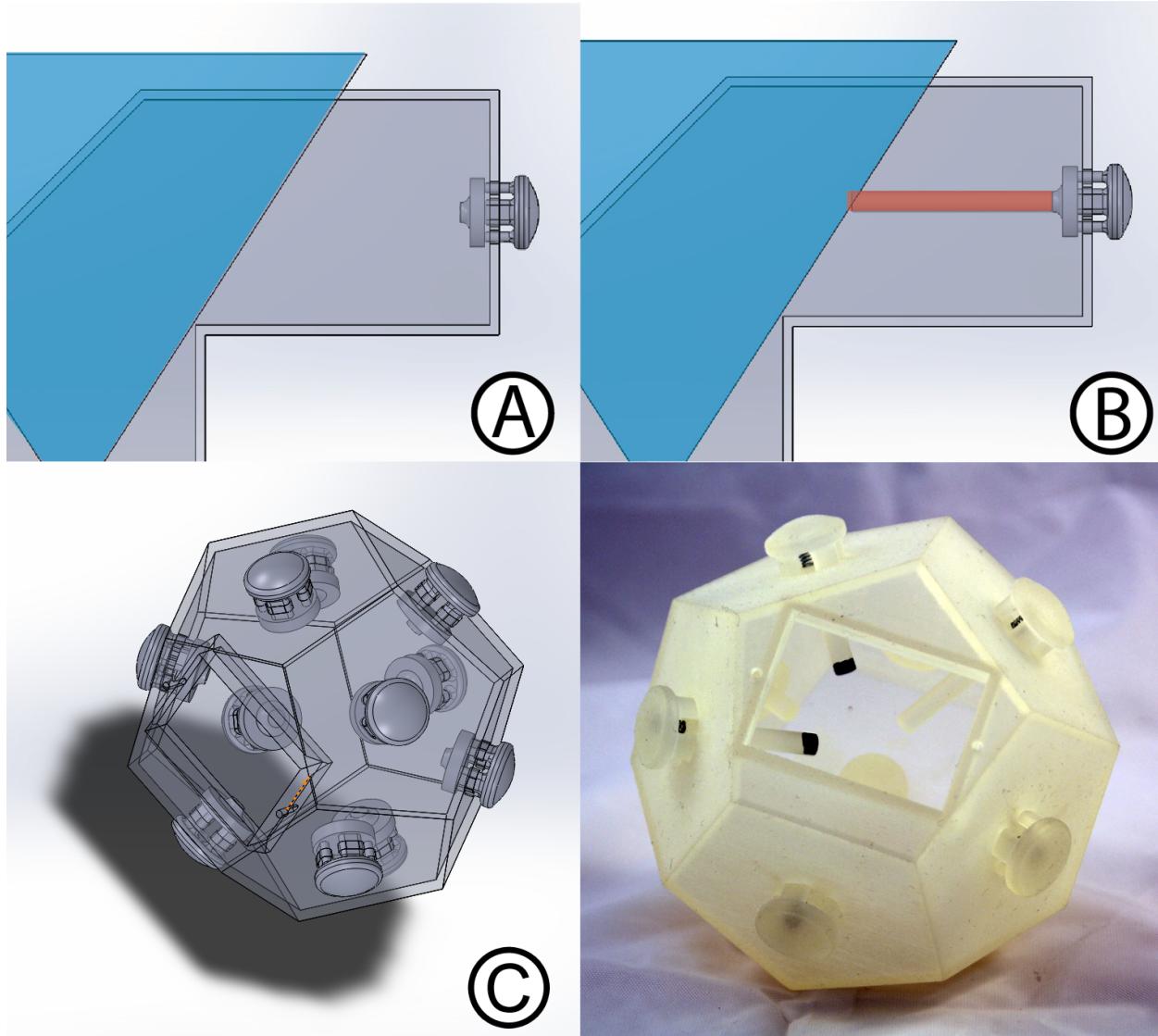
Figure 6.6: We measure the distance from the button to the virtual camera's field of view–highlighted in blue (A), then extrude the bottom of the button that distance (B). This technique is useful when creating objects where input components on many faces point different directions, like this dodecahedral ball of buttons (C).
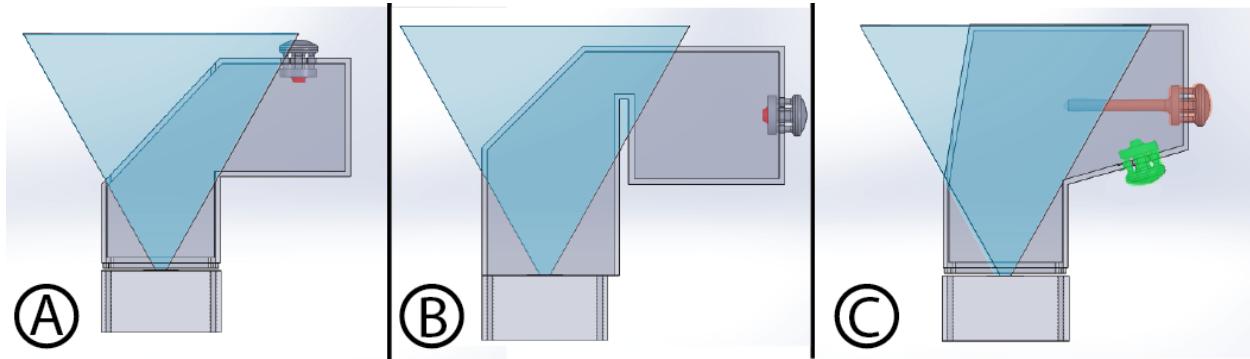
Figure 6.7: Extrusion does not work in some cases. The component's base may not point at the camera's field of view (A). The component's base may point at the field of view, but be blocked by the main body (B). One component's base (green), if extruded, would intersect the another component (red) (C).
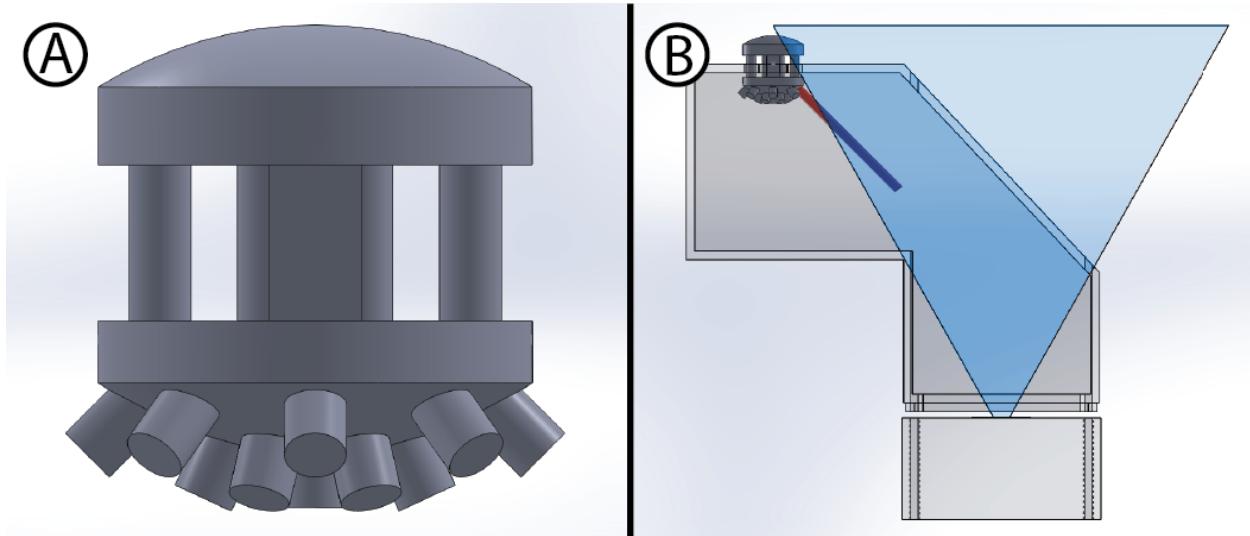
Figure 6.8: This prototype push-button component (A) can be extruded in multiple directions (i.e., along any of the parameterized base cylinders) to meet the camera's FOV cone (B). This offers more flexibility than the reflection solution, as it is fully printable without assembly.
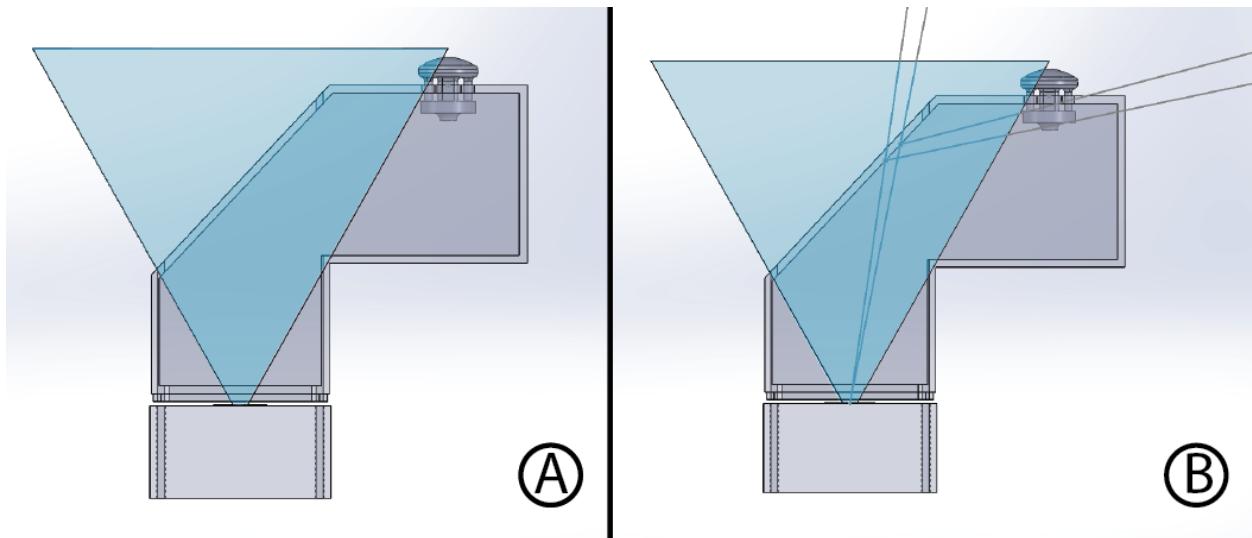
Figure 6.9: An illustration of the raytracing algorithm used for mirror placement. Note that the button in the figure cannot be extruded to meet the field of view cone. A mirror will be glued at the spots where the rays successfully reflect (seen in B) during assembly.

maximum and minimum positions of the component (e.g., we slide sliders to each end of their tracks).

We use raytracing to determine how to place mirrors for components where extrusion failed (Figure 6.9). The designer has to manually insert these during post-print assembly.

We begin by assuming that all device surfaces are candidates for mirror placement. Each ray is cast from the camera to the body of the device, and from there reflected based on the surface normal of the body at the intersection point: i.e., we assume that during assembly the mirror will be placed tangent to the body's inner face. The reflected rays are traced to determine if they intersect any components which were not successfully modified in the extrusion step. If such a component is encountered by the reflected ray, the location on the body that it was reflected from is marked. This leaves a cloud of points per component, which informs the designer where to place mirrors during assembly (see Figure 6.2). Our prototype traces a coarse grid of 20x20 rays because of limitations of the SolidWorks API, in which a single trace takes up to 250ms. A more efficient reimplementation can increase rays to one per camera pixel.

The raytracing algorithm also finds occlusions. If a component is not the first object hit by any direct rays cast or any rays reflected off the main body, the user is alerted that the component needs to be moved or manually modified because it is out of the camera's view. For example, in a case with two buttons in a row and the camera's view parallel to the row, if mirror placement is not possible then the rear button would trigger this alert because all rays cast from the camera hit the front button first. Some cases of this type may be solveable using extrusion after the occlusion is detected: we did not implement this

multi-stage correction in our prototype.

Mirrors can also be used to redirect motion to increase its visibility. For example, buttons moving along the Z-axis (toward the camera) are harder to track than buttons that move in the XY plane. A 45 degree mirror placed next to the button can redirect visible motion. Our prototype does not automatically calculate the locations of these mirrors yet.

The Sauron Hardware

We discuss the components of the Sauron hardware: the post-print processing of fabricated input devices, and our computer vision processing pipeline.

Post-Print Assembly

Due to the nature of our sensing approach, we require that designers' models be hollow and contain a hole of suitable size for the lighting and camera rig to be inserted. Many prototypes are designed to be hollow because such designs conserve printing material. However, this requirement places some restrictions on how other elements, e.g., an LCD screen, can be placed inside the model.

We also require a few steps of assembly to make the prototype suitable for use with our vision pipeline. To increase visibility of the input components versus the background, we require the addition of some distinctive material to the input components. This material can be printed in multi-color 3D printers (see Figure 6.10). Alternatively, coloring the bottoms of the input components with a pen is sufficient. We use a silver permanent pen on dark model material or a black permanent pen on light model material (see Figure 6.11).

Because most current materials used for 3D printing are too brittle to create small compliant parts, users must add springs manually after printing (e.g., to restore buttons after being pressed). This limitation is not unique to Sauron. We designed our buttons to allow for insertion of springs using tweezers (see Figure 6.12, bottom right). Any mirrors will need to be inserted as well. We use small craft mirrors which we affix to the printed device's interior surface with epoxy.

Sauron generates a basic set of step-by-step instructions, automatically displayed in the designer's browser, to assist in correct model assembly. These instructions include automatically-created screenshots of the model highlighting parts that require their attention and example images showing them how to apply mirrors and how to mark components (see Figure 6.12).

