# Encoding Data into Physical Objects with Digitally Fabricated Textures



by

Travis Rich

B.S., Boston University (2010)
M.S., Boston University (2011)

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013



Author .....
Program in Media Arts and Sciences,
May 6, 2013

Certified by .....
Andrew Lippman
Associate Director & Senior Research Scientist, MIT Media Lab
Thesis Supervisor

Accepted by .....
Professor Patricia Maes
Associate Academic Head, Program in Media Arts and Sciences

# Encoding Data into Physical Objects with Digitally Fabricated Textures

by

Travis Rich

Submitted to the Program in Media Arts and Sciences,,
School of Architecture and Planning
on May 18, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science in Media Arts and Sciences

## Abstract

This thesis presents and outlines a system for encoding physical passive objects with deterministic surface features that contain identifying information about that object. The goal of such work is to take steps towards a self-descriptive universe in which all objects contain within their physical structure hooks to information about how they can be used, how they can be fixed, what they're used for, who uses them, etc. By exploring modern manufacturing processes, several techniques for creating these deterministic textures are presented. Of high importance is the advancement of 3D printing technologies. By leveraging the rapid prototyping capabilities such machines offer, this thesis looks at how personalized objects and draft models may be encoded with data that allows annotations, ideas, and notes to be associated with physical points across that object. Whereas barcodes, QR codes, and RFID tags are often used to associate a single object with a single piece of data, this technique of encoding surfaces will allow for many points of identification to be placed on a single object, enabling applications in learning, group interaction, and gaming.

Thesis Supervisor: Andrew Lippman
Title: Associate Director & Senior Research Scientist, MIT Media Lab

**Encoding Data into Physical Objects with Digitally Fabricated Textures**

by

Travis Rich

The following people served as readers for this thesis:

Thesis Reader .....

.....
Patricia Maes
Professor of Media Technology
Program in Media Arts and Sciences

Thesis Reader ....

.....
V. Michael Bove, Jr.
Principal Research Scientist
Media Lab

# Acknowledgments

Like all good adventures, landing at the end of writing your thesis is filled with as much excitement for the end result as bewilderment at the path it took to get there. As twisting and windy as that path may have been, there are always hoards of people who have kept me from skidding past the sharp turns and sinking too far into the deep dives. To that end, thank you everyone who helped me get from A to B (and Z, H, K, and M in between).

The Media Lab is unlike any community I've ever been a part of and I'm extremely grateful to have been able to complete this work as part of it. Thank you everyone for the Friday teas, 99 Fridays, ping pong, free appetizers, foodcam rations, and Muddy brainstorms.

With the utmost gratitude, I thank my advisor, Andy Lippman, for bringing me to the Lab and providing amazing experiences throughout this two years Master's. I'm extremely grateful for all the support, advice, and ideas you've given me.

Thank you to my readers, Pattie Maes and Mike Bove, for helping me funnel the sparks in my head down into something worthwhile and exciting. It's been wonderful working with you.

Thanks to the Viral Spaces group (Matt Blackshaw, Kwan Hong Lee , Julia Ma, Dawei Shen, Shen Shen, Dan Sawada, Jonathan Speiser, Eyal Toledano, Deb Widener, Grace Woo, and Polychronis Ypodimatopoulos) for being in the trenches with me and giving me enough fuel for thought to last decades.

Thanks to all the MIT friends who have been there to get food, drinks, and free t-shirts for these past two years. Nothing is sadder than the fact that we don't get to always play in the same building.

In more ways than I can count, I'm indebted to my family — Dad, Mom, Josh, and Peter — for raising me to be eager and interested in exploring the world's questions. Thank you. To my non-genetic family at 9 Rollins, thanks for keeping me sane and happy.

To Grace Lin, thanks for being there every step of the way. I'm couldn't be

happier that I get to share countless adventures with you.

To all my friends inside and outside the Lab, thank you for taking me to where I am today.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Connecting all life, as we know it, is the powerful mechanism of self-encoding. The source-code for any life form is contained within nearly every part of its being. The DNA found in the skin, hair, bone, muscle, or other tissue not only completely describes the being, but it can be used to produce copies of the being. Interestingly, this fundamental paradigm that directs our very existence is not replicated in the worldly objects that we as humans produce. The objects we build and fabricate and often completely separate from the plans and designs that were used to build those objects. While this arose out of simplicity, our manufacturing and technical understanding is reaching a point where the capability of embedding our physical objects with descriptive information is simple and achievable. This is due to both advancements in the resolution with which we can fabricate objects and advancements in the resolution with which we can sense objects. This thesis explores the paradigm of encoding information into the physical structure of our tangible objects, from the manufacturing techniques and procedures to the sensing modalities and architectures.

## 1.1 Thesis Work Context and Perspective

This work was performed in the Viral Spaces group at the MIT Media Lab, led by Dr. Andrew Lippman. From its origins in viral communications and infrastructure-free radio systems, it has grown to explore mobile device ecologies, their influence on our

physical spaces, and most recently, their ability to create meaningful experiences in our everyday lives. Building on this past experience, the group has transitioned from designing communicative spaces to creative spaces. Spaces that enable experiences that are instructive, creative, and engaging. This history and bias provides a useful perspective from which to understand this work and the motives. To that end, this thesis explores beyond the domain of radios and electronics to the 'classic' part of our environments - the passive objects that don't have built in communication hardware, sensors, and batteries - in hopes of creating more instructive, creative, and engaging physical environments.

Furthermore, this thesis work strives to uphold a design paradigm in which new technologies are built to start small (providing value at that small scale) and scale efficiently (growing in value as the idea scales). For example, technologies like the internet (which provides value to the original owner of the network and grows in value as more participate) are preferred over technologies like fax machines (a technology which only provides value at large scale and is useless to the first owner). This technological design bias is maintained through much of this thesis work, and there is a strong emphasis to create a system which is useful even if it is only ever used by a single person, but would surely grow in value if it were to become well adopted.

## 1.2 A Self-descriptive Universe

The ultimate goal of this direction of research is to create a self-descriptive universe. In a self-descriptive universe, all of our objects reveal themselves to you: how they're made, who uses them, how they're used, how they can be fixed, what principles and concepts guide their function. This ties with the DNA metaphor in by taking one step further. Our objects will not only have encodings that describe themselves, as biological beings have DNA, but we will have the tools to interact with and read these encoded objects.

The idea of a self-descriptive universe can be seen, in some respect, as an extension beyond the typical internet-of-things idea. The internet-of-things is an idea that

suggests the connection of all of our gizmos; if it plugs into a wall, it should plug into the network. In such a network, my oven can talk to my toaster, which can talk to my fridge and turn on my lights which react to my alarm, all of which are controlled by my phone, for example. A self-descriptive universe extends beyond that to passive objects: my hat, my shirt, my car engine, a bike chain, etc. What would these things have to say, if we could make them talk? Perhaps they would tell you where they were made, or how people use them, or what their CAD file is so that we can 3D print another copy. They would become self-descriptive.

While this vision is decidedly longer-term than a Master's thesis, steps in the right direction can be taken. While the future of sensing and the wearable tools is unknown, we can begin by creating this self-descriptive world to leverage the sensors we currently carry - and evolve over time as need be. That said, this thesis takes a strong bias towards devising a sensing paradigm that leverages the sensors found in common mobile phones. While this introduces a level of industrial bias (perhaps the sensors Samsung has decided are best for profits are not the best technical solution for my cause), it drives an important idea that the sensing used is to be common and ubiquitous.

## 1.3 The Right Information at the Right Time

Throughout our lives we encounter many interactions that make for rich learning experiences in which an expert teaches a subject matter around a physical object. For example, a mechanic teaching an apprentice about car engines, a doctor using a model to explain an injury to a patient, or a physics teacher using demo kits to teach Newtonian mechanics. In these situations, I claim that the richness of the experience comes from two main factors: 1) having an expert that can act as a knowledge source, and 2) having the physical object present to allow specific pointing, touching, and exploration. However, these occurrences are rare and we typically do not have a knowledge expert or an object that allows direct interaction available. So the question becomes, how can we enable normal, everyday objects to afford that same experience,

same interactivity of being able to touch and either submit or consume information about that spatially specific touch point? How can we deliver the right information, at the right time? We explore these questions to create technologies that enable us to learn from what is physically around us.

The growth of the internet and the era of digitization has led to a state where much of the world's knowledge it cataloged and accessible to a large population. However, the mechanisms for accessing this data are not always convenient or even useful. Take, for example again, the scenario of trying to learn about your car engine. In this sort of scenario, it is difficult to search for the information you want as you don't know the name of the thing you're interested in. The ideal situation is to have a mechanic near-by who could tell you, based on their experience, what any given part of a car engine is when you point (an attempt to solve similar problems has been performed be Maes and Merrill [22] and will be further discussed in Chapter 3). The middle ground that this thesis tries to create is one where the digitized information of the web can be associated with the specific object through a physical encoding that is found on the object. In this way, rather than a mechanic, one could use their mobile sensors to 'read' the physical encoding and gather information about their car engine. Referring again to the DNA analogy, such interaction can be akin to the example of a biologist using DNA testing to understand the relatives and specific characteristics of millennia-old, preserved animal tissue.

In this way, delivering the right information at the right time and place (i.e. based on the context and situation a person is in) can be extraordinarily valuable for exploration and learning. The vision that I am driven by is one of a real-world wikipedia. One where the span of human knowledge regarding a physical object is not accessed through a web-browser, but through physical interaction with the item. In this way, users could stumble across information based on the new things they encounter everyday. Higher level representations could also be made. By mapping the objects a person interacts with and understands, unique learning experiences can be produced. By seeing that a person is interacting with bike chains and fishing rods, perhaps a lesson on mechanical advantage could be delivered with higher efficacy.

## 1.4 A Rational for Physical Encoding

The enabling technology of this thesis is advanced digital fabrication tools.Key advancements have led to an increase in accessibility and power of digital fabrication. With advancements in CNC cutters, laser cutters, 3D printers, and casting compounds, to name a few, it is now possible to fabricate objects with a specificity and resolution that was previously out of reach. This allows us to create deterministic textures that encode data in their physical variations.

In addition to making such fabrication physically possible, such advancements raise interesting social and community based needs. In a world where at-home 3D printing and fabrication is common place, it will be important to establish a mechanism for identifying the source and ownership of a given object. Encoding the physical object itself with identifying surface textures could be one solution. Furthermore, it may be beneficial to create a technology by which sharing designs and source models would be simple and straightforward. By encoding the surface texture of an object, one could either link a source file to the encoded data, or the encoded data itself could contain the raw CAD file source material. This would allow for people to duplicate physical 3D objects simply by scanning and reading the encoded CAD file embedded within that object, and then 3D printing the direct result. At the extreme, this approaches an ecosystem like that seen in the open-software world. Source code is easily shared, distributed, reproduced, and remixed in different contexts for different purposes.