Machine Vision

A computer vision pipeline tracks user manipulations of components once they have been printed. We run each camera frame through a series of steps: binarization, connected components detection, and previous frame differencing. This highlights movement of components between frames.
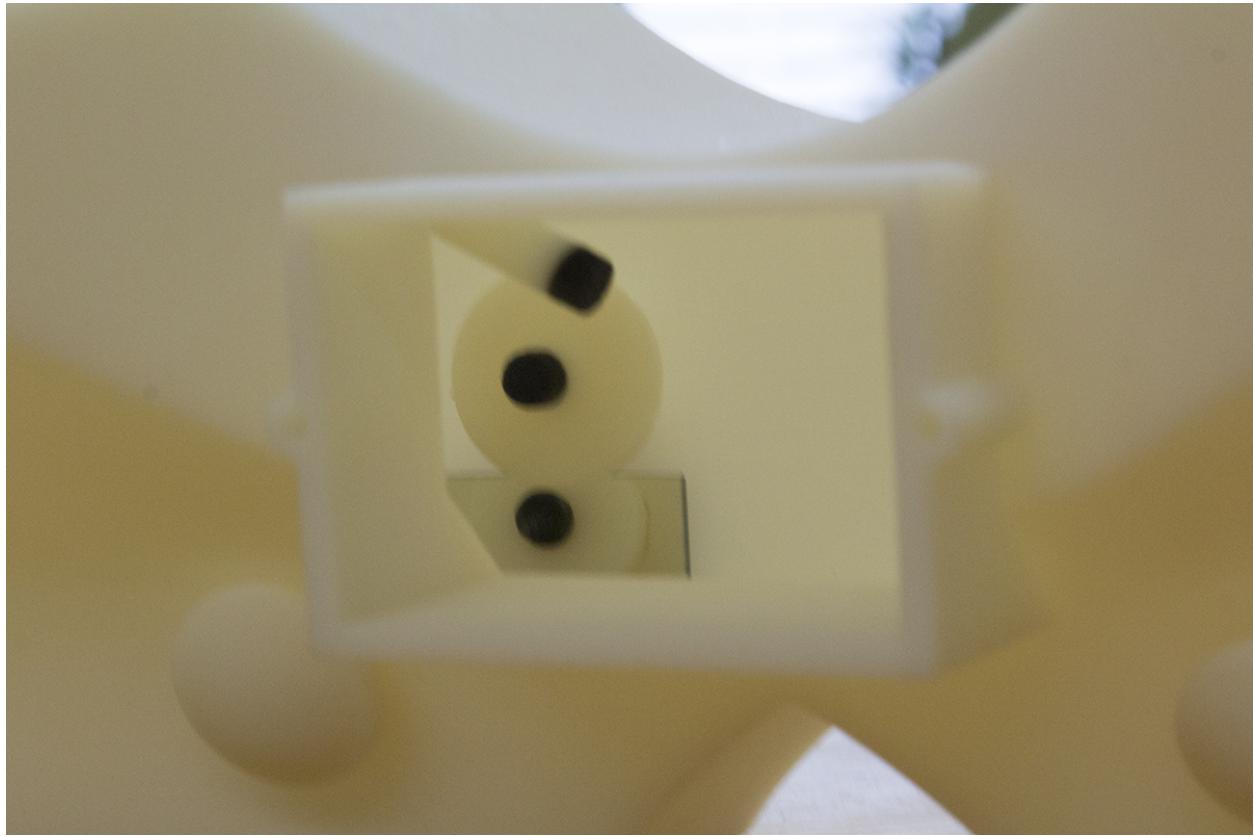
Figure 6.10: Using a multi-color 3D printer (Stratasys Objet Connex 260), we created our video game controller object with distinctive material built in.

Figure 6.11: Components with reflective ink on black material (left) and black ink on white material (right).

2

stick a circle of reflective material on the bottom of each of the direction pad's protrusions as in the example below

3

use tweezers to put a small spring between the direction pad and the body
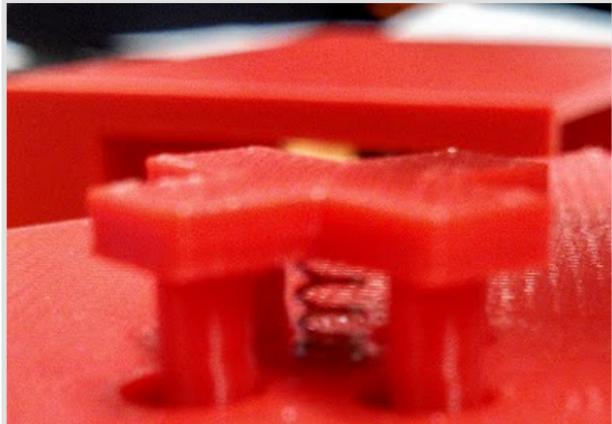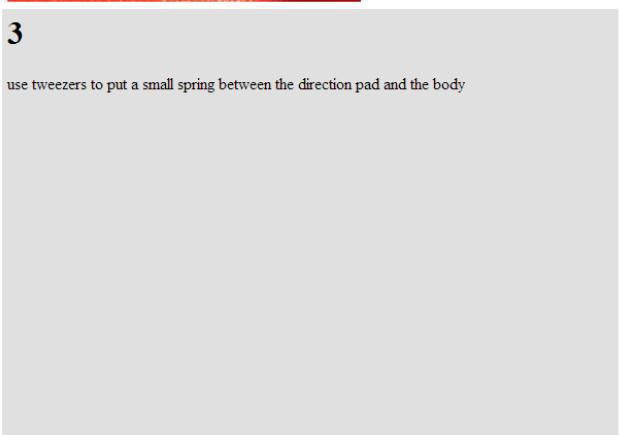
Figure 6.12: Sauron generates instructions that include screenshots of the designer's component with each relevant piece highlighted, as well as instructions for post-print marking and assembly.

Registration

Users have to register components before they can be tracked. During the registration process, regions of interest for each component are determined. A designer is prompted by SolidWorks to actuate each of his components in turn, and a bounding region is created that encompasses all the points through which the component moved (Figure 6.13). These regions determine the relative position of the component within its bounds during the testing phase.

In future work, we would like to explore more detailed simulation in the CAD environment. This could eliminate the physical registration phase by either generating and printing visual markers (and using sensing similar to [26]) in a contrasting material, or by predicting the position of the components in the camera's image and sending that information to the vision software.
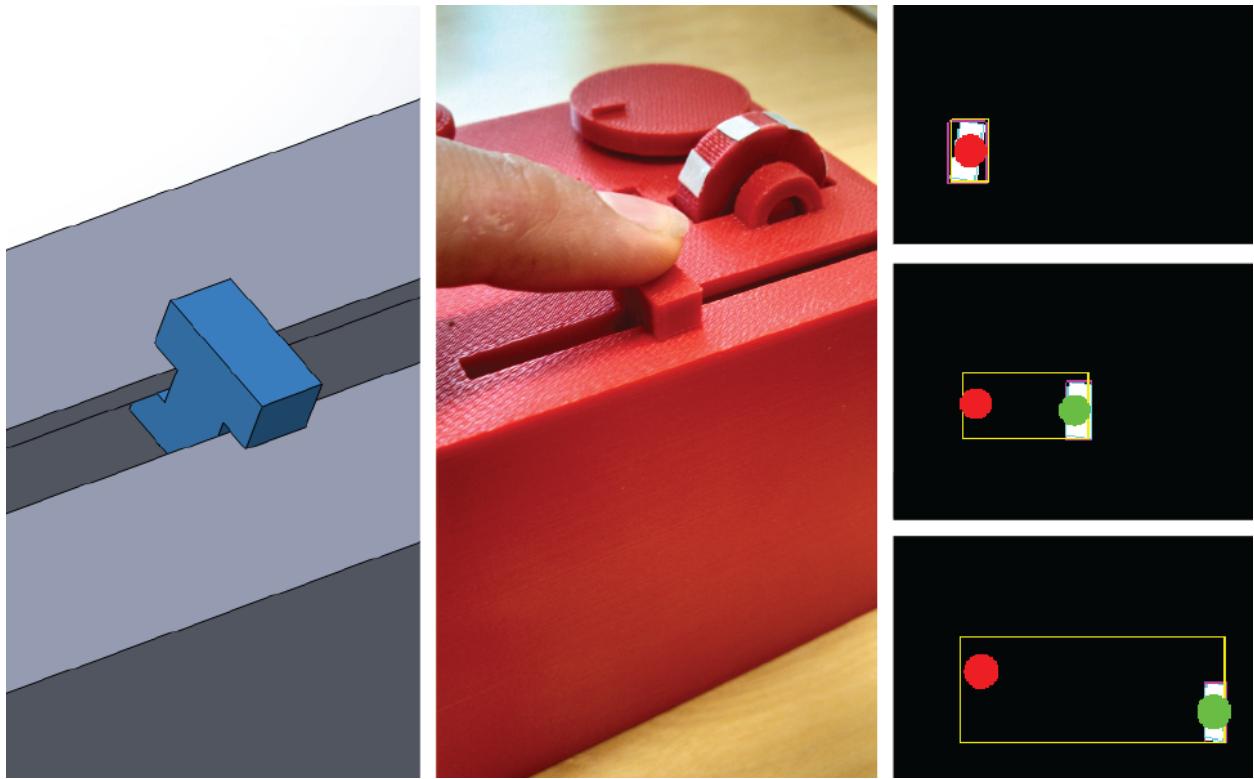
Figure 6.13: Sauron's SolidWorks plugin highlights each model component in turn and asks the designer to move it. The vision software creates a bounding box as the component moves through its range and also saves any information required by the component type. For example, to determine slider position later the vision software saves the two most extreme tracked center points (the red and green dots).

Tracking

After registration, different detection algorithms apply to each input component. The techniques we use for each component are visualized in Figure 6.14.

For *buttons*, we extract one bit of status from movement of its tracked blob. The direction pad generalizes this approach to track four cardinal directions: direction pad motion is based not just on movement and location of blobs, but of their *deformation* with respect to the camera's view. The *joystick* tracks movement of X and Y axes separately, using its main body for one axis and the outer "wings" on the independent arch piece for the second. We find the absolute position of a *slider* in a unit interval by finding its blob on a line connecting the minimum and maximum positions observed during calibration (see also Figure 6.13). The *dial* tracks position as orientation of a blob around an elliptical path, measuring theta between the current location and the first saved location, and using captured $(x_{min}, y_{min}), (x_{max}, y_{max})$ as the major and minor axis measurements of the ellipse. The *scroll wheel* and *trackball* use optical flow to determine amount and direction of relative
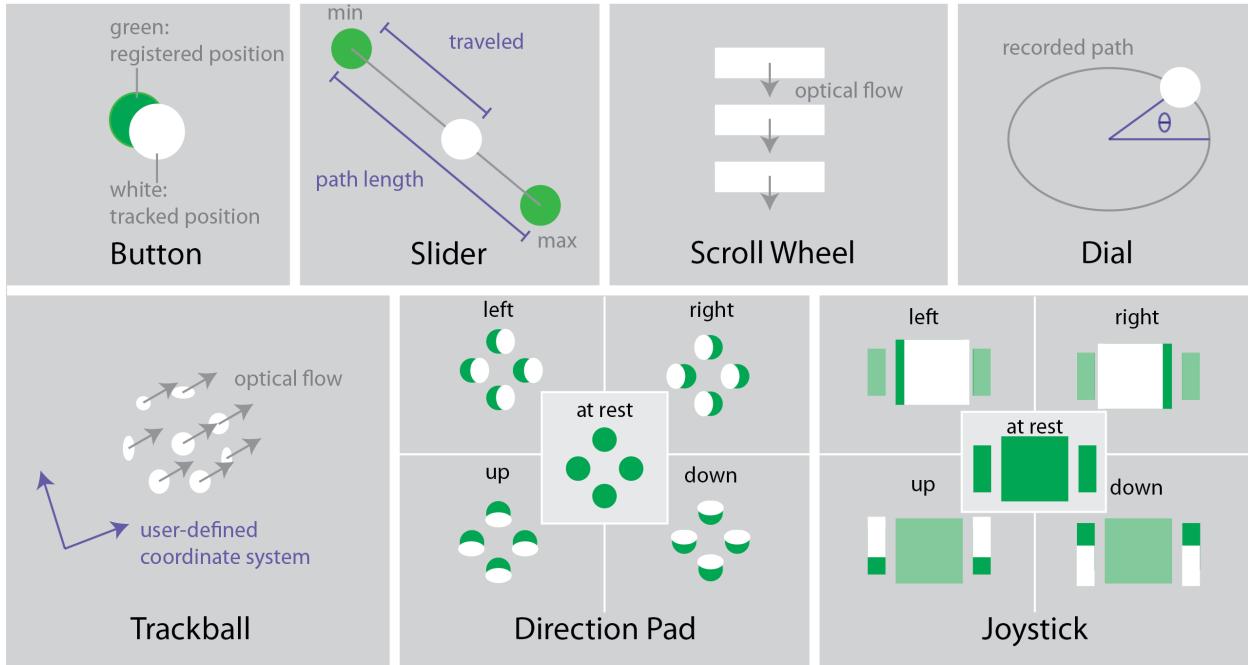
Figure 6.14: The different types of components in the Sauron library require tailored computer vision tracking approaches to extract state information.

movement; the scroll wheel works one-dimensionally as blobs move up or down, while the trackball has the user capture X and Y axis motion separately, and it then measures flow in coordinates relative to those.

We currently do not correct for perspective in our images, which leads to non-linear behaviors in components like the slider and dial. It would be possible to account for perspective analytically since we know position and orientation of a component with respect to the camera in the model. For example, a slider follows a known line through $(x_{min}, y_{min}, z_{min})$ and $(x_{max}, y_{max}, z_{max})$ in the CAD model. Given a slider located at (u, v) in the image, we can find the point that is mutually closest (in a least-squares sense) to the line from the focal point through the image plane at (u, v) and the line of the slider's movement.

The vision component of our prototype is implemented in C++ and runs at interactive speeds (>32fps) on a 2011 Macbook Pro. We rely on the open-source computer vision library OpenCV [83] and OpenFrameworks [84]. Messages are passed between SolidWorks and OpenFrameworks via the OpenSoundControl (OSC) protocol. OSC messages are sent over UDP and contain an address (e.g., *"/button/1"*) and payload (e.g., "on" or "off"). Our prototype uses these messages to communicate processed events, to start and stop test mode, and to start and stop registration of a particular component (see Figure 6.15).
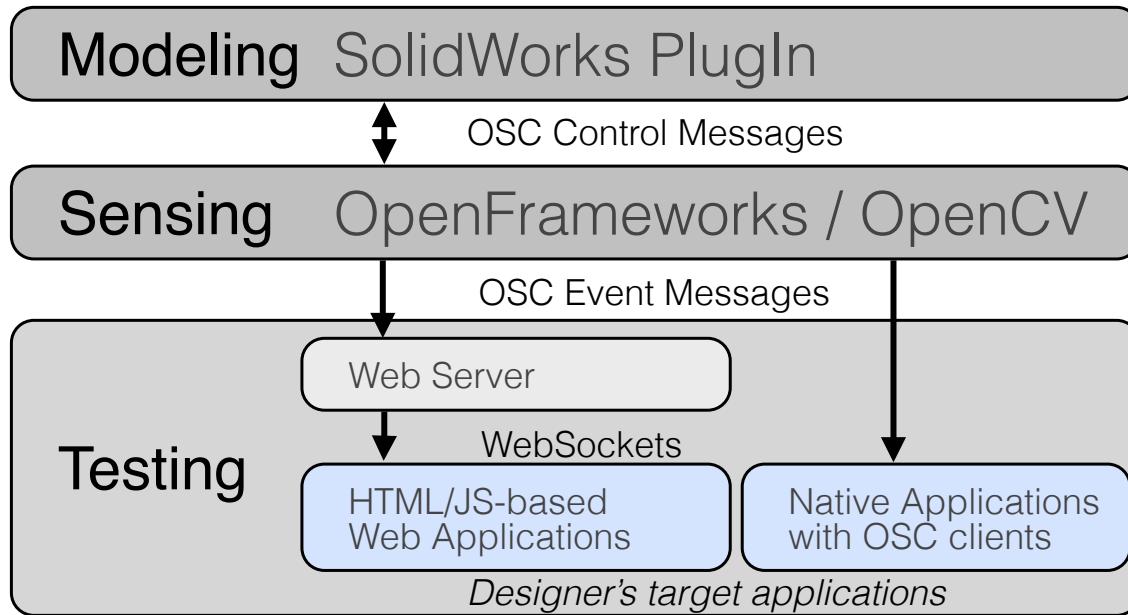
Figure 6.15: SolidWorks and OpenFrameworks exchange messages via OpenSoundControl. OpenFrameworks also sends OSC messages containing processed data to a WebSockets server to deliver events to a user's application.

Event Output

Sauron can deliver input events to application using either OpenSoundControl or WebSocket messages.

OSC messages for simple control

Existing third-party tools can transform OSC messages into keyboard, mouse, or game controller events, without the need to write code. For example, using the third-party program OSCulator, a designer could simply assign messages coming from */joystick/x* to move the mouse cursor in the X direction and from */joystick/y* to move it in the Y direction. This strategy can also be employed to generate USB HID game controller events and key presses automatically without code.

WebSocket communication with web applications

For designers who wish for more control and who are familiar with programming, we enable event consumption in web applications written in HTML and Javascript. Leveraging web ap-

plications as a platform allows interface prototyping on any device with an internet-connected web browser. We use a node.js server which exposes processed events over WebSockets. We adopt this strategy from Midas [99].

6.5 Evaluation

In order to demonstrate that Sauron's sensing and fabrication technique fits our criteria of being a cheap, fast, and flexible method of prototyping, we elaborate on each of these criteria below.

Cost-Effective

Sauron's sensing hardware for our prototype includes a repurposed webcam with a custom circuitboard to hold an integrated ring light: a setup costing roughly $35. Our initial experiments indicate that existing cameras with ring lights (as in Figure 6.3) can also be used for sensing, without the need for custom electronics. In addition, a single sensing setup can be used for multiple prototypes, albeit not simultaneously, allowing amortization of cost over many projects.

To work with our computer vision pipeline, prototype objects are hollow. This not only enables our sensing technique, but also saves materials. The fabricated prototypes we used for our research were created on a Stratasys uPrint SEPlus, at the cost of approximately $8/in^3$. Today, more and more hobbyist machines are capable of laying down the sacrificial support material that is necessary to create mechanisms in-place that do not require assembly, for roughly $50/kg$.

Fast

We performed an informal evaluation with three mechanical engineers. All were proficient SolidWorks users. We first explained how Sauron works and demonstrated a printed prototype containing examples of all our input components. We then asked them to prototype a disk jockey (DJ) controller that could be tested with Sauron. Common functions on such controllers are volume and EQ control knobs, large "scratch" wheels for two audio channels, and a crossfader. We emphasized thinking aloud, as we wished to determine how the constraints of our vision-based system affected their design process. Participants did not run the plug-in itself during the modeling sessions due to time constraints, but we ran it on the resulting models and fabricated one of their designs (see Figure 6.20).

All of our participants modeled DJ mixing boards that could be successfully used with our vision-based sensing approach (Figure 6.16). They followed different approaches to place the camera – though all showed concern for the aesthetics of their design and accordingly tried to mount the camera inside the main enclosure or otherwise out of the way. One user mounted the camera sideways (Figure 6.16A), but at a location such that the mixer's
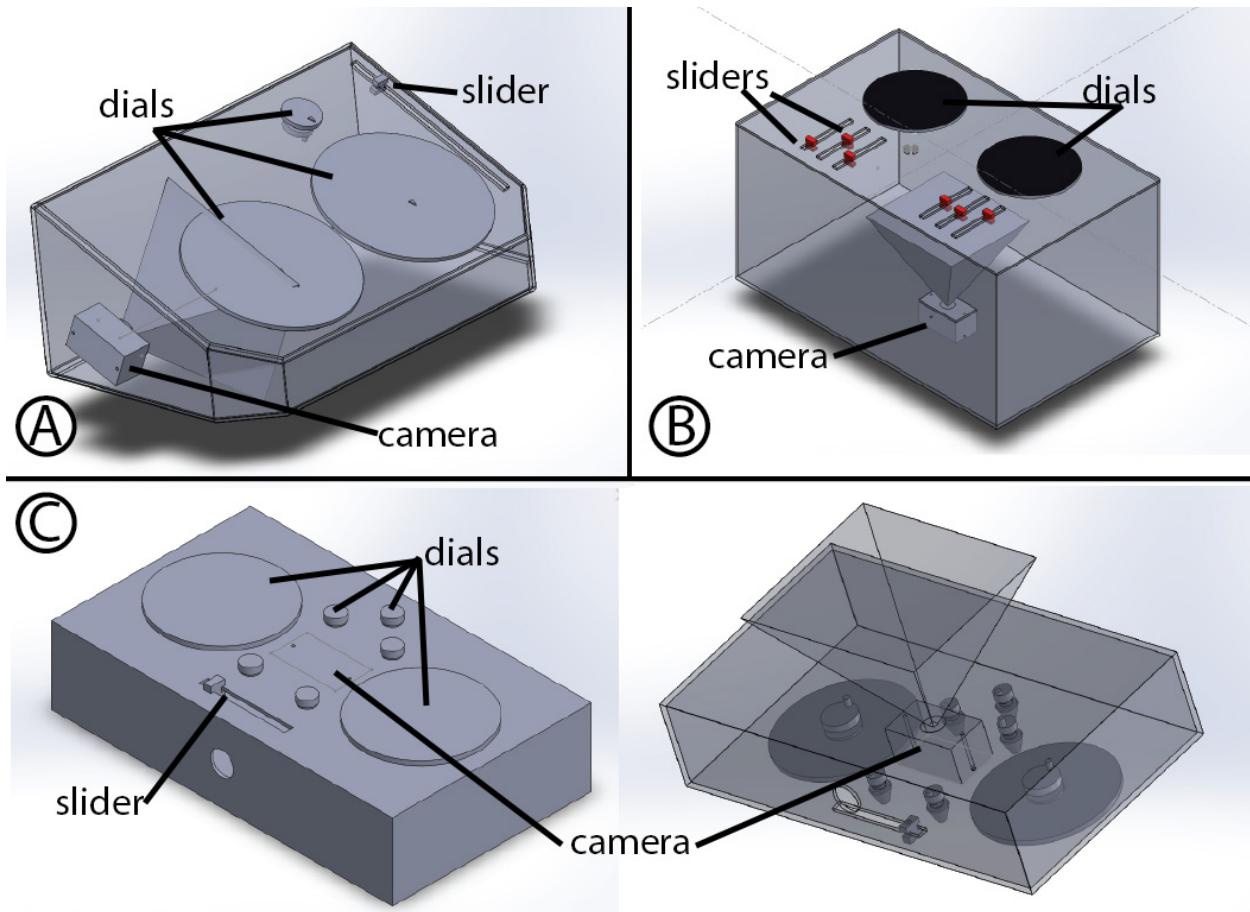
Figure 6.16: Our three user study participants prototyped DJ mixing boards using our component library. Each had a very different strategy for ensuring the camera could see all components. The assembly on the bottom (with interior cutaway view at right) was designed to have the camera inside reflecting off mirrors placed on the back wall.

components would not occlude each other; another created a very deep box at the start, stating that he preferred "to focus on the user side, rather than the camera because I don't care about the box size" (Figure 6.16B). The most ingenious design mounted the camera on the top, pointing down, so that all components would be visible in a single large mirror placed at the bottom of the controller (Figure 6.16C). In aggregate, while users had to plan for the visibility constraints of camera sensing in their design, these constraints were not seen as overly burdensome.

One user wished that an interactive design checker was available to test his design iteratively for visibility. A complete model modification pass currently requires ≈ 5 minutes to process a non-rectilinear model with 10 components, because of slow calls to the SolidWorks API. Based on this feedback we implemented the "quick check" feature which highlights

components that are immediately within the viewing area without performing ray-tracing or extrusion.

Participants also successfully modified the library of parameterized components. One participant stated that it was important to her that the sensing portion of each component was decoupled from the user-facing portion. For example, the scratch wheels are large on the user's side to enable users to place their entire hand on them, while the internal dial diameter is small so it can be seen by the camera in its entirety (see Figure 6.16C). The same user also wished that there was better documentation for the component library, describing how large holes for mounting needed to be.

Flexible

To determine if Sauron allows sufficient design freedom for users, we performed an analysis of models created without our approach in mind, as well as modeled and processed several objects ourselves.

Analysis of Pre-Designed Models

To determine if designers working without our constraints in mind would create prototypes that are compatible with our vision-based system, we downloaded several online 3D models and analyzed them. The models, which comprised a deduplicated set of all models with keywords "interactive" or "controller" on the model-sharing site grabCAD.com, ranged from XBOX and Guitar Hero controllers to interactive desks with keyboards. None of the devices that we analyzed were designed for 3D printing, but rather for rendering or as engineering drawings. Our first step in processing them was estimation of the internal geometry of the bodies, for which we assumed simple shelling (i.e., no internal supports, wall thickness approximately .1", interior curves following the curves of the outside of the body). After this was done, we selected several candidate camera locations which would not interfere with what we understood to be the user-facing functions of the device, and we measured which components would be visible to the camera directly, which via extrusion, and which via reflection.

Out of 10 devices we analyzed, we believe that 7 of them could be successfully processed by Sauron (e.g., see Figure 6.17). Three devices were too thin—this caused serious occlusion problems between components. Their bodies also did not allow space for the inclusion of mirrors to solve the occlusion problem (Figure 6.18). One of the failing devices, a steering-wheel-style device, had two handle areas with buttons at their far ends and thin, continuously-curving surfaces bending away from the main body. Using just one mirror bounce, it would be impossible to see around these bends to the buttons at the ends.
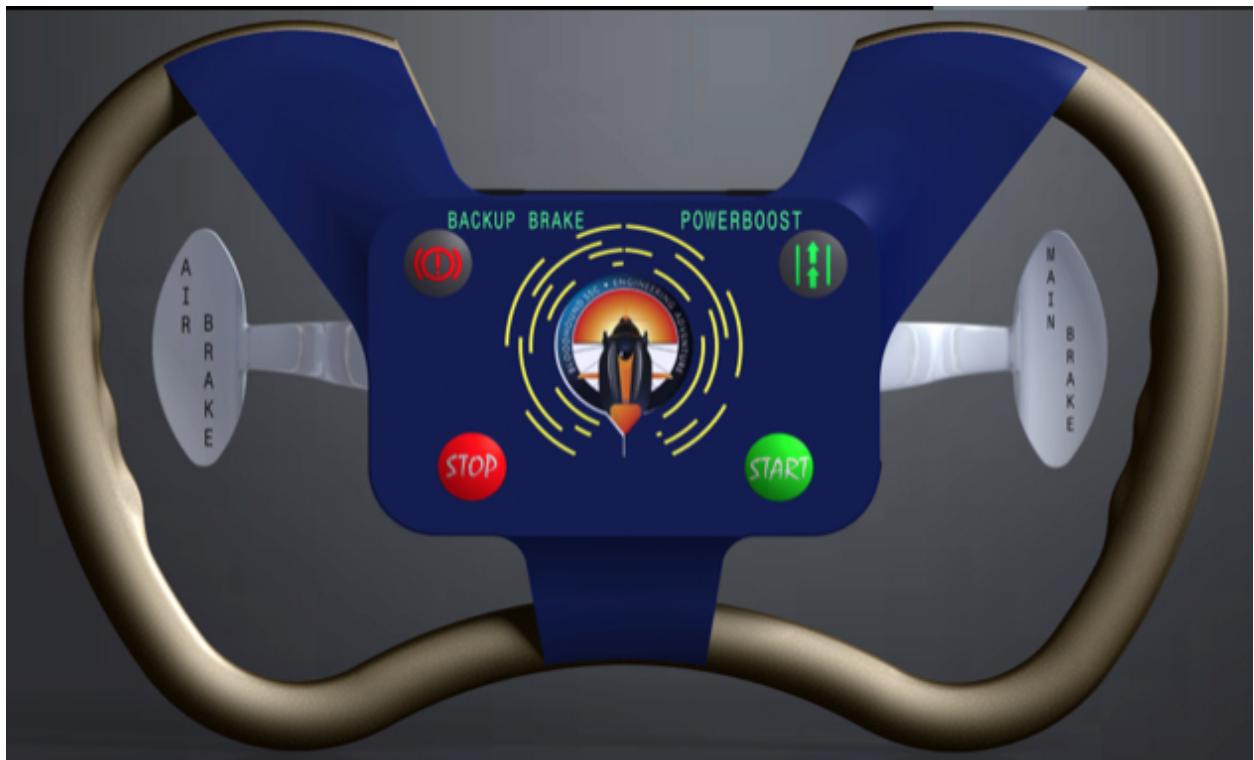
Figure 6.17: This model found online would work well with Sauron's sensing technique; all components are centrally located within a body that is not superlatively shallow. (By user Florin Traila on GrabCAD, from `https://grabcad.com/library/bloodhound-ssc-steering-wheel--22`, by permission.)