Figure 1-1 demonstrates a cycle that is possible with an encoded surface technology. Here, the evolution of an object is cyclical and revisions can be made directly on top of the existing model by retrieving the CAD model from the object itself. This contrasts with a typical linear manufacturing chain. In such a manufacturing chain, an object is designed, fabricated, and distributed. Future iterations of that object are then only made by the original manufacturer. Slight alterations or personalized changes could not be made and fabricated by another party without first making an effort to duplicate the entire original object.

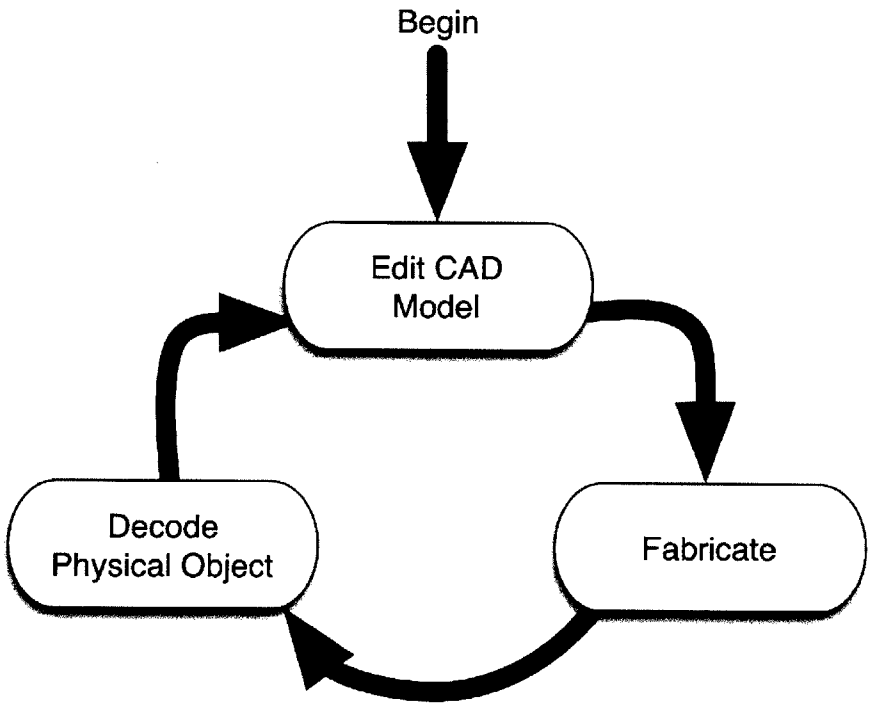


Figure 1-1: The edit, fabricate, scan cycle.

Physically encoding the surface of an object also provides a unique robustness. Given sufficient encoding resolution, broken pieces could be read and the entire object as a whole could be re-printed from the broken piece (in the same way a single cell could be used to clone an entire biological being). This could provide value in repairing larger systems. For example, finding a broken component in a home refrigerator would no longer be a cause for worry. Regardless of whether the name of the broken piece is known, it can be scanned, re-fabricated, and replaced.

Most importantly, a physical encoding means that the data is entirely contained in the source CAD file that describes this object. In this way, the software file can be distributed and shared through traditional means and every iteration of the fabricated piece will contain an identical encoding (as dictated by the source CAD file). Thus, by creating a web-connected backend interface driven by this encoding, the physical, passive objects can in a sense be networked. The physical data markings on the object can be used to link to a single community web location where comments, tips,

and details about that object can be stored.

### 1.4.1 Modalities of Sensing

When considering how to physically encode an object, one must simultaneously consider the matching decoding mechanism that will be used. Furthermore, it is important to also consider the earlier mentioned Viral Spaces value of leveraging technology ecosystems such that a system is designed to scale effectively and with value. In this sense, it is powerful to leverage the existing mobile device ecosystem as our decoding backbone. While this created a potentially arbitrary limit, it aids in creating an argument for the viability of this idea. Thus, for this thesis, I consider leveraging the sensors and devices found commonly in mobile phones: speakers, microphones, cameras, accelerometers, and magnetometers. However, for the majority of this work I use optical means for my method of detection due to the robustness and ubiquity of modern optical sensors, the large amount of existing image processing reference, and the simplicity of the user experience it provides.

### 1.4.2 Optical

One of the most heavily used mobile sensors today is the camera installed in many new phones. Given the ubiquity of camera phones and smart phones, many applications have been made to leverage this new widespread optical sensor. Many augmented reality (AR) applications have been envisioned and produce and there exist many mobile-based image processing libraries for real-time processing. Furthermore, beyond the mobile domain, there exist many image processing techniques and software packages that could be applied to this project.

Using an optical solution (i.e. the camera) has the advantage of being a very common sensor with much existing prior and open work on image processing, yet does still suffer from a few drawbacks. One of the largest challenges with image processing techniques and optical means of sensing is that the results are often influenced by the given lighting situation. Separate considerations must be made to perform the

same task in outdoor day versus outdoor night versus indoors. Furthermore, the optical properties of the material that is being sensed can play a large and varying role. Detecting opaque surfaces of certain colors is a different process than detecting surfaces of translucent or varying color materials.

One technique that can be used to make optical sensing more powerful and robust, given the mobile form factor that we are constrained to, is exploring the use of optical attachments. Additional lensing, lighting, or shielding can be used to create a more defined optical environment to sense. This can result in much higher resolution sensing and high quality results. Prior work that has been done in the domain of optical sensing at micro-scales will be further discussed in Section 3.5.

### 1.4.3 Tactile

An alternative method of sensing would be to use the accelerometer found in many mobile devices to map the surface features of a given object. Such a technique has the advantage of ubiquity, as accelerometers are increasingly common among newly made mobile devices. However, using the accelerometer proposes a scenario that, while technically arbitrary, is constrained by the manufacturing constrains imposed by mobile developers. To accurately map a surface using the accelerometer, a high sampling rate (kHz at least) is desired. However, because this is not the use case that is often envisioned by the phone manufacturers, the sample rate of the accelerometer is often limited to around 100Hz. This is a constraint that is imposed in the hardware design of the mobile handsets. An alternative solution would be to manually attach an additional accelerometer chip to the case of the phone, with the argument that in the future, it would be trivial to change the software implementation that currently limits accelerometer read rates.

However, the accelerometer also suffers from the issue that to map a surface, the phone must be dragged or swiped across the surface. This can present issues and challenges associated with rotation and orientation of the device. Dragging in one direction may not provide the same result as dragging over the same area in a different direction. This may prove to be a challenging user experience issue. Prior work that

has been done to map surfaces using an accelerometer will be presented in Section 3.5.

### 1.4.4 Acoustic

Acoustic, or rather ultrasonic, methods of identifying surfaces are also an option. The idea is to design the surface material such that when probed with ultrasound, it demonstrates a unique response. Here, the probe may be an array of ultrasonic emitters that can be sequentially transmitting or receiving to effectively image the surface properties. Because we have full control of the system (we design the material and the probe), we can model the ultrasonic transmission for a given surface or texture and match this to what our probe actually reads. Figure 1-2 demonstrates this idea.

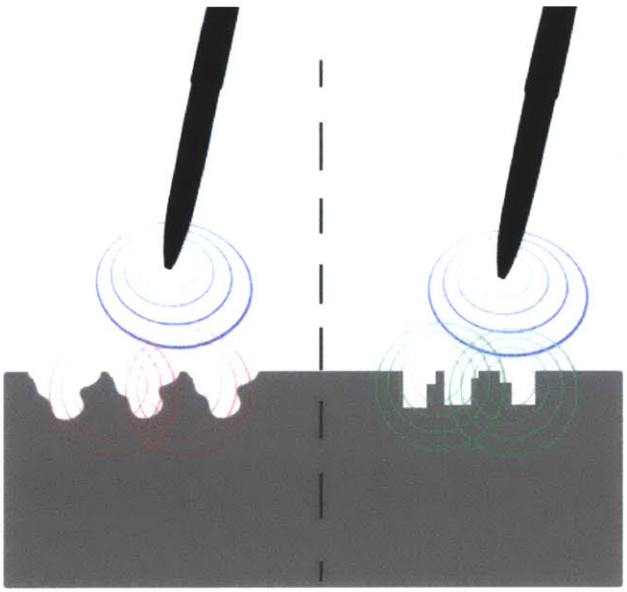

Figure 1-2: Probing an ultrasonic response as determined by surface texture.

By creating varying surfaces with different cavities, features, resonances, or compositions, we can tailor the response. The optimal solution would be to use the microphone and speakers that are found in a mobile device to produce and receive these ultrasonic pings. Unfortunately, the chips used in current mobile phones do not perform at the acoustic frequencies that are needed, so this would have to be more

of an exercise in exploring future possibilities.

### 1.4.5 Electromagnetic

Leveraging the magnetometer or NFC chips often found in mobile devices could also provide a means to sensing encoding objects. In the case of the magnetometer, the idea would be to embed deterministic magnetic particles across the surface of a material. By then swiping the phone across the material and detecting the pattern of magnetic material, it would be possible to decode some amount of information. This suffers from the challenge of actually encoding the material, as methods for creating deterministically magnetic materials are not well explored.

Alternatively, there is the possibility of using the NFC chips that are being put in some new phone models for reading and interacting with RFID tags. RFID tags have undergone significant research and come in many shapes, sizes, and material containers. There has been some work ([42] [44]) on printing RFID tags which may provide a fruitful route of exploration. However, because these processes used to create the RFID tags do not directly interface with other digital fabrication methods, it would likely be a cumbersome process to integrate such encoding into the surface of a material (and certainly not possible to have that represented in modern CAD systems).

## 1.5 Thesis Map

Chapter 2 of this thesis will provide additional in-depth background regarding the history and state-of-the-art of many of the fundamental technologies and processes that are discussed and explored through this work. Chapter 3 goes on to describe the more recent related work that has been done with a similar vision as the one I outline here.

Chapter 4 begins to dive into work performed for this thesis and documents the relatively simpler challenge of encoding and decoding planar surfaces. Chapter 5 presents an evaluation of the technical features of this work and takes a look at live

user interaction studies.

Chapter 6 moves onto non-planar surfaces and explores the fabrication (with a heavy emphasis on 3D printing), encoding schemes, and decoding algorithms that are involved. Chapter 7 leverages the previous chapter and describes the technical implementation of a mobile and web interface for interacting with these 3-dimensional surface encodings. Chapter 8 goes on to explore some simple applications of the work described in Chapters 6 and 7, beyond the domain of 3D printing.

Chapter 9 begins to close out this thesis document by taking a step back and looking at the future work that could be performed and discussing some of the implications of the ideas presented in this thesis. Chapter 10 provides thoughts and a conclusion.