Figure 6.18: This model found online is too shallow to sense with Sauron—occlusion and curvature would prevent correct sensing with computer vision. (By user Kevin Schneider on GrabCAD, from `https://grabcad.com/library/game-controler`, by permission.)
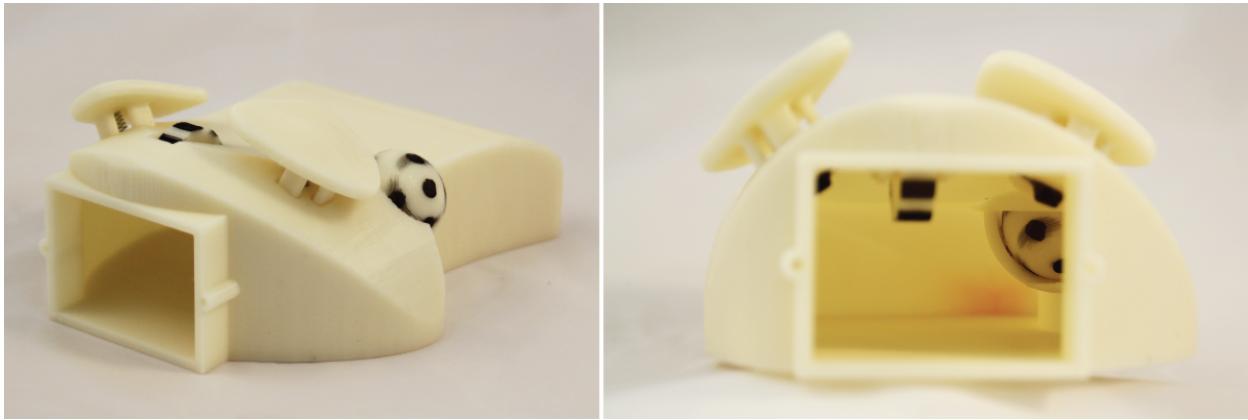
Figure 6.19: Our ergonomic mouse prototype has a trackball the user can manipulate with his thumb as well as two buttons and a scroll wheel. On the right is the camera's view of the inside of the mouse.

Example Devices

We also fabricated three prototypes that display the range of interactive components our prototype system offers.

Ergonomic Mouse

Our ergonomic mouse (see Figure 6.19) has a trackball the user can manipulate with his thumb as well as two buttons and a scroll wheel. We configured the mouse to control the mouse cursor on a laptop using OSCulator. Due to large tolerances in our model, the scroll wheel tended to oscillate between "up" and "down" states after being released. This problem could either be addressed through modifications to the model or by double thresholding in our computer vision component.

DJ Mixer

We constructed a DJ mixing board—based on a study participant's design—in two pieces to fit on our 3D printer's bed size. We converted the OSC messages sent out by Sauron's vision software to MIDI messages to control Traktor, a professional DJ application (see Figure 6.20). One issue this prototype raised was that disparities between the virtual and physical camera parameters affected visibility. While the components were designed to fit within the virtual camera's field of view, an offset between the lens axis and the center of the sensor on our (manually-modified) camera led to some components falling outside the physical field of view. We are confident that better calibration and measurement can overcome such problems.
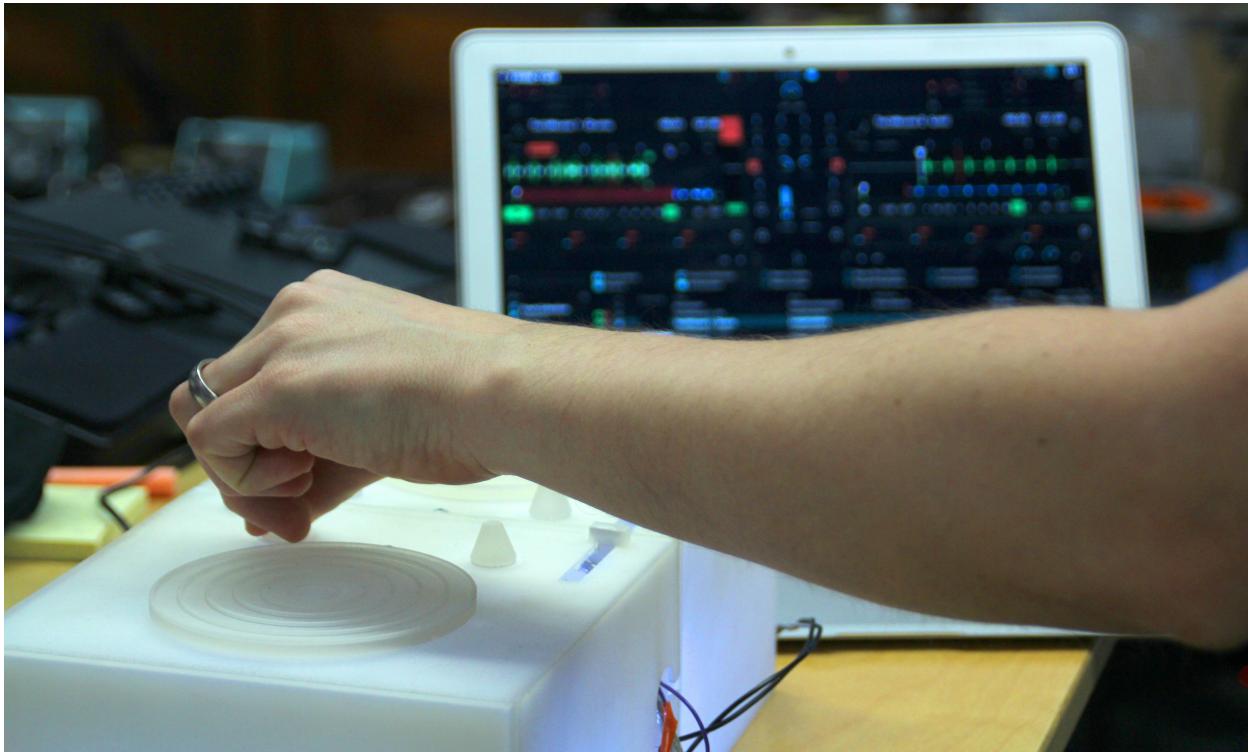
Figure 6.20: Our DJ mixing board, based on one of our users's designs, has sliders and two dial configurations: raised knobs for easy manipulation of volume, and a larger flat wheel for seeking and scratching songs. The different types of dials share a sensing algorithm, however, as their interior parts are similar.

Game Controller

We developed two versions of a video game controller, shown in Figures 6.1 and 6.2. To test responsiveness, we built a simple browser-based game to accept data from Sauron's WebSockets server. The controller moves the player character around (joystick) and shoots fireballs (buttons). We found the game was playable, although detection of the joystick position was noisy. This seems to be due to the fact that the blobs tracked for the main base and the two flanks were lumped together when the joystick was in certain configurations, e.g., at extreme right. We believe this is not a fundamental issue and could be mitigated by iterating on the joystick's interior design or by using a higher-resolution camera.

6.6   Discussion

Current Sauron prototypes are all tethered to a PC. There are opportunities to explore interactive devices not connected to computers. For example, tangible peripherals for mobile

device could also be prototyped using our system. Modern smartphones have on board cameras and LED flashes, and enough on-board processing to perform computer vision. Modeling the phone and its camera parameters could enable mobile prototypes designed to encase the phone.

We believe that an exciting use for Sauron is in the development of entirely novel input devices which are not supported by traditional electronics. One example is a curved slider: electronics typically measure either linear or rotary motion, but a slider on a curved or irregular track would be easily prototyped using Sauron.

The creation of interactive prototypes also need not be limited to 3D printed plastic. Digital fabrication opens the doors to many new areas of exploration: any process which fabricates material according to a model created in software could be processed similarly. One such promising technology is laser cutting, where we already see the ability to create 3D models through sliceforms or layering of 2D cross-sections of an object. Laser Origami [71] has pushed the bounds further, and it is not difficult to imagine fully laser-cuttable mechanisms that could be tracked by Sauron.

For future work we hope to test our tool more extensively with designers in the context of a workshop or class. We are also planning to explore tools to simplify the physical design process for users unfamiliar with CAD tools.

Sweet Spots

Sauron works very well for prototyping the hand-feel and interactions of devices. It does not allow prototyping realistic weights, as it requires hollow models. Sauron also works best for roughly hand-sized devices (with our camera). Prototyping larger devices would be possible using a higher-resolution image device.

Limitations

We have identified several limitations of the Sauron system. Some are inherent to the approach, while others could be mitigated with additional engineering.

Post-print assembly

Currently, our prototypes still require some post-printing assembly for inserting mirrors, if they are necessary. However, we believe this step is significantly less time-consuming than the process of wiring up a prototype with discrete electronic components. Thanks to multi-material printing, it is possible to print distinctive marks into the object (see Figure 6.10), and some current work is experimenting with printable reflective surfaces [86].

Necessity of registering components post-print

A second limitation is the required registration process after printing. In future work we plan to create more sophisticated algorithms which can pre-determine bounding boxes of

printed components using the digital model, or which can generate visual markers to denote end points and motion type. This would allow designers to skip the registration step.

Visible light does not work in all environments

Because we currently use visible light sensing, environmental lighting can interfere with our algorithms. For example, our prototypes behave erratically when tested with bright fluorescent lights directly overhead. Some components, like the slider and joystick, require a certain amount of clearance around them to move properly. When bright light shines through these gaps, vision tracking can become problematic. One remedy is to move sensing into the infrared spectrum.

Model modifications do not chain

Our algorithms do not deal with cases where chaining of model modifications is required: i.e., if a component could be seen by first extruding, then reflecting, it will not be correctly processed by our algorithm. We provide the field of view of our camera as a reference to designers so that they can correct cases like this on their own, however more complex automatic interior geometry modifications are possible.

Limited component library

Finally, we support only a limited library of components, and not all components can be modified through extrusions. However, this library is extensible by expert users who can define and label faces for extrusion and who can choose or program appropriate tracking algorithms. Our informal evaluation suggests though that configuring and changing existing components to suit the needs of a particular prototype may be sufficient to cover a useful design space.

6.7 Conclusion

Where Midas explored touch sensing and Lamello explored acoustic sensing, Sauron examines Fabbing to Sense with vision-based sensing. Using 3D printed mechanisms and distinctive materials, it allows for manipulable, tangible input devices designed with a sensing technique in mind.

Chapter 7

The Final Word

> When I look into the future, it's so bright it burns my eyes.
> — Oprah Winfrey

This thesis has presented techniques for leveraging and modifying digital geometry of objects to aid in creating fast, cheap, and flexible prototyping tools. We conclude with a discussion of the links between and importance of described projects, as well as pointers to future work.

7.1 Discussion of Projects

The large thrust of this thesis is this: designing tangible input devices is challenging—far more so than creating prototypes of graphical user interfaces—because prototypes must combine software, hardware, and custom enclosures. For this case, we believe digital fabrication can help. With digital fabrication, we have a model of an object *before we have the object itself*, which we can manipulate and simulate in light of the sensing mechanism we plan to use for our final prototype. We call this process, which links fabrication-for-interaction to models-for-simulation, "Fabbing to Sense."

The three projects we discussed—Midas, Lamello, and Sauron—serve as exemplars in exploring the space of Fabbing to Sense. Between them, they leverage touch, audio, and vision: several very common sensing modalities today. For this thesis, that has the benefit that they are easy to communicate to other researchers. Ultimately, though, designers and makers, and those they create for, do not need to know the workings of the sensing mechanisms.

As with existing graphical user interface toolkits, which can abstract away callbacks, container structures, rendering styles, and other low-level details, we hope to see our techniques and others like them be abstracted into black boxes. As such, they can support designers' high-level goals, like "button here," or "continuous volume knob," without requiring additional expertise. From designers' goals, the key pieces are simulation and modification—using knowledge of the sensing technique—to allow final fabrication. We dipped into such high-

level specifications in Midas and Sauron, and found that designers were able to create unique input devices using the systems.

Beyond simply supporting design tasks, we see simulation as a way to ensure that unusual sensing techniques can disappear from the perspective of the final end-users of prototype devices. Midas can check for assembly errors to ensure its sensors will behave properly. Sauron ensures that its modifications will not interfere with users' manipulating input components through their full ranges of motion.

It may also be the designer's choice that the sensing mechanism *not* fade into the background. In the Lamello project, we used sounds within typical human hearing ranges as control signals. While some may view this as a downside (and likely one that could be corrected using different materials and/or smaller structures), it is possible for designers to integrate this as a feature: as one example, a slider mechanism could be programmed to play a favorite song as it is manipulated. Thus, depending upon the selected modality, a designer may choose to either hide or accentuate a device's sensing technique.

In any case, integration of physical form-finding with specification of input components can be a challenge. This work is currently relegated to those with expertise in Computer-Aided Design (CAD) tools. While it does not fall under the banner of Fabbing to Sense, we have also done work on allowing users to create such physical specifications that are authored in the real world, using easily reconfigurable materials and leveraging computer analysis, simulation, and optimization [102]. We see this as part of the future ecosystem of Fabbing to Sense.

All of the above is part of a larger trend in computer science, and especially in Human Computer Interaction, whereby computing devices are imbued with domain expertise, and further they are given the ability to assist users via guidance or corrective actions. Such creativity support tools can exploit the strengths of each system. That is, they marry human creativity and perceptive skills with machine precision and simulation. Systems connecting the two have been tested already for use in surgical (e.g., [53]) and cooking (e.g., [96]) environments, as well as innumerable other physical and virtual Augmented Reality (AR) environments for teaching novices or supporting experts. As more systems successfully demonstrate this technique, the human-machine interlink for aiding designers in their creation of tangible input devices represents a natural research direction, as well as a necessary component for designing the future of interaction.

Another piece of the future may be changing some of the constraints for which Fabbing to Sense systems optimize. Multi-material printers can now create conductive traces [121], but may someday be integrated with pick-and-place machines to allow for fully-integrated electronics in a single pass. When a designer has to perform the assembly, we prefer to have a single sensor that can sense all inputs at once. However, if a machine performs the assembly, future systems could employ mixtures of multiple types of sensing, with automatic optimizations around cost, size, or other design factors. This would allow for further abstraction: designers can stipulate that some inputs should be continuous and others discrete, allowing the system to assign appropriate sensors to each input.

Moving beyond prototypes, Fabbing to Sense could aid designers in creating low-power

environmental inputs for long-term use. As mentioned in the Lamello example of a wall-mounted slider, devices which can be sensed through-air by a user-carried sensor need not be continually powered. Most 3D printing processes today create plastic objects that are not suitable for installations such as this—particularly for Lamello-style sensors which subject the devices to significant physical forces in typical use. However, more and better materials are becoming available to those who work with digital fabrication at every level, whether hobbyists or professionals, and the use of metal-based 3D printers or multi-axis CNC mills with sturdier materials may allow for longer-term usefulness of these devices. Aside from materials improvements, another vector by which to approach more viable objects would be constraining sensing: input and sensing modalities which do not require friction, striking, or other continuous contact between fabricated or electronic parts will likely outlast those that do.

7.2 Future Work

Over the course the thesis, we have pointed to a variety of limitations and possible improvements per-project. In general, we see making each project's sensing apparatuses more mobile to be an useful engineering task. All three projects use sensing types common to today's mobile smart phones (capacitive touch, microphone-based audio, and camera-based video), and additional engineering could better leverage those and other sensors already available to designers from within their environment. Some projects already use in-phone multi-touch screens for custom-manufactured capacitive inputs [22, 23], and the built-in microphone and camera come with easy-access APIs for use as sensors.

Beyond per-project improvements, we have laid out a design space for linking geometry to sensing. Midas, Lamello, and Sauron each represent a single point in this design space, and fuller exploration of the space may lead to companion projects to those described here. In particular, the advent of multi-material PolyJet 3D printers seems to open a wide variety of options for exploring colors, transparency, and flexibility as features in a sensing design space, and the Voxel8 [121] printer, which can lay arbitrary conductive materials and plastic together in a single pass, points to opportunities to explore induction, human body heat sensing, or magnetism as sensing operations. Or, as discussed in Chapter 2, level sensors may pair well with translucent tubing filled with liquids.

Overall, we recognize that there are several assumptions made by the projects presented in this dissertation. Namely, our projects leverage a *single fabrication machine* for creating *one prototype* at a time, which is *hand-optimized* by a designer and sensed by a *single* sensor. We discuss possibilities opened by willfully subverting each of these assumptions in turn.

Ecosystems of fabrication machines

The projects described in this dissertation have largely focused on single fabrication machines working to create a finished prototype object. We also see opportunities for combining the

abilities of several machines, whether to speed up the prototyping process or to investigate unique properties that allow exploration of the machines and sensors design space.

For celerity

While 3D printers allow for near-infinite flexibility in the forms that they are able to create, they still run very slowly. This can be compared to laser cutters, which offer significant speedups in exchange for only producing 2- or 2.5D prototypes (or limited 3D prototypes, see [71]). Some prior work has investigated speeding up fabrication through integration of lasercut and 3D printed pieces [12], or use of building blocks with 3D printed parts [73]; however, these speedups do not make any use of their knowledge of the completed object post-fabrication. As demonstrated in the Lamello project, lasercut tines integrated with 3D printed bodies allow for faster fabrication with similar accuracy. However, having a designer hand-assemble highly complex input mechanisms fabricated on multiple machines may be prohibitive in terms of time spent: better would be using pre-fabrication simulations to make these devices (partially) self-assembling [115].

For properties

Speed is a factor worth considering, but by leveraging multiple machines designers can additionally access a larger variety of properties. Plastic can offer a sturdy base with configurable haptics [116], while inkjet-printed circuitry can provide a slide-in base for electronics. Lasercut or cnc-milled wood may pair well with delicate paper to create shape-changing interfaces [136]. Leveraging multiple properties (potentially in combination with multiple sensors, as below) may allow for Fabbing to Sense a greater variety of devices. And in some cases, these multi-machine integrations could allow for output in addition to sensing [136].

Branching prototypes

One important benefit of digital models of prototypes is that they become like code: they can be stored, shared, replicated, versioned, and unit-tested. Version-control website github [33] in 2013 added a built-in viewer for STL files in user repositories on the site, offering a powerful tool for those who wished to version their physical designs. However, each design is hand-crafted by the designer. In the future, we would like to explore tools which can create likely *spaces* of prototype designs given an initial seed from a designer, and which then allow testing multiple similar designs in parallel. This could follow prior work on understanding design spaces and how computers can support designer exploration, particularly through parametric modeling [134, 133]. Given the nature of the sensing performed by the toolkits presented, this type of small multiples testing should be as straightforward as attaching the sensing module to each new prototype.

Machine-optimized prototype designs

While describing a design space and smartly *exploring* points in it to test in parallel could support a designer who has something in mind, another opportunity lies in automatic *generation* of digital interfaces to suit particular people and/or tasks. This is similar in theory to Cogtool [52]—which allows designers of web applications to demonstrate tasks for their interfaces, then optimizes the interface to make completion of those tasks as quick as possible for end-users—, and Supple/Supple++ [32, 31]—which model users' motor and vision capabilities and automatically adapt their GUI to suit.

We believe that there is significant territory to be explored in modeling users' individual capabilities as relevant to tangible input devices, as well as understanding how to create optimal inputs devices suited to specific tasks.

To suit particular users

Individual users have wide-ranging abilities and preferences, especially when it comes to something as personal as the hands. In addition, mobility-impaired users may have special requirements for input devices. Traditional mass-produced input devices are designed to be comfortable for 95% of target users, however the advent of digital fabrication allows for one-off objects with no startup costs like those associated with tooling in traditional manufacturing. Thus, we can measure the capabilities of a single person (How large are Giorgia's hands? How far can Ethan bend his thumbs? How fast can Shiry pinch her fingers together?), and use our results to design for that person specifically. Such measurements might inform dimensions (e.g., overall size of an input device), locations (e.g., spacing between buttons), or even sensitivity (e.g., matched to user grip strength). This has seen a bit of exploration in the form of anatomical scanning for medical devices [111], but *motion* is critical for interaction.

To suit particular tasks

Our lives contain a variety of general-purpose input devices, like the mice and keyboards that we find at our desks, or the video game controllers we use when we unwind. As described above, with digital fabrication machines we can imagine a future in which each input device is uniquely suited to a single task: we can ask questions about a task, and how a particular person approaches it (What abilities does Pat tend to use most when playing League of Legends? How often does Friedhelm scroll through documents while editing them, compared to insert/delete tasks?), and use these to inform the layout (e.g., more frequently-used functions close to dominant hand/fingers) or sensitivity (e.g., a lighter push will activate a time-critical function, versus a more substantial push necessary for an irreversible function) of input devices.

Multi-sensor Units

In the interest of reducing assembly time, all of our projects leverage a *single* sensor in a single location. However, many commercially-available sensor bundles could offer additional types of data. For example, modern smartphones collect a wide array of sensors—e.g., microphone, camera, accelerometer, gyroscope, magnetometer—into a package that could easily be attached to a single point in a prototype; likewise, Texas Instruments' BLE SensorTag sports an ambient light sensor, humidity sensor, barometric pressure sensor, magnetometer, temperature sensor, and more [107]. By leveraging multiple of these sensors in a single prototype, we can create objects which respond to multiple modalities of interaction: for example, supporting the design of objects which integrate squeezable soft components sensed using air pressure, yet can also determine their orientation in space using an accelerometer, allows designers greater flexibility in their process. Modelling potential interference between different sensor types may present an interesting problem here.

7.3   Closing Remarks

As computing moves off the desktop and into the world, we see designers exploring many varieties of input devices to suit new and specific tasks. These new physical devices necessitate a new kind of prototyping, that does not rely fully on *virtual* software but which can help create functional *physical* objects in a fast, cheap, and flexible way. This dissertation has described digital fabrication as a means of accomplishing this: by linking a digital design to a completed physical object, we can shorten the drudgery associated with each iteration cycle by offloading expertise—that previously would have been required of the designer—to her computer. We have presented three examples of this: Midas performed sensor routing and used this knowledge to pre-program its capacitive sensor microcontroller and ensure assembly correctness. Lamello used 3D geometry to predict tines' resonant frequencies for audio sensing. Sauron manipulated an object's digital model to ensure that a single camera would be able to sense all the components inside, as well as to keep pieces from colliding in the hands of an end-user. All three represent the paradigm of "Fabbing to Sense," which employs knowledge of the sensing technique that will ultimately be used *throughout* the design process, and optimizing prototypes for that technique.

Our suite of tools, and the space that they explore, will hopefully empower designers to invent and hone designs for the future of interaction.

Appendix A

Thesis Talk Video

As part of UC Berkeley's graduation requirements, I gave a public talk about the contents of this thesis, representing an abridged version of the information contained in this document. This is available online: at time of writing, it is accessible as a YouTube video at `https://www.youtube.com/watch?v=SgZUxloEo4s`. It may move, but you'll almost certainly be able to find a link to it off of my homepage in the future at `http://valkyriesavage.com`.

Appendix B

Definitions

additive fabrication In additive fabrication, material is deposited and a shape is built up.

subtractive fabrication A subtractive fabrication process removes material to create a form. Excess material may be reused in another project or discarded.

3D printer A 3D printer is one of a class of machines that additively create a three-dimenstional model from one or more materials.

FFF FFF (fused-filament fabrication) 3D printers lay down material by melting and depositing a filament in a precise pattern.

model material Model material is the substrate that composes the final object.

support material Many modern 3D printers are capable of laying two types of materials, model material and a secondary, sacrificial material that can support overhangs in the model during printing, then be removed.

SLA SLA (stereolithography) printers use a bath of UV-curable polymer and a controllable UV laser. The laser "draws" each layer on the polymer, causing photopolymerization where it strikes. Excess material is simply poured out for reuse.

SLS SLS (selective laser sintering) 3D printers contain a bed of material (e.g., metal powder) which is compacted and formed into a solid mass of material by heat and/or pressure without melting to the point of liquefaction. Excess material can be brushed off and reused.

binder jetting Binder jetting is a powder-based printing technique similar to SLS, but instead of melting a powder together to create layers this method uses inkjet heads that drip a binder (e.g., epoxy) to adhere the powder particles. Unbound powder can be brushed off and reused.

PolyJet PolyJet printers have print heads similar to those of inkjet printers which sweep across the build area depositing material. Following the printer head is a UV light, which cures deposited material droplets.

vinyl cutter A vinyl cutter subtractively processes 2D materials with a 2-axis knife blade, cutting patterns into them. Vinyl cutters are typically used for thin, flexible materials.

laser cutter A laser cutter guides a laser's output over a 2D domain for processing flat materials. Laser cutters can cut or engrave into materials, and are often used for rigid materials $< \frac{1}{4}$ inch thick. Some have rotary attachments for engraving on circular surfaces like the outside of a glass.

CNC router A CNC router uses a 3-axis rotary mill to cut through thick, rigid materials, like wood or certain metals. Some CNC routers are portable and can attach to many materials, while some are stationary with beds into which material is loaded.

CNC mill A CNC mill is a multi-axis machine which subtractively creates a 3D shape from a block of material, usually metal or wood.

Appendix C

Open-Sourced Code from Thesis

I have open-sourced the code for the Midas and Sauron projects (available under the GPL 2.0), as well as this document and other projects—both research and personal—on my github account: `http://github.com/valkyriesavage`. Note that these pieces of software are "research code": at the time of publishing this document, I haven't fully cleaned and documented them for easy and straightforward use by others.