# Chapter 2

# Background

## 2.1 Computer Aided Design

Computer Aided Design (CAD) tools are programs that allow designers, engineers, architects, and technicians to construct and represent physical 3-dimensional drawings of arbitrary objects. CAD tools are often used when prototyping new products or devices to define the physical dimensions and properties of the given object. CAD tools can often be favorable over hand-drawn documentation because they provide exact dimensional precision and offer relational dimensions. If one is designing a ball-and-socket joint, for example, the socket can be defined in respect to the ball, such that if the ball size changes, the socket also changes accordingly.

One of the first instances of a CAD tool was created by Ivan Sutherland. His tool, Sketchpad [37], allowed users to interact with a computer screen with a specialized pen to draw lines and 2D geometric shapes. These shapes could be connected to one another through geometric and constraint-based relationships.

Since Sutherland's initial work, CAD tools have made enormous advancements, in part due to the growth of computational power. Modern CAD tools now offer full design suites for whatever domain a user may be working in. Several CAD tools provide engineers with full physics simulators to provide them with quantitative measurements of how their design will perform under certain conditions. Using CAD programs has become a staple in workflow of designing nearly every modern product.

Figure 2-1: Ivan Sutherland in front of his Sketchpad demonstration.

Buildings, cars, airplanes, pens, mobile devices, and many other products first exist as a 3D CAD model.

These models are typically represented as a mesh, though this design decision is often debated. A mesh defines the object by defining its outward facing surface and any features it may have – it defines the shell of the object. That is to say, most CAD models are 'hollow', even if the final product they represent is not. Such a representation allows for relatively simple and quick graphical rendering (as only the parts that could possibly be seen are rendered), but produces some challenges under certain conditions. With the advent of 3D printers, for example, using a mesh to define the volume that the 3D printer will fabricate sometimes leads to ambiguities and challenges. For this reason, there is much completed and ongoing work exploring the use of volumetric representations of CAD models [32] [27]. Such representations were not practical during the early development days of CAD tools, as the computational power that was available was simply insufficient.

The current state of CAD tools is roughly divided between commercial (paid) tools and open-source products. The commercial tools, while generally regarded as providing better interfaces, features, and efficiency, often cost several thousands

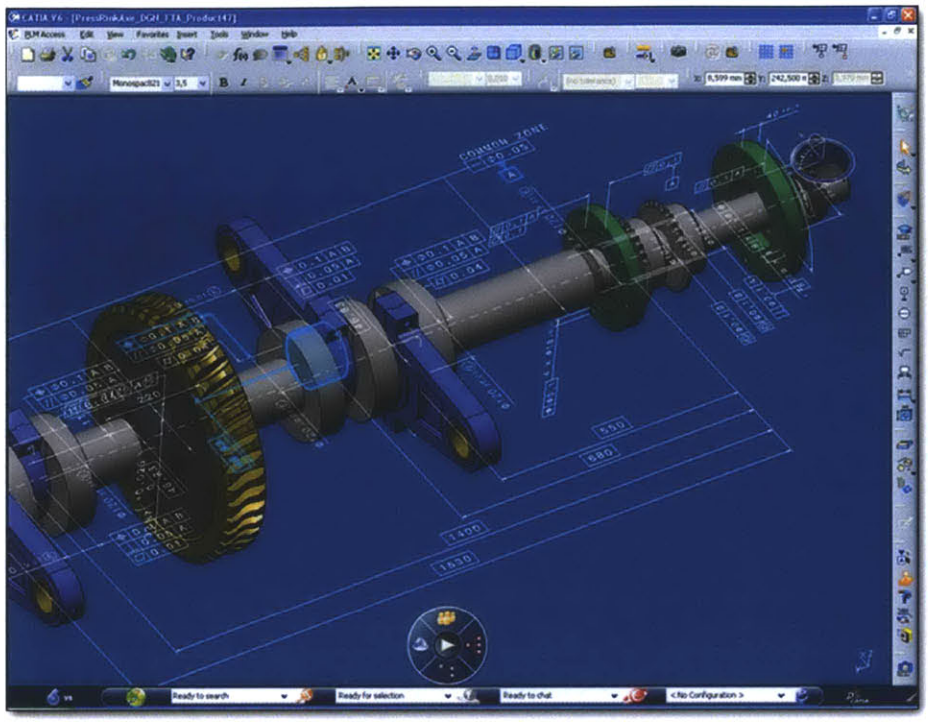

Figure 2-2: CAD Model as represented in the Catia software package.

of dollars. Open-source tools on the other hand, while seen as marginally weaker in comparison, are free and are helping to drive the current maker movement, a movement that seeks to involve all people in the process of design and fabrication.

## 2.2 Digital Fabrication

The real power of a rich CAD ecosystem is its ability to integrate with Computer Aided Manufacturing (CAM) tools. A full CAM toolset is able to take the output of a CAD program and convert it into the necessary tool paths, motor speeds, etc, to drive a CNC (computer numerical control) machine to produce the model as described by the CAM software. A CNC machine can be anything from numerically driven mills, lathes, high-powered lasers, plastic deposition head, plasma beams, drills, or any other tool that can add or subtract matter from a given model. In this way, a CAM toolset provides the direct means to take a given CAD model and fabricate it into a real physical prototype [43].

For the majority of time since the advent of modern CNC machines (around the middle of the 1900s), they have been used in large-scale industrial settings. Due to the cost, noise, and technical expertise needed to drive these machines, they were typically relegated to commercial manufacturing means. They also found a nice home in industry because of their ability to reliably reproduce identical objects. A single set of instructions could exactly define the process for producing thousands of identical (to a tolerance) parts. This drastically lowered the cost of many consumer products and led to a growth in the automation of manufacturing.

### 2.2.1 Personal Fabrication

Recently, thanks to lowered production costs, heightened motor efficiency, and widespread access to software tools and packages, there has been a growing trend in the domain of personal fabrication. Most notably, laser-cutting and 3D printing tools have made it extremely easy for a user of no prior experience to quickly build prototypes they have designed in any number of CAD tools [3] [9] [26].

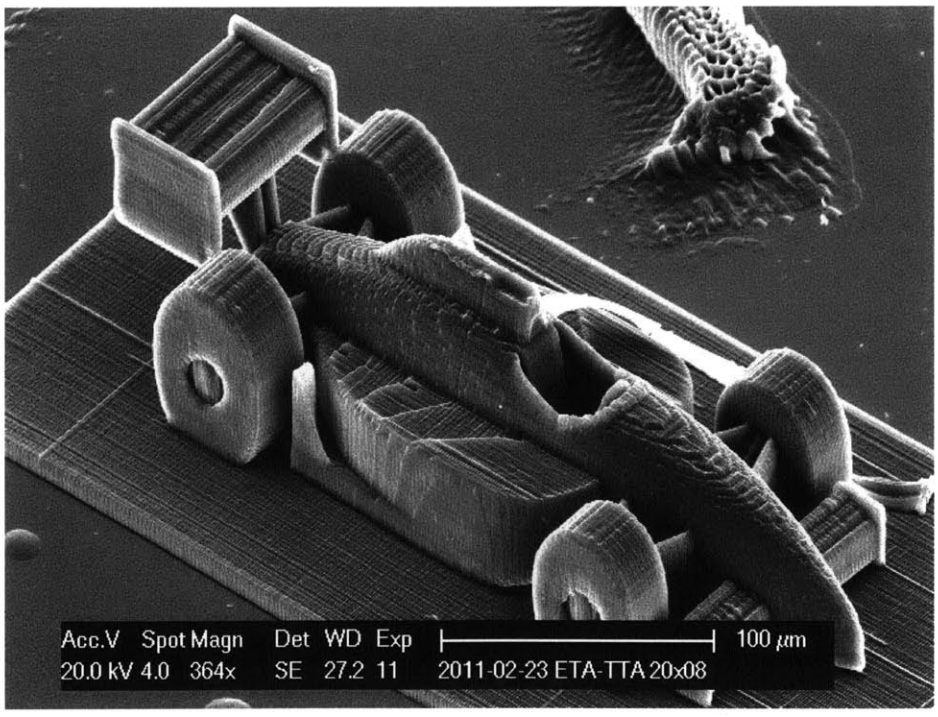


Figure 2-3: Nanoscale 3D printed F1 racecar model.

Growth in 3D printing technology has especially fueled this trend, as novice users are able to precisely fabricate 3-dimensional objects. There exist a large swath of different printing technologies (most of which are additive, meaning the incrementally add material to form the final object), such as fuse deposition modeling, selective laser sintering, stereolithography, or laminated object manufacturing, to name a few. These processes vary in their minimum resolutions (some in the nano-scale – see Figure 2-3 [4]), materials, and speeds.

However, regardless of the precise mechanisms, these efforts demonstrate the large and growing field that 3D printing represents. Many of the efforts in this domain are also targeted towards the simplification, minimization, and cost-reduction of these processes, with the end goal of providing affordable at-home 3D printers.

Such a proposition, everyone having access to 3D printers at home, raises interesting social and community based issues. It is foreseeable that many manufacturing industries will go through a similar challenge that software companies experienced during the growth of personal computers regarding the reproduction of patented or copyrighted products. When the quick and inexpensive fabrication of arbitrary products and objects is feasible in everyone's home, the manufacturing industries that currently perform these tasks will likely have a major restructuring.

As a step in between industrial manufacturing and at-home production, several companies have begun to emerge that offer to fabricate the 3D CAD models that are designed by individual users. Companies such as Shapeways [34] and Sculpteo [33] offer to 3D print CAD models that are uploaded through their website. A user simply uploads their file, chooses from a selection of available materials and colors, and waits for their uniquely designed product to arrive. Around these sites, cultures of remixing and sharing models and designs have spurted; a phenomenon that is similar to what was being seen with the advent of the open-software community.