Bibliography

[1] *123D Circuits.* https://123d.circuits.io/.

[2] *3D CAD Design Software Solidworks.* https://www.solidworks.com/.

[3] *3D printing wood is possible.* https://i.materialise.com/blog/3d-printed-wood-is-coming-to-i-materialise.

[4] Eric Akaoka and Roel Vertegaal. "DisplayObjects: Functional Prototyping on Real Objects". In: *CHI '09 Extended Abstracts on Human Factors in Computing Systems.* CHI EA '09. Boston, MA, USA: ACM, 2009, pp. 3507–3508. ISBN: 978-1-60558-247-4. DOI: 10.1145/1520340.1520515. URL: http://doi.acm.org/10.1145/1520340.1520515.

[5] *Arduino : an open-source electronics prototyping platform.* http://arduino.cc/. URL: http://arduino.cc/.

[6] Hugh Dutton Associés. *Developable Surfaces.* http://complexitys.com/english/geometry/developable-surfaces/.

[7] *AutoCAD for Mac and Windows.* https://www.autodesk.com/products/autocad/overview.

[8] Daniel Avrahami and Scott E. Hudson. "Forming Interactivity: A Tool for Rapid Prototyping of Physical Interactive Products". In: *Proceedings of the 4th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques.* DIS '02. London, England: ACM, 2002, pp. 141–146. ISBN: 1-58113-515-7. DOI: 10.1145/778712.778735. URL: http://doi.acm.org/10.1145/778712.778735.

[9] Moritz Bächer et al. "Fabricating Articulated Characters from Skinned Meshes". In: *ACM Trans. Graph.* 31.4 (July 2012), 47:1–47:9. ISSN: 0730-0301. DOI: 10.1145/2185520.2185543. URL: http://doi.acm.org/10.1145/2185520.2185543.

[10] Gill Barequet and Micha Sharir. "Filling gaps in the boundary of a polyhedron". In: *Computer Aided Geometric Design* 12.2 (1995), pp. 207–229. ISSN: 0167-8396. DOI: http://dx.doi.org/10.1016/0167-8396(94)00011-G. URL: http://www.sciencedirect.com/science/article/pii/016783969400011G.

[11]   Patrick Baudisch and Gerry Chu. "Back-of-device Interaction Allows Creating Very Small Touch Devices". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. Boston, MA, USA: ACM, 2009, pp. 1923–1932. ISBN: 978-1-60558-246-7. DOI: `10.1145/1518701.1518995`. URL: `http://doi.acm.org/10.1145/1518701.1518995`.

[12]   Dustin Beyer et al. "Platener: Low-Fidelity Fabrication of 3D Objects by Substituting 3D Print with Laser-Cut Plates". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: ACM, 2015, pp. 1799–1806. ISBN: 978-1-4503-3145-6. DOI: `10.1145/2702123.2702225`. URL: `http://doi.acm.org/10.1145/2702123.2702225`.

[13]   Bernd Bickel et al. "Design and Fabrication of Materials with Desired Deformation Behavior". In: *ACM Trans. Graph.* 29.4 (July 2010), 63:1–63:10. ISSN: 0730-0301. DOI: `10.1145/1778765.1778800`. URL: `http://doi.acm.org/10.1145/1778765.1778800`.

[14]   John E Bigelow. *Capacitive touch control and display.* 1981. URL: `https://www.google.com/patents/US4264903`.

[15]   Jan Helge Bøhn and Michael J. Wozny. "Automatic CAD-model Repair: Shell-Closure". In: *Solid Freeform Fabrication Proceedings*. University of Texas at Austin, 1992, pp. 154–160.

[16]   Stuart Brown. "Simulation of solid freeform fabrication". In: *Solid Freeform Fabrication Proceedings*. University of Texas at Austin, 1993, pp. 154–160.

[17]   N G de Bruijn. "A combinatorial problem". In: *Koninklijke Nederlandse Akademie v. Wetenschappen 49*. Indagationes Mathematicae, 1946, pp. 758–764.

[18]   Leah Buechley et al. "The LilyPad Arduino: Using Computational Textiles to Investigate Engagement, Aesthetics, and Diversity in Computer Science Education". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Florence, Italy: ACM, 2008, pp. 423–432. ISBN: 978-1-60558-011-1. DOI: `10.1145/1357054.1357123`. URL: `http://doi.acm.org/10.1145/1357054.1357123`.

[19]   Jacques Calì et al. "3D-printing of Non-assembly, Articulated Models". In: *ACM Trans. Graph.* 31.6 (Nov. 2012), 130:1–130:8. ISSN: 0730-0301. DOI: `10.1145/2366145.2366149`. URL: `http://doi.acm.org/10.1145/2366145.2366149`.

[20]   Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. "A Morphological Analysis of the Design Space of Input Devices". In: *ACM Trans. Inf. Syst.* 9.2 (Apr. 1991), pp. 99–122. ISSN: 1046-8188. DOI: `10.1145/123078.128726`. URL: `http://doi.acm.org/10.1145/123078.128726`.

[21]   Duygu Ceylan et al. "Designing and Fabricating Mechanical Automata from Mocap Sequences". In: *ACM Trans. Graph.* 32.6 (Nov. 2013), 186:1–186:11. ISSN: 0730-0301. DOI: `10.1145/2508363.2508400`. URL: `http://doi.acm.org/10.1145/2508363.2508400`.

[22] Liwei Chan et al. "CapStones and ZebraWidgets: Sensing Stacks of Building Blocks, Dials and Sliders on Capacitive Touch Screens". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '12. Austin, Texas, USA: ACM, 2012, pp. 2189–2192. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2208371. URL: http://doi.acm.org/10.1145/2207676.2208371.

[23] Tzuwen Chang et al. "Clip-on Gadgets: Expandable Tactile Controls for Multi-touch Devices". In: *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services Companion*. MobileHCI '12. San Francisco, California, USA: ACM, 2012, pp. 163–166. ISBN: 978-1-4503-1443-5. DOI: 10.1145/2371664.2371699. URL: http://doi.acm.org/10.1145/2371664.2371699.

[24] Chin-yu Chien et al. "FlexiBend: Enabling Interactivity of Multi-Part, Deformable Fabrications Using Single Shape-Sensing Strip". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. UIST '15. Daegu, Kyungpook, Republic of Korea: ACM, 2015, pp. 659–663. ISBN: 978-1-4503-3779-3. DOI: 10.1145/2807442.2807456. URL: http://doi.acm.org/10.1145/2807442.2807456.

[25] Brett G. Compton and Jennifer A. Lewis. "3D-Printing of Lightweight Cellular Composites". In: *Advanced Materials* 26.34 (2014), pp. 5930–5935. ISSN: 1521-4095. DOI: 10.1002/adma.201401804. URL: http://dx.doi.org/10.1002/adma.201401804.

[26] Tanja Doering et al. "Design by Physical Composition for Complex Tangible User Interfaces". In: *CHI '10 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '10. Atlanta, Georgia, USA: ACM, 2010, pp. 3541–3546. ISBN: 978-1-60558-930-5. DOI: 10.1145/1753846.1754015. URL: http://doi.acm.org/10.1145/1753846.1754015.

[27] Jerry Fails and Dan Olsen. "A Design Tool for Camera-based Interaction". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '03. Ft. Lauderdale, Florida, USA: ACM, 2003, pp. 449–456. ISBN: 1-58113-630-7. DOI: 10.1145/642611.642690. URL: http://doi.acm.org/10.1145/642611.642690.

[28] Claude Fleury and Vincent Braibant. "Structural optimization: A new dual method using mixed variables". In: *International Journal for Numerical Methods in Engineering* 23.3 (1986), pp. 409–428. ISSN: 1097-0207. DOI: 10.1002/nme.1620230307. URL: http://dx.doi.org/10.1002/nme.1620230307.

[29] Sean Follmer, Björn Hartmann, and Pat Hanrahan. "Input Devices are like Onions: A Layered Framework for Guiding Device Designers". In: *Workshop of CHI*. 2009.

[30] Jacob Fraden. *Handbook of Modern Sensors*. 4th ed. New York, NY, USA: Springer, 2010. ISBN: 978-1-4419-6465-6.

[31] Krzysztof Z Gajos, Jacob O Wobbrock, and Daniel S Weld. "Automatically generating user interfaces adapted to users' motor and vision capabilities". In: *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM. 2007, pp. 231–240.

[32] Krzysztof Gajos and Daniel S Weld. "SUPPLE: Automatically Generating User Interfaces". In: *Proceedings of the 9th International Conference on Intelligent User Interfaces*. IUI '04. Funchal, Madeira, Portugal: ACM, 2004, pp. 93–100. ISBN: 1-58113-815-6. DOI: 10.1145/964442.964461. URL: http://doi.acm.org/10.1145/964442.964461.

[33] *GitHub: Where software is built*. https://github.com/. Accessed: 2015-10-20.

[34] Saul Greenberg and Chester Fitchett. "Phidgets: Easy Development of Physical Interfaces Through Physical Widgets". In: *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*. UIST '01. Orlando, Florida: ACM, 2001, pp. 209–218. ISBN: 1-58113-438-X. DOI: 10.1145/502348.502388. URL: http://doi.acm.org/10.1145/502348.502388.

[35] Ankit Gupta et al. "DuploTrack: A Real-time System for Authoring and Guiding Duplo Block Assembly". In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST '12. Cambridge, Massachusetts, USA: ACM, 2012, pp. 389–402. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380167. URL: http://doi.acm.org/10.1145/2380116.2380167.

[36] E. Levent Gursoz, Lee E. Weiss, and Fritz B. Prinz. "Geometric Modeling for Rapid Prototyping and Tool Fabrication". In: *Solid Freeform Fabrication Proceedings*. University of Texas at Austin, 1990, pp. 135–145.

[37] Chris Harrison and Scott E. Hudson. "Providing Dynamically Changeable Physical Buttons on a Visual Display". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. Boston, MA, USA: ACM, 2009, pp. 299–308. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518749. URL: http://doi.acm.org/10.1145/1518701.1518749.

[38] Chris Harrison and Scott E. Hudson. "Scratch Input: Creating Large, Inexpensive, Unpowered and Mobile Finger Input Surfaces". In: *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*. UIST '08. Monterey, CA, USA: ACM, 2008, pp. 205–208. ISBN: 978-1-59593-975-3. DOI: 10.1145/1449715.1449747. URL: http://doi.acm.org/10.1145/1449715.1449747.

[39] Chris Harrison, Desney Tan, and Dan Morris. "Skinput: Appropriating the Body As an Input Surface". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: ACM, 2010, pp. 453–462. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753394. URL: http://doi.acm.org/10.1145/1753326.1753394.

[40] Chris Harrison, Robert Xiao, and Scott Hudson. "Acoustic Barcodes: Passive, Durable and Inexpensive Notched Identification Tags". In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST '12. Cambridge, Massachusetts, USA: ACM, 2012, pp. 563–568. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380187. URL: http://doi.acm.org/10.1145/2380116.2380187.

[41] Björn Hartmann et al. "Authoring Sensor-based Interactions by Demonstration with
      Direct Manipulation and Pattern Recognition". In: *Proceedings of the SIGCHI Con-
      ference on Human Factors in Computing Systems*. CHI '07. San Jose, California, USA:
      ACM, 2007, pp. 145–154. ISBN: 978-1-59593-593-9. DOI: `10.1145/1240624.1240646`.
      URL: `http://doi.acm.org/10.1145/1240624.1240646`.

[42] Björn Hartmann et al. "Reflective Physical Prototyping Through Integrated Design,
      Test, and Analysis". In: *Proceedings of the 19th Annual ACM Symposium on User
      Interface Software and Technology*. UIST '06. Montreux, Switzerland: ACM, 2006,
      pp. 299–308. ISBN: 1-59593-313-1. DOI: `10.1145/1166253.1166300`. URL: `http://doi.`
      `acm.org/10.1145/1166253.1166300`.

[43] Miloš Hašan et al. "Physical Reproduction of Materials with Specified Subsurface
      Scattering". In: vol. 29. 4. New York, NY, USA: ACM, July 2010, 61:1–61:10. DOI:
      `10.1145/1778765.1778798`. URL: `http://doi.acm.org/10.1145/1778765.1778798`.

[44] David Holman and Hrvoje Benko. "SketchSpace: Designing Interactive Behaviors with
      Passive Materials". In: *CHI '11 Extended Abstracts on Human Factors in Computing
      Systems*. CHI EA '11. Vancouver, BC, Canada: ACM, 2011, pp. 1987–1992. ISBN:
      978-1-4503-0268-5. DOI: `10.1145/1979742.1979867`. URL: `http://doi.acm.org/10.`
      `1145/1979742.1979867`.

[45] David Holman and Roel Vertegaal. "TactileTape: Low-cost Touch Sensing on Curved
      Surfaces". In: *Proceedings of the 24th Annual ACM Symposium Adjunct on User In-
      terface Software and Technology*. UIST '11 Adjunct. Santa Barbara, California, USA:
      ACM, 2011, pp. 17–18. ISBN: 978-1-4503-1014-7. DOI: `10.1145/2046396.2046406`. URL:
      `http://doi.acm.org/10.1145/2046396.2046406`.

[46] Jonathan Hook et al. "Making 3D Printed Objects Interactive Using Wireless Ac-
      celerometers". In: *CHI '14 Extended Abstracts on Human Factors in Computing Sys-
      tems*. CHI EA '14. Toronto, Ontario, Canada: ACM, 2014, pp. 1435–1440. ISBN:
      978-1-4503-2474-8. DOI: `10.1145/2559206.2581137`. URL: `http://doi.acm.org/10.`
      `1145/2559206.2581137`.