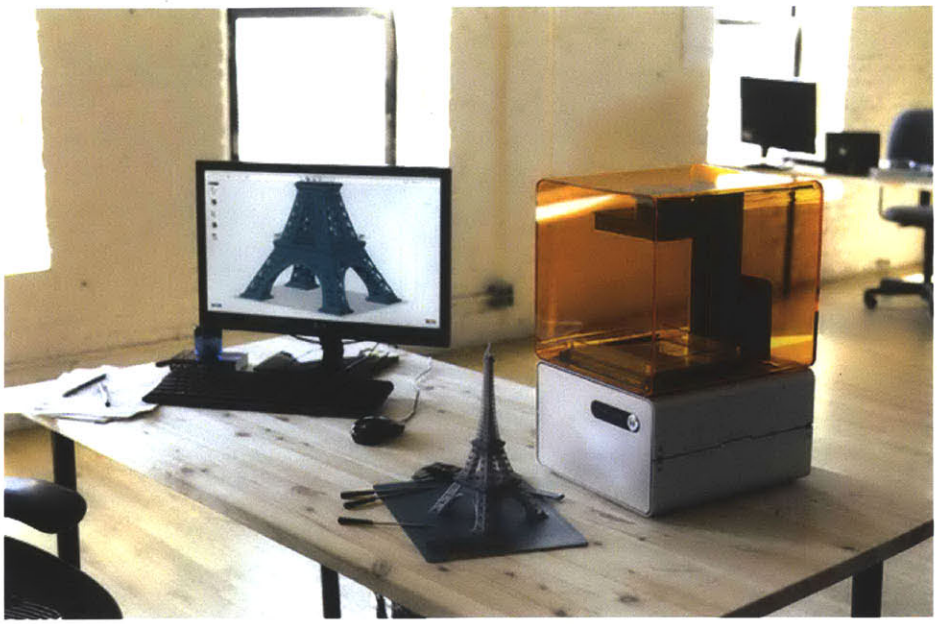

Figure 2-4: A Form1 3D desktop printer and sample output. ©Formlabs

## 2.3 Rapid Hardware Production

Software development has advanced to its current state, in part, thanks to the open-source software movement. Open software is the idea that code should be accessible and free to all, granting each user the right to duplicate, edit, and republish any bit of code. This movement has led to not only a rich and diverse code ecosystem, but also to a growing developer population. By encouraging anybody and everybody to access open software, non-professionals are becoming engaged and contributing members of the community. Similar strides are being made for hardware. The advent of these rapid prototyping machines, advanced software design tools, and access to resources like Shapeways and others, is creating a situation very similar to the early days of the open software movement. Hardware design is very quickly becoming more accessible to a broader audience. We are near (or even at) the threshold where sending a digital CAD file to a person who then fabricates that CAD model, is faster and easier than simply sending the original object itself.

Furthermore, efficiencies in supply chains and increased factory automation have led to significant growth in the in industrial manufacturing side of things. Develop-

ment tasks that used to be very expensive, requiring an order of tens of thousands, can now be done on an individual prototype scale for a cost-effective price. This not only allows large companies to more efficiently and quickly prototype new ideas and products, but it also, importantly, lowers the barrier of entry and levels the playing field for many startup companies and smaller corporations.

## 2.4 Physical Tagging

The increase in manufacturing efficiencies, capabilities, and accessibility has led to a growth in the demand for physical tags. As the product ecosystem becomes more diverse, and created by a larger set of participants, it is helpful and necessary at times to have some mechanism for identifying or interacting with these physical objects. In the case of this thesis, tagging is explored as I seek to create materials that are self-descriptive. Materials that contain within them some digitally identifiable code. Such materials can then be used for delivering targeting learning opportunities, storing documentation, or any other application where a bridge between physical objects and their digital representation is needed. Thus, identifying a strategy to embed these digital codes into a material is of key importance. There exist many tagging technologies that have been designed for varying applications and given the broad amount of previous research, it is useful to have a taxonomy that outlines some of this existing work.

For my system, I am seeking a tagging solution that demonstrates several key characteristics:

1. Is passive (i.e. does not require active powering)

2. Can be made ubiquitous across the material (i.e. the material can be identified even if it breaks, or is probed from an arbitrary direction)

3. Can be simply decoded. That is, an ideal solution does not require processing at a lab or any expensive, cumbersome equipment. In the best case, a solution can

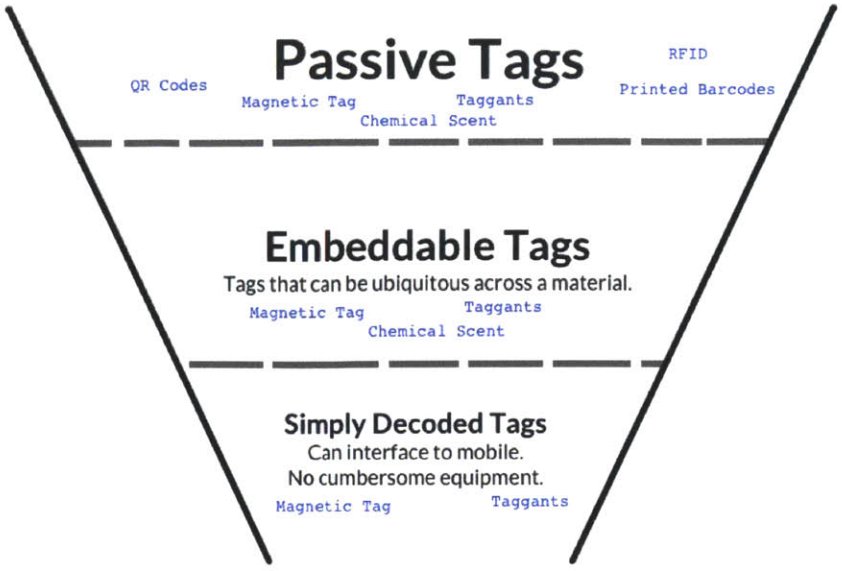


Figure 2-5: The successive narrowing of each requirement. The examples in blue are for simple demonstration.

be integrated to work with existing mobile devices with little (to no) additional hardware.

These tend to limit the number of approaches that are desirable, and each is successively more restrictive. The funnel diagram in Figure 2-5 depicts the narrowing scope of each characteristic.

The following taxonomy explores passive tagging technologies and provides a brief description along with references for more detailed specifications.

### 2.4.1 Taxonomy

| Technology | Description | Strengths | Weaknesses | Ref |
|---|---|---|---|---|
| RF Harmonic Tags | RFID tags that can be sensing at a variety of ranges using a variety of frequencies. | Passive. Cheap. Large address space. | Localized. Dependent on geometry (antenna cross-section must be flat). | [1] [38] [7] |

| | | | | |
|---|---|---|---|---|
| 1D Barcodes | Printed stripes of varying contrast that can be optically read. | Well known. Cheap to produce. Scanning technology easily available. | Dependent on geometry of material. Direction of scan dependent. Not suited for harsh or dirty environments | [12] [38] |
| Organic Compounds | UV absorbing, visible emitting inks and dyes with known spectral patterns | High absorption cross section. Invisible under normal conditions. | Broad absorption bands. | [1] |
| Organic Compounds | UV absorbing, visible emitting inks and dyes with known spectral patterns | High absorption cross section. Invisible under normal conditions. | Broad absorption bands. | [1] |
| Rare earth oxides | Doping materials with rare earth oxides to give it specific narrow-band optical properties | Narrow absorption and emission bands. | Cost. Dispersing rare earth oxides into material is non-trivial. | [1] |
| Chemical Scent | Using fluorocarbons of nitro compounds to create distinct negative ion mass spectra | Sensing at a distance | Move useful for detection, than precise identification. Expensive detecting technology. Small address space. | [1] |
| 2D Printed codes (including QR) | Printed arrays of varying contrast that can be optically read | Well known and cheap to produce. Scanning technology widely available. | Dependent on geometry of material. Direction of scan dependent. Using common scanners, the tag must be large. | [12] [31] [25] |

| | | | | |
|---|---|---|---|---|
| Taggants | Small pieces of plastic with unique microscopic color bands. Often used in explosive material to identify bombs after detonation. | Very robust. Can be easily integrated into materials. | Requires microscopically capable optics. Not easily commercially available due to military use. | [39] [41] |
| Magnetic Tags | A substrate with a number of varying magnetic sections. Magnetic sections can then be sensed in a variety of ways to deliver tag. | Invisible. | Difficult to manufacture. | [8] [6] |
| GPS | Using GPS location of a fixed item with reference to a database to identify the object. | Widely available technology. Cheap to implement. | Address space is spatially limited to resolution of GPS. Not passive. | [2][40] |
| Holographic Tag | Embedding small holographic tags into various materials for identification. | Relatively cheap to produce. Provide data over angular dimension. | Many techniques outlined are only for authenticity identification - not for use with computer readable IDs. | [16] [14] |
| Acoustic Barcode | Microphone detects that noise produced from scraping over ridges cut into material | Simple to manufacture. Simple interaction. | Direction dependent. Microphone must be attached to surface of interest. Only shown for use with planar surfaces. | [13] |

| | | | | |
|---|---|---|---|---|
| Radioactive Tags | Radioactive isotopes are put into an object for detection using standard Geiger instruments | Can be sensed at large distances. Leaves trail of movement. | Health and safety concerns. Little way of creating large address space. | [10] |
| Acoustic Tag | Specific acoustic frequencies are generated and detected for the sake of unique identification. | Sense at a distance. | Not passive. Difficult to incorporate into material. | [24] |
| Bio/DNA ID | Covering a material in some biological substance that contains specific properties that can be tested. Often used in biology to identify specific proteins, etc. | Invisible. Large address space. | Health and safety concerns. Cost of decoding. Lab required. | [23] [28] |

Table 2.1: Taxonomy of tagging technologies.

# Chapter 3

# Related Work

This thesis contributes to a host of projects that are designed around the goal of merging the digital and physical worlds. Many of the design principles that are featured in prior works are also seen in this work, and as such, it is important to review and understand the motives and accomplishments of these past projects.

When working in this domain the question often comes down to where you place the intelligence that is facilitating the digital/physical interaction. One option, which has been gaining traction as camera systems become more powerful, is to put the intelligence in the environment - where the environment is often a camera and lots of computer vision processing. In this situation, the object itself has no intelligence and a camera is using any marker it can find to orient and understand what it's looking at. This often comes with the disadvantages that the system needs a lot of training, is dependent on the lighting in a given space, and is frequently not precise enough to enable specific touch interactions. To avoid some of these problems, you could aid the environmental systems in recognizing objects by placing tags on the object, thus creating a more distinct environment and easier processing task. The final iteration in this progression would be to put the intelligence into the object itself through an extra post-processing step. A host of projects have explored the relative strengths and weaknesses of these approaches, as I will discuss in the following sections.

My unique contribution to this type of work it to introduce an alternative means of merging the physical and digital interaction world by altering the manufacturing

process itself to create objects which are inherently machine readable, yet do not significantly differ from their original intended form.

## 3.1 Tag-Facilitated Interaction

In many situations, it is simplest and easiest to apply a tag to an object that one wishes to interact with. Given the very low cost of laserjet and inkjet printing, many of these tags come in the form of paper-based barcodes or QR codes [31]. The benefit of using barcodes and QR codes in the current technology ecosystem, is that their readers are widely, and freely, available. This makes them an obvious choice in many advertising and industrial applications. The most familiar form of such tagging is found on all of our consumer products, whose packaging often contains a UPC barcode for identification.

More elaborate interactions have also been created. The Electronic Pricetags project by Matt Blackshaw, Rick Borovoy, and Andy Lippman from the MIT Media Lab is a demonstration of using QR codes to facilitate interaction with large datasets. The project consists of a large shelf of cereal boxes, each with a dedicated screen displaying their price. Upon tapping the screen, a QR code is presented, which, when scanned, provides the user with a list of options such as price, calorie content, sugar content, gluten content, etc. Upon selecting an option, all screens update to display the detailed information about their dedicated cereal box. In this way, data about an entire array of products can be quickly and easily consumed.

A similar project, produced in Korea by the Tesco Home Plus supermarket chain placed large banners across subway walls. The banners depicted a typical supermarket shelf found in a Tesco Home Plus store. Each item on the shelf had an associated QR code that could be scanned to allow the user to purchase that item. Upon selecting all of their items, the subway commuter could place an order to have those groceries delivered to their home.

Moving beyond the simplicity of a printed QR code, the Bokode project from the MIT Media Lab offers an active printed code with distinct benefits [25] . The

Figure 3-1: An installation by Tesco Home Plus for QR-code grocery shopping.

project relies on the bokeh effect to allow cameras to focus on an extremely small, and otherwise unresolvable, printed code. To achieve this effect, the codes must be actively backlit, but in exchange allow for the tag to be scanned at much further distances.

## 3.2 Physical Encoding

As mentioned above, a second option for creating interactive physical objects is to embed some sort of intelligence into the object itself.

The Acoustic Barcodes [13] project is a recent notable example that uses the physical structure of a material to encode information (see Figure 3-2). The project relies on objects that have parallel grooves etched into the body of an object and a microphone that rests on the same surface as that object. When a rigid object (pen, fingernail, phone, etc) is dragged over the etched lines, a time sequenced series of clicks is generated. These clicks are captured by the microphone and decoded. This technique is dependent on the correct drag direction and the drag time.

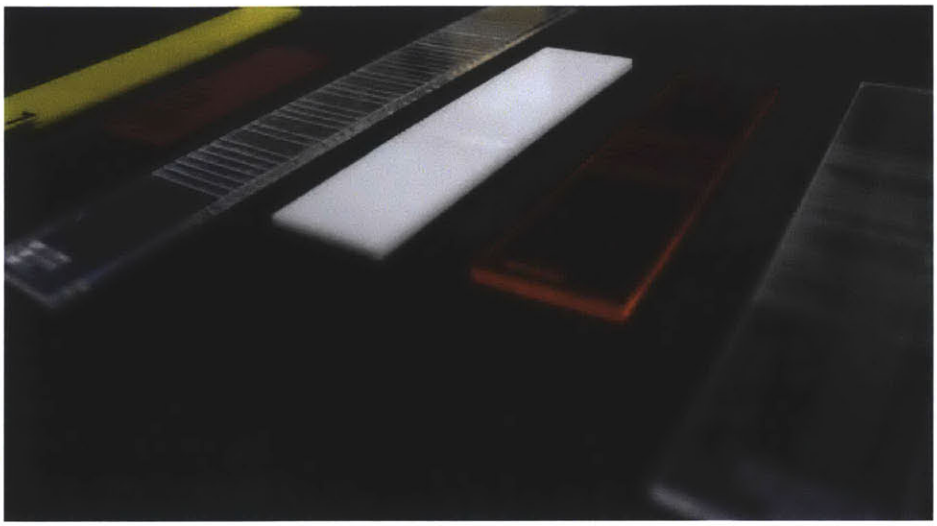

Figure 3-2: Acrylic strips that have been etched to create acoustic barcodes.

Commercial options have also been hinted at, specifically by the company Visualant [17], who proposes the idea of using their LED based spectroscopy tools for decoding implanted optical codes. Visualant has recently patented the technique of using sequential illumination of varying light frequencies to identify the unique optical properties of biological samples, materials, and certain dyes. While the technique requires complex illumination, it is certainly a solution that could fit into the larger idea of identifying self–descriptive materials and objects.

In a similar realm, digital paper is a technology that uses very fine ink dots on a regular piece of paper to encode many locations across the surface [11]. The small dots can be resolved with a special pen that has a built-in camera. This pen is frequently used to track a user's writing across the surface of the paper so that it can be digitally stored and viewed. Anoto is a company that has produced a popular version of digital paper that they sell along with proprietary pens for these uses. One group has attempted to map the Anoto pattern to 3D objects through applying the pattern on top of produced object and through the use of a multi-color 3D printing process. The group found difficulty in replicating the pattern accurately using the transfer technique and had no success in reading the pattern that had been directly 3D printed [36]. This thesis is interested in creating a experience similar to that of

Anoto paper for arbitrary 3D objects through the use of physical features.

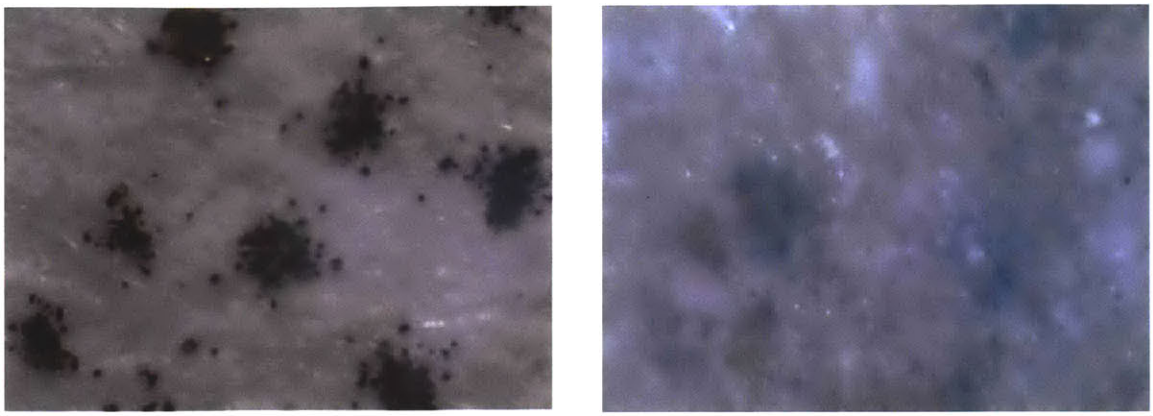

Figure 3-3: Comparison from [36] showing the Anoto pattern printed on paper (left) and printed on an object with a multi-color 3D printer (right).

## 3.3 Just-in-Time Information Systems

Another important aspect of this thesis, as first introduced in Section 1.3, is the idea of presenting the right information at the right time [30]. Such systems that achieve this are often called just-in-time information systems.

Figure 3-4: Bradley Rhodes wearing his remembrance agent.

One strong example of prior work in this domain comes from the MIT Media Lab, where Rhodes introduced the Remembrance Agent: a wearable system for augmented memory [29]. This system was worn on the head and provided a display in front of the eye that provided contextual information that the system deemed useful to the user.

Another example, also from the Media Lab, is the Invisible Media Project [22]. This project, in addition to proposing a tagging technology, explores an interaction scheme that is enabled through multiply-tagged objects. In this work, active infrared transponders are placed over the surface of an object and an IR sensor attached to an earpiece is used to determine where the user's gaze is focused. The system delivers audio information corresponding to where the user is looking at any given time to the worn earpiece (see Figure 3-5).

Figure 3-5: David Merrill using the Invisible Media system to learn about a car engine.

## 3.4 Collaborative Annotation

There also exists work in designing and building around collaborative annotation spaces. There are many web-based tools for collaboratively creating documents, presentations, and other files. Most notably, Google Docs is a full web-based system for

creating and collaborating, in real-time, such documents. An earlier effort, and one that pertains to CAD models, was performed at the University of Washington, where work was done to explore web-based tools for interacting and annotating models in a 3D virtual environment [20].

Second Surface provides a more recent entry into this field. Using augmented reality computer-vision libraries, tablet devices are used to allow collaborative drawing in a 3D space (see Figure 3-6) [21].

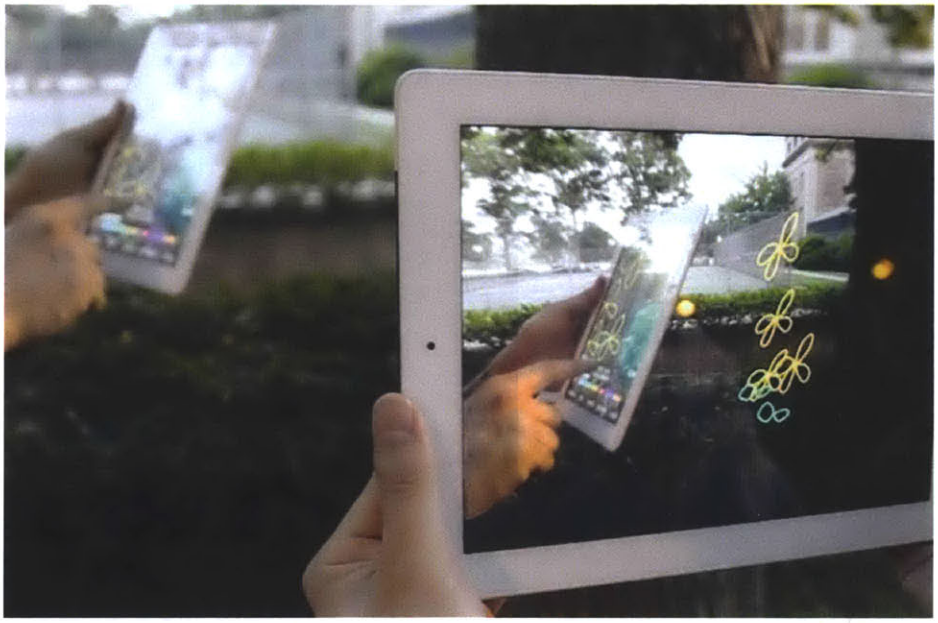

Figure 3-6: Two users interacting with the Second Surface project.

The key enabled in many of these projects is the growth and ubiquity of fast, always-on network connections. By using the internet as a backbone, near real-time experiences can be created that connect two parties through their devices.

## 3.5 Methods for Decoding Texture

More prior work that is critical to explore in relation to this thesis is various technical work on sampling and 'reading' textures. While these prior works are often performed with different motives than the ones presented in this thesis, they provide critical technical routes that will allow us to decode surface textures into binary data.

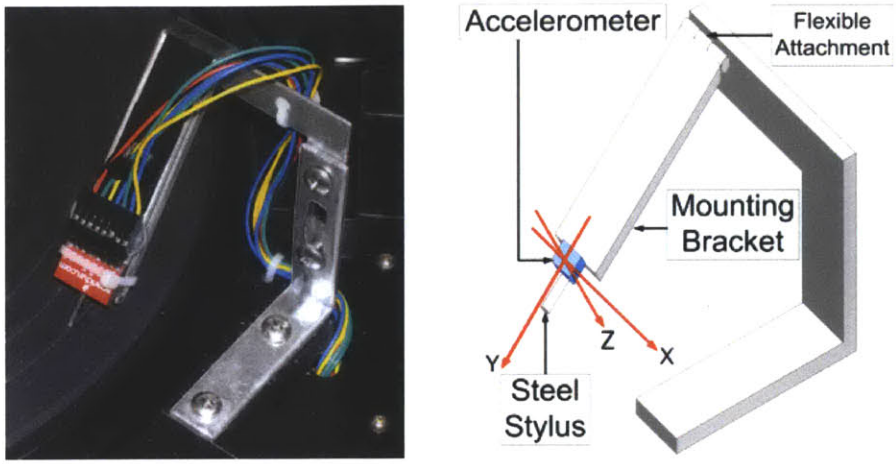


Figure 3-7: An example setup for using an accelerometer to detect surface features.