[47] Stephanie Houde and Charles Hill. "What do Prototypes Prototype?" In: *Handbook
      of Human-Computer Interaction*. Ed. by M. Helander, T. Landauer, and P. Prabhu.
      2nd ed. Amsterdam: Elsevier Science B., 1997. Chap. 16, pp. 367–381.

[48] Michael B. Hsu. "Numerical Simulation of Viscous Sintering under Mechanical Loads".
      In: *Solid Freeform Fabrication Proceedings*. University of Texas at Austin, 1992,
      pp. 154–160.

[49] Scott E. Hudson and Jennifer Mankoff. "Rapid Construction of Functioning Physical
      Interfaces from Cardboard, Thumbtacks, Tin Foil and Masking Tape". In: *Proceedings
      of the 19th Annual ACM Symposium on User Interface Software and Technology*.
      UIST '06. Montreux, Switzerland: ACM, 2006, pp. 289–298. ISBN: 1-59593-313-1. DOI:
      `10.1145/1166253.1166299`. URL: `http://doi.acm.org/10.1145/1166253.1166299`.

[50] Scott E. Hudson et al. "Whack Gestures: Inexact and Inattentive Interaction with Mobile Devices". In: *Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction*. TEI '10. Cambridge, Massachusetts, USA: ACM, 2010, pp. 109–112. ISBN: 978-1-60558-841-4. DOI: `10.1145/1709886.1709906`. URL: `http://doi.acm.org/10.1145/1709886.1709906`.

[51] *Java AWT Robots API*. `http://docs.oracle.com/javase/$\delimiter"026E30F$$\delimiter"026E30F$7/docs/api/java/awt/Robot.html`.

[52] Bonnie E. John et al. "Predictive Human Performance Modeling Made Easy". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '04. Vienna, Austria: ACM, 2004, pp. 455–462. ISBN: 1-58113-702-8. DOI: `10.1145/985692.985750`. URL: `http://doi.acm.org/10.1145/985692.985750`.

[53] Kanav Kahol et al. "Measuring Movement Expertise in Surgical Tasks". In: *Proceedings of the 14th ACM International Conference on Multimedia*. MM '06. Santa Barbara, CA, USA: ACM, 2006, pp. 719–722. ISBN: 1-59593-447-2. DOI: `10.1145/1180639.1180792`. URL: `http://doi.acm.org/10.1145/1180639.1180792`.

[54] Scott R. Klemmer, Björn Hartmann, and Leila Takayama. "How Bodies Matter: Five Themes for Interaction Design". In: *Proceedings of the 6th Conference on Designing Interactive Systems*. DIS '06. University Park, PA, USA: ACM, 2006, pp. 140–149. ISBN: 1-59593-367-0. DOI: `10.1145/1142405.1142429`. URL: `http://doi.acm.org/10.1145/1142405.1142429`.

[55] Scott R Klemmer et al. "Papier-Mâché: toolkit support for tangible input". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '04. Vienna, Austria: ACM, 2004, pp. 399–406. ISBN: 1-58113-702-8. DOI: `10.1145/985692.985743`. URL: `http://doi.acm.org/10.1145/985692.985743`.

[56] Yanxiang Lan et al. "Bi-scale Appearance Fabrication". In: *ACM Trans. Graph.* 32.4 (July 2013), 145:1–145:12. ISSN: 0730-0301. DOI: `10.1145/2461912.2461989`. URL: `http://doi.acm.org/10.1145/2461912.2461989`.

[57] Gierad Laput et al. "Acoustruments: Passive, Acoustically-Driven, Interactive Controls for Handheld Devices". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: ACM, 2015, pp. 2161–2170. ISBN: 978-1-4503-3145-6. DOI: `10.1145/2702123.2702414`. URL: `http://doi.acm.org/10.1145/2702123.2702414`.

[58] Manfred Lau et al. "Modeling-in-context: User Design of Complementary Objects with a Single Photo". In: *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*. SBIM '10. Annecy, France: Eurographics Association, 2010, pp. 17–24. ISBN: 978-3-905674-25-5. URL: `http://dl.acm.org/citation.cfm?id=1923363.1923367`.

[59] C Y Lee. "An Algorithm for Path Connections and Its Applications". In: *IRE Transactions on Electronic Computers* 10.2 (1961), pp. 346–365.

[60] Jay Lee et al. "HandSCAPE: A Vectorizing Tape Measure for On-site Measuring Applications". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '00. The Hague, The Netherlands: ACM, 2000, pp. 137–144. ISBN: 1-58113-216-6. DOI: `10.1145/332040.332417`. URL: `http://doi.acm.org/10.1145/332040.332417`.

[61] Johnny C. Lee et al. "The Calder Toolkit: Wired and Wireless Components for Rapidly Prototyping Interactive Devices". In: *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*. DIS '04. Cambridge, MA, USA: ACM, 2004, pp. 167–175. ISBN: 1-58113-787-7. DOI: `10.1145/1013115.1013139`. URL: `http://doi.acm.org/10.1145/1013115.1013139`.

[62] Yang Li et al. "FrameWire: A Tool for Automatically Extracting Interaction Logic from Paper Prototyping Tests". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: ACM, 2010, pp. 503–512. ISBN: 978-1-60558-929-9. DOI: `10.1145/1753326.1753401`. URL: `http://doi.acm.org/10.1145/1753326.1753401`.

[63] *littleBits: DIY Electronics For Prototyping and Learning*. `http://littlebits.cc`.

[64] Blair MacIntyre et al. "DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences". In: *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*. UIST '04. Santa Fe, NM, USA: ACM, 2004, pp. 197–206. ISBN: 1-58113-957-8. DOI: `10.1145/1029632.1029669`. URL: `http://doi.acm.org/10.1145/1029632.1029669`.

[65] C Majidi, R Kramer, and R J Wood. "A non-differential elastomer curvature sensor for softer-than-skin electronics". In: *Smart Materials and Structures* 20.10 (2011), p. 105017. URL: `http://stacks.iop.org/0964-1726/20/i=10/a=105017`.

[66] Dan Maynes-Aminzade, Terry Winograd, and Takeo Igarashi. "Eyepatch: Prototyping Camera-based Interaction Through Examples". In: *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*. UIST '07. Newport, Rhode Island, USA: ACM, 2007, pp. 33–42. ISBN: 978-1-59593-679-0. DOI: `10.1145/1294211.1294219`. URL: `http://doi.acm.org/10.1145/1294211.1294219`.

[67] Leonard Meirovitch. *Analytical Methods in Vibration*. The MacMillan Company, 1967.

[68] David Molyneaux. "KinectFusion Rapid 3D Reconstruction and Interaction with Microsoft Kinect". In: *Proceedings of the International Conference on the Foundations of Digital Games*. FDG '12. Raleigh, North Carolina: ACM, 2012, pp. 3–3. ISBN: 978-1-4503-1333-9. DOI: `10.1145/2282338.2282342`. URL: `http://doi.acm.org/10.1145/2282338.2282342`.

[69] Yuki Mori and Takeo Igarashi. "Plushie: An Interactive Design System for Plush Toys". In: *ACM Trans. Graph.* 26.3 (July 2007). ISSN: 0730-0301. DOI: `10.1145/1276377.1276433`. URL: `http://doi.acm.org/10.1145/1276377.1276433`.

[70] Catarina Mota. "The Rise of Personal Fabrication". In: *Proceedings of the 8th ACM Conference on Creativity and Cognition*. C&C '11. Atlanta, Georgia, USA: ACM, 2011, pp. 279–288. ISBN: 978-1-4503-0820-5. DOI: 10.1145/2069618.2069665. URL: http://doi.acm.org/10.1145/2069618.2069665.

[71] Stefanie Mueller, Bastian Kruck, and Patrick Baudisch. "LaserOrigami: Laser-cutting 3D Objects". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. Paris, France: ACM, 2013, pp. 2585–2592. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2481358. URL: http://doi.acm.org/10.1145/2470654.2481358.

[72] Stefanie Mueller, Pedro Lopes, and Patrick Baudisch. "Interactive Construction: Interactive Fabrication of Functional Mechanical Devices". In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST '12. Cambridge, Massachusetts, USA: ACM, 2012, pp. 599–606. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380191. URL: http://doi.acm.org/10.1145/2380116.2380191.

[73] Stefanie Mueller et al. "faBrickation: Fast 3D Printing of Functional Objects by Integrating Construction Kit Building Blocks". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: ACM, 2014, pp. 3827–3834. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557005. URL: http://doi.acm.org/10.1145/2556288.2557005.

[74] Roderick Murray-Smith et al. "Stane: Synthesized Surfaces for Tactile Input". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Florence, Italy: ACM, 2008, pp. 1299–1302. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357257. URL: http://doi.acm.org/10.1145/1357054.1357257.

[75] Brad A. Myers and William Buxton. "Creating Highly-interactive and Graphical User Interfaces by Demonstration". In: *SIGGRAPH Comput. Graph.* 20.4 (Aug. 1986), pp. 249–258. ISSN: 0097-8930. DOI: 10.1145/15886.15914. URL: http://doi.acm.org/10.1145/15886.15914.

[76] Tek-Jin Nam. "Sketch-based Rapid Prototyping Platform for Hardware-software Integrated Interactive Products". In: *CHI '05 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '05. Portland, OR, USA: ACM, 2005, pp. 1689–1692. ISBN: 1-59593-002-7. DOI: 10.1145/1056808.1056998. URL: http://doi.acm.org/10.1145/1056808.1056998.

[77] Tek-Jin Nam and Woohun Lee. "Integrating Hardware and Software: Augmented Reality Based Prototyping Method for Digital Products". In: *CHI '03 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '03. Ft. Lauderdale, Florida, USA: ACM, 2003, pp. 956–957. ISBN: 1-58113-637-4. DOI: 10.1145/765891.766092. URL: http://doi.acm.org/10.1145/765891.766092.

[78] Misael Navarrete et al. "Integrated layered manufacturing of a novel wireless motion sensor system with GPS". In: *Technical Report, University of Texas at El Paso* (2007).

[79] B B C News. *Playing pop music via paper posters with conductive ink.* http://www.bbc.com/news/tec 17339512. 2012.

[80] Simon Olberding, Michael Wessely, and Jürgen Steimle. "PrintScreen: Fabricating Highly Customizable Thin-film Touch-displays". In: *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology.* UIST '14. Honolulu, Hawaii, USA: ACM, 2014, pp. 281–290. ISBN: 978-1-4503-3069-5. DOI: `10.1145/2642918.2647413`. URL: `http://doi.acm.org/10.1145/2642918.2647413`.

[81] Simon Olberding et al. "A Cuttable Multi-touch Sensor". In: *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology.* UIST '13. St. Andrews, Scotland, United Kingdom: ACM, 2013, pp. 245–254. ISBN: 978-1-4503-2268-3. DOI: `10.1145/2501988.2502048`. URL: `http://doi.acm.org/10.1145/2501988.2502048`.

[82] Makoto Ono, Buntarou Shizuki, and Jiro Tanaka. "Touch & Activate: Adding Interactivity to Existing Objects Using Active Acoustic Sensing". In: *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology.* UIST '13. St. Andrews, Scotland, United Kingdom: ACM, 2013, pp. 31–40. ISBN: 978-1-4503-2268-3. DOI: `10.1145/2501988.2501989`. URL: `http://doi.acm.org/10.1145/2501988.2501989`.

[83] *OpenCV : open source computer vision.* `http://opencv.org/`.

[84] *OpenFrameworks : an open-source C++ toolkit for creative coding.* `http://www.openframeworks.cc/`.

[85] Yong-Lae Park, Bor-Rong Chen, and Robert J Wood. "Design and Fabrication of Soft Artificial Skin Using Embedded Microchannels and Liquid Conductors". In: *IEEE Sensors Journal* 12.8 (Aug. 2012), pp. 2711–2718.

[86] Kai Parthy. *Reflective Filament.* `https://www.youtube.com/watch?v=Xh5bIgIoRA0`.

[87] Hal Philipp. "Charge Transfer Sensing". In: *Sensor Review* 19.2 (1999), pp. 96–105.

[88] *Pinball Paper Toys.* `http://www.matica.com/papercraft-toys/61/Classic-Miniature-Pinball-Table-Paper-ToyModels.html`.

[89] Ivan Poupyrev, Chris Harrison, and Munehiko Sato. "TouchÉ: Touch and Gesture Sensing for the Real World". In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing.* UbiComp '12. Pittsburgh, Pennsylvania: ACM, 2012, pp. 536–536. ISBN: 978-1-4503-1224-0. DOI: `10.1145/2370216.2370296`. URL: `http://doi.acm.org/10.1145/2370216.2370296`.

[90] Romain Prévost et al. "Make It Stand: Balancing Shapes for 3D Fabrication". In: *ACM Trans. Graph.* 32.4 (July 2013), 81:1–81:10. ISSN: 0730-0301. DOI: `10.1145/2461912.2461957`. URL: `http://doi.acm.org/10.1145/2461912.2461957`.

[91] *Pro Engineer.* `https://www.ptc.com/`.

[92] Raf Ramakers, Kashyap Todi, and Kris Luyten. "PaperPulse: An Integrated Approach for Embedding Electronics in Paper Designs". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: ACM, 2015, pp. 2457–2466. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702487. URL: http://doi.acm.org/10.1145/2702123.2702487.

[93] Mitchel Resnick et al. "Scratch: Programming for All". In: *Commun. ACM* 52.11 (Nov. 2009), pp. 60–67. ISSN: 0001-0782. DOI: 10.1145/1592761.1592779. URL: http://doi.acm.org/10.1145/1592761.1592779.

[94] *Rhinoceros*. https://www.rhino3d.com/.