Several projects have attempted to leverage vibro-tactile surface responses to categorize varying materials [5] [35]. This work typically uses an accelerometer with a high sampling rate attached to a rigid tip (see Figure 3-7). The tip is then dragged over varying surfaces and the resulting response is used as a sort of fingerprint for identifying the material. The work has demonstrated the ability to differentiate between varying materials, but in both cases, suffers from the requirement that the same drag pattern is always performed.

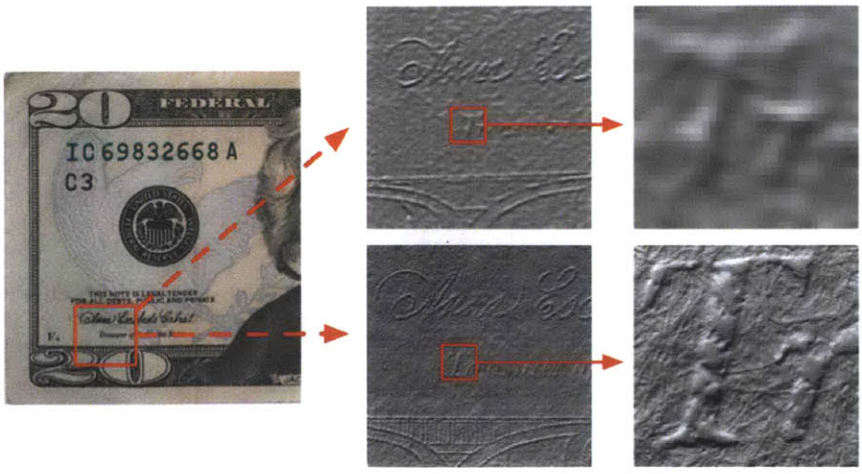


Figure 3-8: Sample results from [18] showing the high resolution that is achievable.

Another well researched method for identifying small surface features is the use of an optical photometric stereo technique [15] [19] . This procedure has shown impressive abilities to capture and interpret very fine surface features (e.g. resolving the depth of the ink on a dollar bill – see Figure 3-8 [18]). These methods use an array of illumination and a simple camera. By illuminating the texture of interest with lights at a different position and color (or time), the shape and depth of the object can be determined from its varying shadows. Using this to sense the surface profile of interest, computer vision algorithms can then be used to determine, in the case of this thesis, an encoded bit sequence.

# Chapter 4

# Encoding Planar Surfaces

The main process of encoding a 3-dimensional (3D) non-planar surface is similar to that of encoding a 2-dimensional (2D) planar surface. More specifically, that process entails identifying a primitive that will be used to denote a bit, planning a scheme to arrange many primitives, and creating algorithms to identify the presence and spatial position of each primitive. By first exploring 2D planar surfaces, an understanding of the available techniques, primitives, and encoding schemes can be attained. This will be an easier first step towards designing 3D encoded textures than making a direct leap. Furthermore, there exist a host of very powerful 2D rapid prototyping and CNC tools (such as laser cutters, vinyl cutters, etc) that offer a unique opportunity to integrate the manufacturing and encoding of an object. Due to the increasing resolution at which these machines operate, it is now possible to not only form our 2D object in a single process, but to simultaneously encode it with deterministic surface features.

## 4.1 Encoding Shape and Pattern

When choosing an encoding scheme, it is important to identify both a primitive and an encoding pattern. Furthermore, issues of scale, rotational invariance, and surface geometries must be explored. To know how to design a robust encoding scheme, it is important to first understand the characteristics of algorithms that will be using

in decoding, what they are well suited for identifying, and to design the encoding scheme in reference to this.

### 4.1.1 Existing Patterns

A good first step is to identify existing patterns and determine whether they could be mapped to a surface encoding scheme. One obvious and direct mapping would be to use braille, given the shared use of physical features for encoding. However, the data density of braille is designed to make it human-readable, rather than achieve the highest possible bits/area. The most common digital 2D tag is a traditional UPC barcode. While this has the benefit of already having many readers in the world, it is designing for print and may not function well when mapped to non-planar surfaces. Mapping the Anoto pattern is also an option - and something that has been tried by the ModelCraft project [36]. However, the group building this project ran into trouble , which I believe stems from the fact that the encoding scheme uses only a single primitive (and blank space). On 3D objects, where the surface can be contoured, a strict reliance on the relative positioning of primitives (and thus the measurement of empty space between them) may not be an ideal choice.

### 4.1.2 Testing Simple Primitives

Depending on the image processing technique that is used, the shape of a data primitive may be relatively easier or harder to decoder. To explore the intricacies of this challenge, I began to test five different primitives:

1. Circles
2. Squares
3. Triangles
4. Horizontal Lines
5. Vertical Lines

The plan was to use multiple primitives to define a patterned code. To decide which primitives are best to use, one must first understand which are most robust — that is, most easily detected by the chosen image processing techniques. Such a decision also requires looking at which primitives are most reliably detected in the presence of noise. To test these properties, arrays of primitives are generated and several different image processing algorithms are used to identify the patterns. Noise is then added to the source array and the processing algorithms are run again, demonstrating the robustness of detection at each noise level.

### 4.1.3 Generating the Primitives

To generate the arrays of primitives, simple Processing code is used. I define functions to build arrays of each of the primitive types and take, as the input variables, the spacing between primitives and the size of the primitives. I have a main function that loads a noise image, generates the primitive array, and then overlays the noise image with varying opacity. Image files are created for each noise image opacity from 1 (0% opacity) to 255 (100% opacity). I run this for each primitive type, yielding a set of images with the arrays behind varying levels of noise. The code can be found in Appendix A, Section A.1.

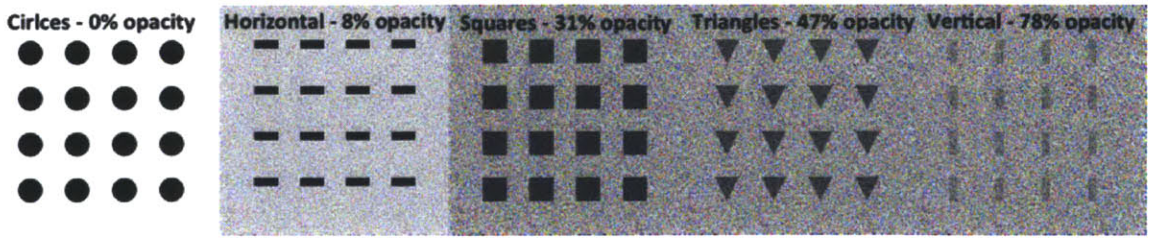


Figure 4-1: Varying patterns and noise levels generated to test a 2D decoding algorithm.

## 4.2 Detecting Simple Primitives

### 4.2.1 Image Processing Environment Setup

There exist many methods to complete image processing tasks. These methods vary between platform, language, and functionality. I spent some time weighing the options and trying to settle on a single workflow that would carry me through the full scope of the thesis (i.e. would be easy to port to mobile devices, embedded devices, etc). I first looked at using Matlab because of the extensive and powerful image processing library it includes. To overcome the challenges of porting these image processing abilities to mobile devices I considered 1) doing the image processing server-side and 2) using the Matlab Java Builder to port any Matlab code to Android devices. In the end, neither of these options are completely desirable as the complication they would introduce would not be worth the slightly simpler image processing work flow. I instead decided to use OpenCV: a common computer vision library. Specifically, the Cinder implementation of OpenCV is used. Cinder is an open-source C++ library that is suited for visual processing and generation. Cinder has the nice characteristics that it can be directly compiled for iOS devices and has OpenCV integration. To use OpenCV in Cinder, one must first add the OpenCV block. Fortunately, OpenCV has an Android implementation as well, so if needed the same algorithms can be used outside of the Cinder environment.

### 4.2.2 Primitive Detection

I began looking at detecting the circle primitives. Searching the OpenCV database of functions for the keyword 'circle' leads to a page describing an implementation of the Hough Circle Transform. After implementing this function, I had initial success in detecting circles, but as I tried to refine the results I ran into a few challenges. The Hough Circle Transform seems to depend heavily on a few parameters that define the behavior of the transform. A couple of tutorials emphasized that the key to the Hough Transform was getting these parameters correct (often through trial and error

routines). This would pose somewhat of an issue for my application given that the material and size of the encoding could depend across the entire application-space of encoded surfaces - meaning that correct parameters may be difficult to set (or slow to find). Furthermore, I found that the Hough Transform would generate some strange false-positives if the parameters were wrong. Empty space would be identified as a circle and some circles would be found while other (identical) circles would be missed. It was not only mischaracterizing circles, it was missing blobs entirely. Given these weak results, I decided to look for another solution.

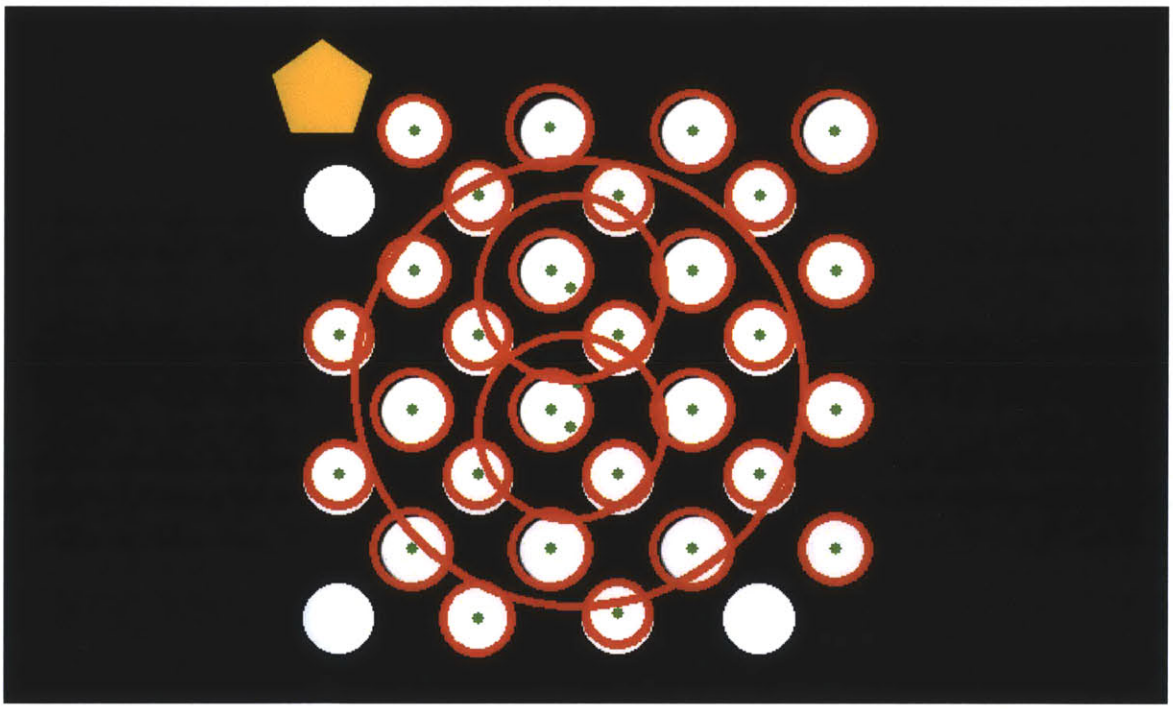

Figure 4-2: Sample Hough decoding showing poor results which led to a decision to search for alternative algorithms.