[95] John Sarik, Chao (Timmy) Li, and Ioannis Kymissis. "Fabricating Electronics with Rapid Prototyping Tools". In: *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*. TEI '12. Kingston, Ontario, Canada: ACM, 2012, pp. 339–342. ISBN: 978-1-4503-1174-8. DOI: 10.1145/2148131.2148212. URL: http://doi.acm.org/10.1145/2148131.2148212.

[96] Ayaka Sato, Keita Watanabe, and Jun Rekimoto. "MimiCook: A Cooking Assistant System with Situated Guidance". In: *Proceedings of the 8th International Conference on Tangible, Embedded and Embodied Interaction*. TEI '14. Munich, Germany: ACM, 2013, pp. 121–124. ISBN: 978-1-4503-2635-3. DOI: 10.1145/2540930.2540952. URL: http://doi.acm.org/10.1145/2540930.2540952.

[97] Greg Saul et al. "SketchChair: An All-in-one Chair Design System for End Users". In: *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction*. TEI '11. Funchal, Portugal: ACM, 2011, pp. 73–80. ISBN: 978-1-4503-0478-8. DOI: 10.1145/1935701.1935717. URL: http://doi.acm.org/10.1145/1935701.1935717.

[98] Valkyrie Savage, Colin Chang, and Björn Hartmann. "Sauron: Embedded Single-camera Sensing of Printed Physical User Interfaces". In: *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. UIST '13. St. Andrews, Scotland, United Kingdom: ACM, 2013, pp. 447–456. ISBN: 978-1-4503-2268-3. DOI: 10.1145/2501988.2501992. URL: http://doi.acm.org/10.1145/2501988.2501992.

[99] Valkyrie Savage, Xiaohan Zhang, and Björn Hartmann. "Midas: Fabricating Custom Capacitive Touch Sensors to Prototype Interactive Objects". In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST '12. Cambridge, Massachusetts, USA: ACM, 2012, pp. 579–588. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380189. URL: http://doi.acm.org/10.1145/2380116.2380189.

[100] Valkyrie Savage et al. "A Series of Tubes: Adding Interactivity to 3D Prints Using Internal Pipes". In: UIST '14 (2014), pp. 3–12. DOI: 10.1145/2642918.2647374. URL: http://doi.acm.org/10.1145/2642918.2647374.

[101] Valkyrie Savage et al. "Lamello: Passive Acoustic Sensing for Tangible Input Components". In: CHI '15 (2015), pp. 1277–1280. DOI: 10.1145/2702123.2702207. URL: http://doi.acm.org/10.1145/2702123.2702207.

[102] Valkyrie Savage et al. "Makers' Marks: Physical Markup for Designing and Fabricating Functional Objects". In: UIST '15 (2015), pp. 103–108. DOI: 10.1145/2807442. 2807508. URL: http://doi.acm.org/10.1145/2807442.2807508.

[103] Ryan Schmidt and Karan Singh. "Meshmixer: An Interface for Rapid Mesh Composition". In: *ACM SIGGRAPH 2010 Talks*. SIGGRAPH '10. Los Angeles, California: ACM, 2010, 6:1–6:1. ISBN: 978-1-4503-0394-1. DOI: 10.1145/1837026.1837034. URL: http://doi.acm.org/10.1145/1837026.1837034.

[104] Ryan Schmidt and Nobuyuki Umetani. "Branching Support Structures for 3D Printing". In: *ACM SIGGRAPH 2014 Studio*. Vancouver, Canada: ACM, 2014, 9:1–9:1. ISBN: 978-1-4503-2977-4. DOI: 10.1145/2619195.2656293. URL: http://doi.acm.org/ 10.1145/2619195.2656293.

[105] Martin Schmitz et al. "Capricate: A Fabrication Pipeline to Design and 3D Print Capacitive Touch Sensors for Interactive Objects". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. UIST '15. Daegu, Kyungpook, Republic of Korea: ACM, 2015, pp. 253–258. ISBN: 978-1-4503-3779-3. DOI: 10.1145/2807442.2807503. URL: http://doi.acm.org/10.1145/2807442. 2807503.

[106] Ed Sells. *Rapid Prototyped Electronic Circuits*. http://fennetic.net/irc/reprap_ circuits.pdf. 2004.

[107] *Simplelink SensorTag*. http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/ tearDown.html.

[108] Pitchaya Sitthi-Amorn et al. "MultiFab: a machine vision assisted platform for multi-material 3D printing". In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), p. 129.

[109] Ronit Slyper and Jessica Hodgins. "Prototyping Robot Appearance, Movement, and Interactions Using Flexible 3D Printing and Air Pressure Sensors". In: *Proceedings of the 21st International Symposium on Robot and Human Interactive Communication*. Ro-Man 2012 (2012), pp. 6–11.

[110] Ronit Slyper, Ivan Poupyrev, and Jessica Hodgins. "Sensing Through Structure: Designing Soft Silicone Sensors". In: *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction*. TEI '11. Funchal, Portugal: ACM, 2011, pp. 213–220. ISBN: 978-1-4503-0478-8. DOI: 10.1145/1935701.1935744. URL: http://doi.acm.org/10.1145/1935701.1935744.

[111] Pieter Smakman et al. *Curatio: World's First Dedicated 3D Hand Scanner*. http: //www.core77.com/projects/45809/Curatio-Worlds-First-Dedicated-3D-Hand-Scanner.

[112] Hyunyoung Song, François Guimbretière, and Hod Lipson. "The ModelCraft Framework: Capturing Freehand Annotations and Edits to Facilitate the 3D Model Design Process Using a Digital Pen". In: *ACM Trans. Comput.-Hum. Interact.* 16.3 (Sept. 2009), 14:1–14:33. ISSN: 1073-0516. DOI: 10.1145/1592440.1592443. URL: http://doi.acm.org/10.1145/1592440.1592443.

[113] *Sophie's super hand.* http://engineering.berkeley.edu/magazine/fall-2015/sophies-super-hand.

[114] *Teensy USB Development Board.* http://www.pjrc.com/teensy/.

[115] Skylar Tibbits. "Design to Self-Assembly". In: *Architectural Design* 82.2 (2012), pp. 68–73. ISSN: 1554-2769. DOI: 10.1002/ad.1381. URL: http://dx.doi.org/10.1002/ad.1381.

[116] Cesar Torres et al. "HapticPrint: Designing Feel Aesthetics for Digital Fabrication". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology.* UIST '15. Daegu, Kyungpook, Republic of Korea: ACM, 2015, pp. 583–591. ISBN: 978-1-4503-3779-3. DOI: 10.1145/2807442.2807492. URL: http://doi.acm.org/10.1145/2807442.2807492.

[117] Nobuyuki Umetani, Jun Mitani, and Takeo Igarashi. "Designing Custom-made Metallophone with Concurrent Eigenanalysis". In: *Proceedings of NIME '10.*

[118] Nobuyuki Umetani and Ryan Schmidt. "Cross-sectional Structural Analysis for 3D Printing Optimization". In: *SIGGRAPH Asia 2013 Technical Briefs.* SA '13. Hong Kong, Hong Kong: ACM, 2013, 5:1–5:4. ISBN: 978-1-4503-2629-2. DOI: 10.1145/2542355.2542361. URL: http://doi.acm.org/10.1145/2542355.2542361.

[119] Nicolas Villar and Hans Gellersen. "A Malleable Control Structure for Softwired User Interfaces". In: *Proceedings of the 1st International Conference on Tangible and Embedded Interaction.* TEI '07. Baton Rouge, Louisiana: ACM, 2007, pp. 49–56. ISBN: 978-1-59593-619-6. DOI: 10.1145/1226969.1226980. URL: http://doi.acm.org/10.1145/1226969.1226980.

[120] Nicolas Villar et al. ".NET Gadgeteer: A Platform for Custom Devices". In: *Proceedings of the 10th International Conference on Pervasive Computing.* Pervasive'12. Newcastle, UK: Springer-Verlag, 2012, pp. 216–233. ISBN: 978-3-642-31204-5. DOI: 10.1007/978-3-642-31205-2_14. URL: http://dx.doi.org/10.1007/978-3-642-31205-2_14.

[121] *Voxel8: 3D Electronics Printing.* http://www.voxel8.co/.

[122] Christian Weichel, Manfred Lau, and Hans Gellersen. "Enclosed: A Component-centric Interface for Designing Prototype Enclosures". In: *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction.* TEI '13. Barcelona, Spain: ACM, 2013, pp. 215–218. ISBN: 978-1-4503-1898-3. DOI: 10.1145/2460625.2460659. URL: http://doi.acm.org/10.1145/2460625.2460659.

[123] Christian Weichel et al. "MixFab: A Mixed-reality Environment for Personal Fabrication". In: *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems.* CHI '14. Toronto, Ontario, Canada: ACM, 2014, pp. 3855–3864. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557090. URL: http://doi.acm.org/10.1145/2556288.2557090.

[124] Christian Weichel et al. "SPATA: Spatio-Tangible Tools for Fabrication-Aware Design". In: *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction.* TEI '15. Stanford, California, USA: ACM, 2015, pp. 189–196. ISBN: 978-1-4503-3305-4. DOI: 10.1145/2677199.2680576. URL: http://doi.acm.org/10.1145/2677199.2680576.

[125] Tim Weyrich et al. "Fabricating Microgeometry for Custom Surface Reflectance". In: *ACM Trans. Graph.* 28.3 (July 2009), 32:1–32:6. ISSN: 0730-0301. DOI: 10.1145/1531326.1531338. URL: http://doi.acm.org/10.1145/1531326.1531338.

[126] Karl D. D. Willis and Andrew D. Wilson. "InfraStructs: Fabricating Information Inside Physical Objects for Imaging in the Terahertz Region". In: *ACM Trans. Graph.* 32.4 (July 2013), 138:1–138:10. ISSN: 0730-0301. DOI: 10.1145/2461912.2461936. URL: http://doi.acm.org/10.1145/2461912.2461936.

[127] Karl D.D. Willis et al. "Interactive Fabrication: New Interfaces for Digital Fabrication". In: *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction.* TEI '11. Funchal, Portugal: ACM, 2011, pp. 69–72. ISBN: 978-1-4503-0478-8. DOI: 10.1145/1935701.1935716. URL: http://doi.acm.org/10.1145/1935701.1935716.

[128] Karl Willis et al. "Printed Optics: 3D Printing of Embedded Optical Elements for Interactive Devices". In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology.* UIST '12. Cambridge, Massachusetts, USA: ACM, 2012, pp. 589–598. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380190. URL: http://doi.acm.org/10.1145/2380116.2380190.

[129] Andrew D. Wilson. "Using a Depth Camera As a Touch Sensor". In: *ACM International Conference on Interactive Tabletops and Surfaces.* ITS '10. Saarbrücken, Germany: ACM, 2010, pp. 69–72. ISBN: 978-1-4503-0399-6. DOI: 10.1145/1936652.1936665. URL: http://doi.acm.org/10.1145/1936652.1936665.

[130] Raphael Wimmer and Patrick Baudisch. "Modular and Deformable Touch-sensitive Surfaces Based on Time Domain Reflectometry". In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology.* UIST '11. Santa Barbara, California, USA: ACM, 2011, pp. 517–526. ISBN: 978-1-4503-0716-1. DOI: 10.1145/2047196.2047264. URL: http://doi.acm.org/10.1145/2047196.2047264.

[131] Jacob O Wobbrock. *Capacitive EdgeWrite implementation.* http://depts.washington.edu/ewrite/capacitive.html.

[132] Jacob O. Wobbrock, Brad A. Myers, and John A. Kembel. "EdgeWrite: A Stylus-based Text Entry Method Designed for High Accuracy and Stability of Motion". In: *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*. UIST '03. Vancouver, Canada: ACM, 2003, pp. 61–70. ISBN: 1-58113-636-6. DOI: 10.1145/964696.964703. URL: http://doi.acm.org/10.1145/964696.964703.

[133] Robert Woodbury. "Elements of parametric design". In: (2010).

[134] Robert F. Woodbury and Andrew L. Burrow. "Whither design space?" In: *AIE EDAM: Artificial Intelligence for Engineering Design, Analysis, and Manufacturing* 20 (02 May 2006), pp. 63–82. ISSN: 1469-1760. DOI: 10.1017/S0890060406060057. URL: http://journals.cambridge.org/article_S0890060406060057.

[135] P.K. Wright. *21st Century Manufacturing*. Prentice Hall, 2001. ISBN: 9780130956019. URL: https://books.google.com/books?id=EwZTAAAAMAAJ.

[136] Lining Yao et al. "PneUI: Pneumatically Actuated Soft Composite Materials for Shape Changing Interfaces". In: *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. UIST '13. St. Andrews, Scotland, United Kingdom: ACM, 2013, pp. 13–22. ISBN: 978-1-4503-2268-3. DOI: 10.1145/2501988.2502037. URL: http://doi.acm.org/10.1145/2501988.2502037.

[137] Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. "Sikuli: Using GUI Screenshots for Search and Automation". In: *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*. UIST '09. Victoria, BC, Canada: ACM, 2009, pp. 183–192. ISBN: 978-1-60558-745-5. DOI: 10.1145/1622176.1622213. URL: http://doi.acm.org/10.1145/1622176.1622213.

[138] Kian Peen Yeo, Suranga Nanayakkara, and Shanaka Ransiri. "StickEar: Making Everyday Objects Respond to Sound". In: *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. UIST '13. St. Andrews, Scotland, United Kingdom: ACM, 2013, pp. 221–226. ISBN: 978-1-4503-2268-3. DOI: 10.1145/2501988.2502019. URL: http://doi.acm.org/10.1145/2501988.2502019.

[139] Amit Zoran and Joseph A. Paradiso. "FreeD: A Freehand Digital Sculpting Tool". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. Paris, France: ACM, 2013, pp. 2613–2616. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2481361. URL: http://doi.acm.org/10.1145/2470654.2481361.