I decided to focus on a routine that, rather than look simply for circles, first looks for blobs and then tries to iterate and identify each blob. This would provide a foundation that could be used for detecting the other primitives. There exists a CVBlob library, but I had trouble integrating it into the Cinder work flow. I leveraged an AForge shape detection tutorial (`http://www.aforgenet.com/articles/shape_checker/`) to implement my primitive detection. Note, this latter tutorial is not

written for OpenCV, but the concepts and algorithms can be applied. Implementing a triangle, square, and rectangle identifier was relatively simple from this point.

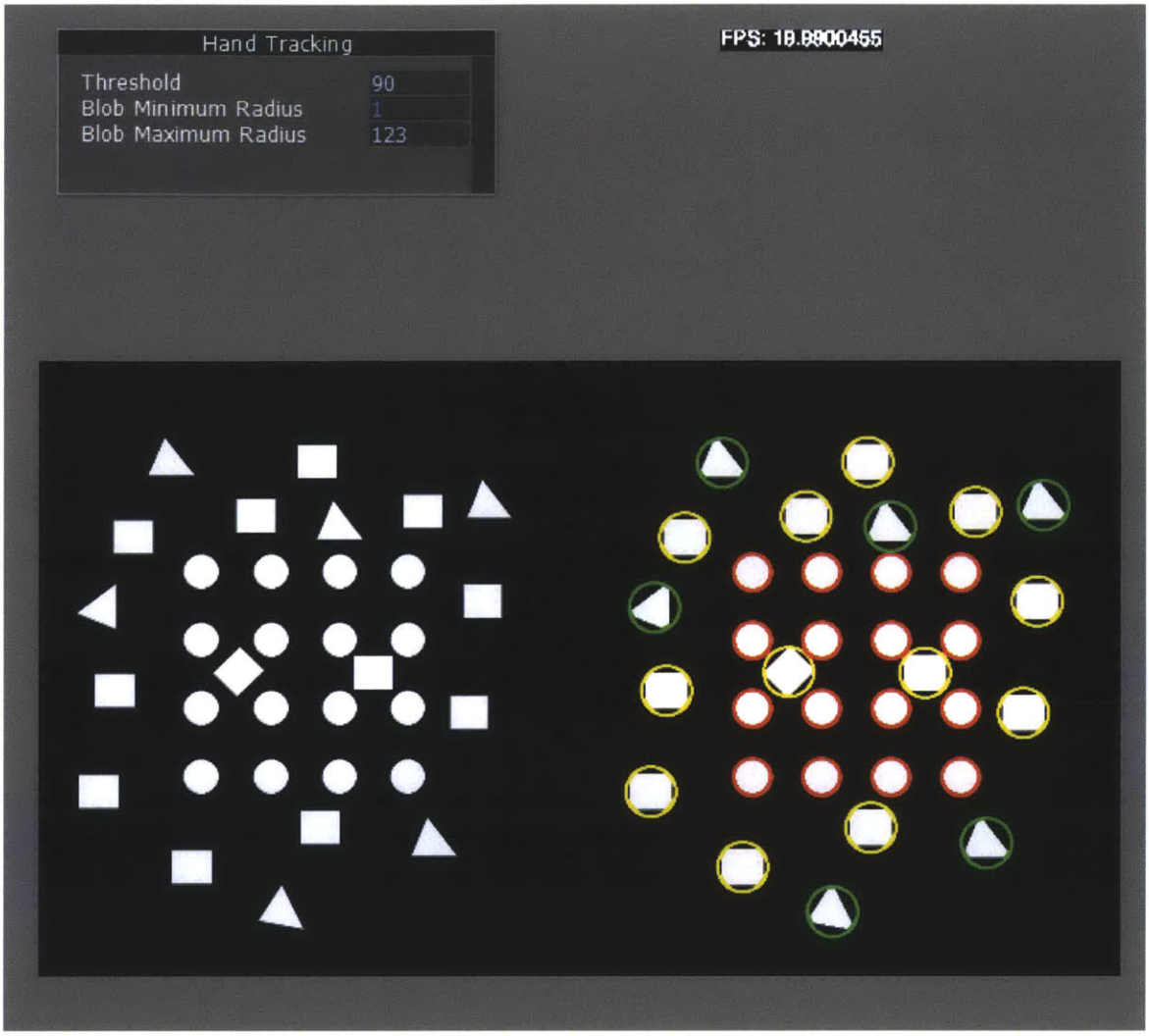


Figure 4-3: Sample decoding showing the ability to differentiate between various shape types.

The code used to properly detect these primitives is given in Appendix A, Section A.2.

### 4.2.3 Reading Codes Through Noise

I then began to run these primitive detection routines on the noisy images I had generated. The idea being that I would be able to identify a noise threshold, after which

the image processing routines would be unable to correctly identify the primitives. I expected this threshold to be different for each primitive, so I ran test cases for each one.

As Figure 4-4 shows, all of the primitives are correctly detected up to a noise opacity of 5% noise. From this point, there are a handful of false-positives mixed in with correct readings. This points to needing dynamically adjusting parameters and identifying routines.

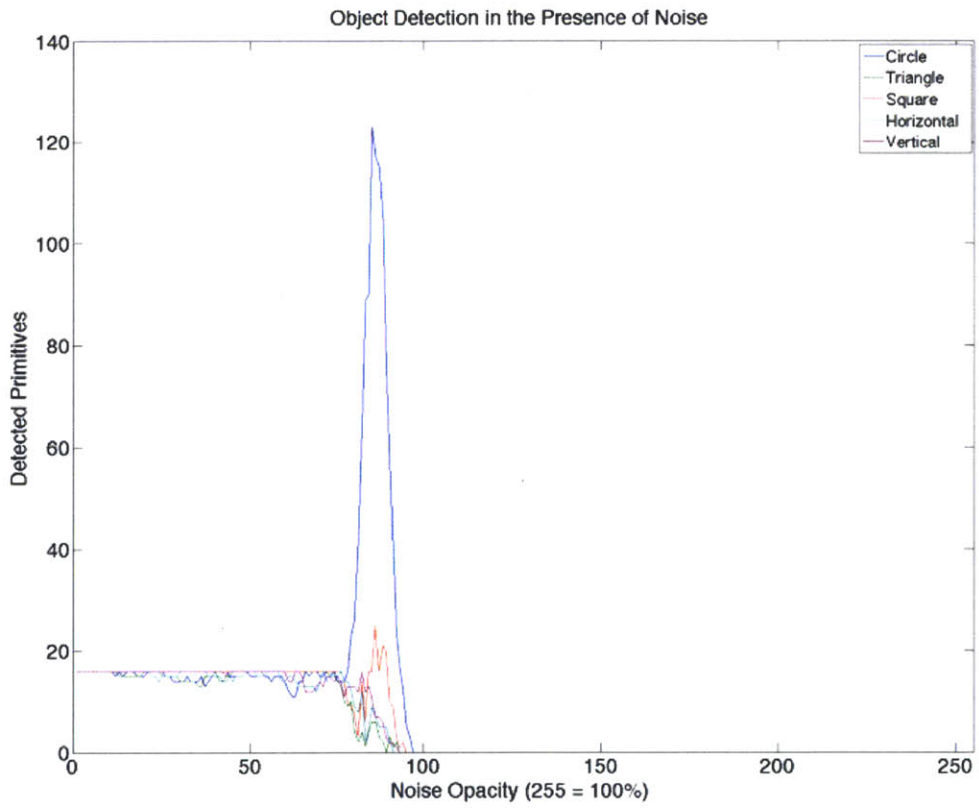


Figure 4-4: Plot showing the number of objects that are detected as noise levels increase. 16 objects are 'intentional'.

It is interesting to note that the circles, in particular, have a large spike as the noise level gets too high, whereas other shapes are simply no longer detected. This may be a good argument for choosing a shape other than a circle, given that there are many false-positive circles under noisy conditions.

While for the time being, it is sufficient to use static photos to test the decoding software, a more robust and real-time implementation must eventually be made to facilitate actual interaction. Such an implementation, and the associated challenges, will be discussed in Chapter 5.

## 4.3 Fabricating the Primitives

After several tests, it became clear that cavities were a great option for fabrication. Not only were they easy to fabricate, but importantly, if made deep enough, they would stay sufficiently dark under many lighting conditions, making the textured surface and optical decoding techniques much more robust.

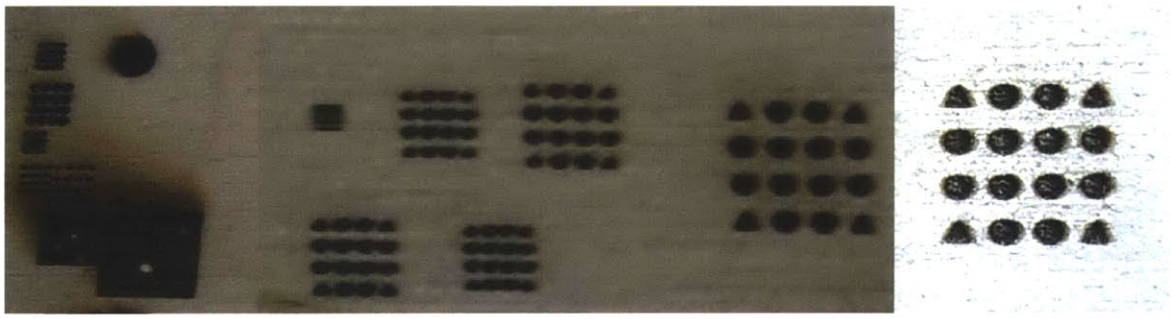

Figure 4-5: Several lasercutting results. The left shows attempts at making positive features (a lengthier and less-desired process), the middle shows arrays of patterns at various sizes, and the right shows a macro shot of a small pattern.

I explored two main method of fabricating 2D planar surfaces. The first was to use the laser cutter. With the laser cutter, a black and white image file is loaded and sent to the laser cutter. By rasterizing this image, the laser cutter takes the dark portions of the file and etches those to a maximum depth and leaves the white portions at a minimum depth. If there had been any grey-levels, they would have been accordingly etched between the maximum and minimum depths. The maxima and minima are determined by the settings one inputs to the laser cutter. There is no exact method for choosing inputs; the exact selection process is more of an art that relies on past experience and guess-and-checking. The laser cutter is able to achieve very fine resolutions. While technically it states it is capable of printing at a

resolution of 1200DPI, I found the real workable DPI to be more around 600DPI; still plenty sufficient for the needs of this work. After many tests, the resulting minimum cavity size that I found achievable is .015"

I also used a milling machine to make a wax mold (of a negative pattern) which I later cast silicone rubber into. The milling machines resolution is limited by the size of the bit that is being used. In this case, the smallest bit available to me was a 1/64" bit. Unfortunately though, the flute of the bit was larger and the bit depth was not long enough, so there is also a maximum depth one can achieve in their cut as set by the length of the bit. The silicone has a nice opaque and non-reflective finish, which makes it quite ideal for optical processing.

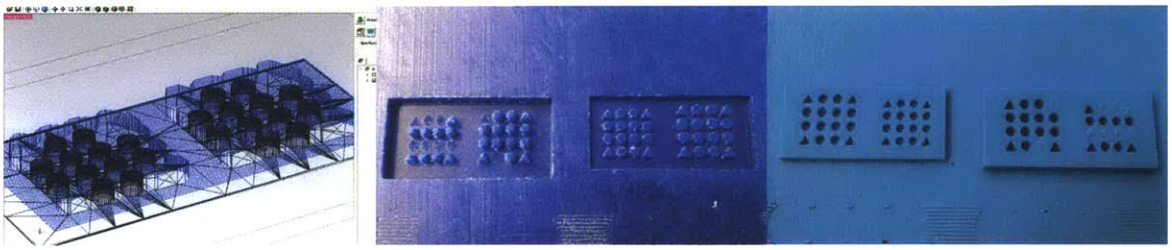

Figure 4-6: An example of the milling process. The left shows the CAD model and mill paths. The middle shows the resulting wax mold. The right shows the final silicone cast.

# Chapter 5

# Planar Surfaces Live Test and Evaluation

To begin transitioning from an encoded surface simulation, to a real functioning system, it is important to make several advancements. The software must first be transitioned from using static images to using a live video stream and, secondly, a wide selection of encoded surface objects must be made with which we can test the system. To build out a test scenario, arrays of circular tokens with encoded patterns were fabricated. These tokens were distributed and used at a local event to grant access. The unique encoded pattern on each token ensures the authenticity of the token. This both pushed the development of exploring fabrication techniques, as well as created a pressure to have a live, working decoder by the end of the week. Furthermore, the event (and the hundreds of scans that ensued), acted as a debug test of the decoder and encoding robustness.

## 5.1 99 Fridays Tokens

The original goal in building these tokens is to work on the decoding algorithms that will be used in this thesis work. As an added bonus however, it also turned out that building these token systems worked to identify patterns and encoding schema worked well and which didn't, as well as provide some useful insight on the behavior of users

with the system.

### 5.1.1 Token Design and Fabrication

The laser cutter was used to quickly develop a handful of test patterns. I found that it was feasible to reliably generate cavities as small as .015". I also explored with creating negative lasercut stencils for the purposes of casting silicone in the future. For these purposes, the resolution was also impressive. I created a couple sample tokens , with the design such that the encoding is put in the "99" label on the token (see Figure 5-1). The final design would vary a bit, but the general concept remained.


Figure 5-1: Left: A few sample cut tokens. Right: A close-up of one of the early token prototypes.

This first iteration of the token encoding had issues with merging cavities at times - the thresholding was quite sensitive. To alleviate this, future designed can be made with the spacing between cavities larger. A few orientation marks may also be included also to ease the process (though the eventual goal is to have the orientation contained in the encoding itself).

I spent more time fine tuning the laser cutter properties to provide surface cavities that are of the highest contrast and precision (see Figure 5-2). Matching the image

resolution and the laser cutter resolution, as well as lowering the speed of the cutter, provided useful results. I also found that editing the horizontal and vertical spacing of the bit cavities was very important. While the laser cutter claims it can cut at a resolution of 1200 DPI, in reality, the precision is not quite that accurate. The burning of the material and effects of the moving laser head cause imperfect features to be made at 1200 DPI, leading to cavities that are merged together, or non-symmetric and difficult to read.

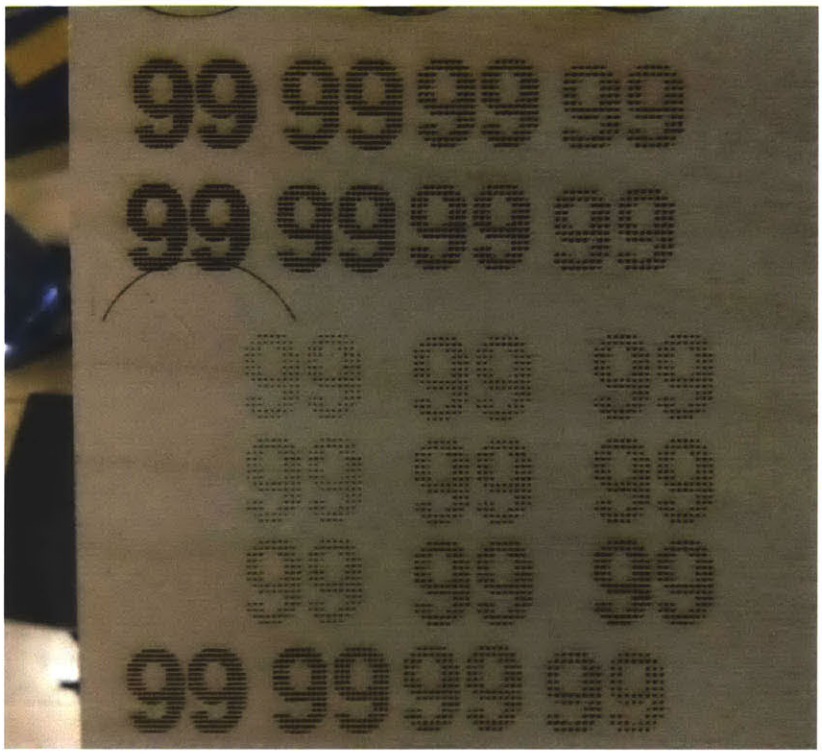

Figure 5-2: Results of laser cutter testing to produce surface cavities that are of the highest contrast and precision.

I also decided to put a horizontal orientation line below the '99' that can be used to determine the correct orientation of the encoding. While I ideally would like to use the bit cavities themselves to perform the orientation step, I included the line as a backup measure given the short timeline I have available to finish this project before the event. The orientation is the width of the '99' numbers and is a single pixel wide. Given that it is the largest 'blob' on the token, it should be fairly straightforward to detect.

To generate the drawing file, I used Processing to procedurally generate all of the varying token patterns. My technique was to start with a full token - i.e. all of the bits as cavities - and then based on the token number, draw white circles over the desired '0' bit locations, effectively removing the cavity. I created a 16x16 array of tokens, yielding a total of 256 varying ID numbers. The tokens have a total of 32 bits, which is split up into 4 redundant codes of 8-bits. The redundancy is used in the event that the decoder misses a bit, it still has other opportunities to read the correct value (and additional readings to verify the correct value). A sample of several final tokens in Figure 5-3.

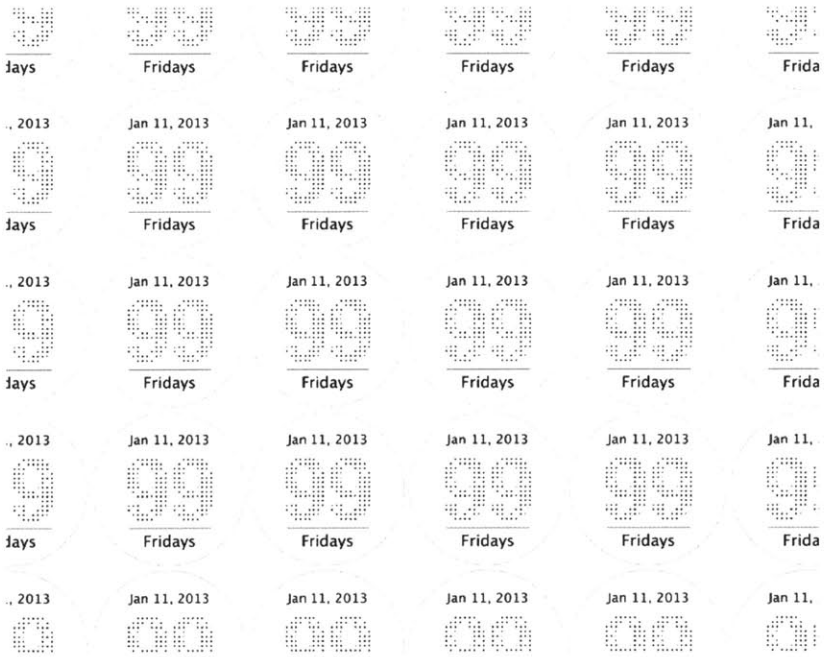


Figure 5-3: Several tokens generated for laser cutting.

The Processing code to generate the tokens can be found in Appendix A, Section A.3.

### 5.1.2 Physical Decoder Box

One issue that I immediately ran into after adjusting the code for the tokens, is that the blob-detecting algorithm does not handle interior blobs well. That is, if the edge of the token itself is detected as a blob, nothing inside that blob (i.e. all the data) will

be detected. The solution I found was that the edge transition from token to token-holder had to be sufficiently homogeneous in comparison to the transition between cavity and no-cavity. This way, a threshold can be applied that will wash out the edge transition, but not the cavity transition. I found that using a white acrylic and bottom illuminating the whole support structure allows for such conditions. Thus, to consistently provide these conditions, I built a simple testing box that has a groove for the token to fit in, and a secured camera below. This allows me to simulate a final environment where the token position relative to the camera is known, and the illumination is standardized. The camera that I used was a PS3 Eye webcam. I added a lens to the front of the webcam to allow me to focus at a shorter range - allowing the token to take up the entire field of view of the camera, while also being in focus. There is an addressable RGB LED strip that is driven by an Arduino board that is used for illumination. This token reader box has the added benefit that it simplifies the user interaction and makes the alignment issues nearly fool-proof.

The final box was created from 1/2" thick white acrylic (see Figure 5-6). There is a middle platform that holds the camera and the camera is kept perpendicular by having it's flat edge glued perpendicular to the plate. The box is press fit, and all sides are fixed in place with acrylic bonder, except for one panel that serves as an access panel. Below the middle panel is the an Arduino that drives the LEDs and a hole that allows cables to be fed through the back. The LEDs are arranged in a circular loop that surround and face inwards towards the camera. A video of the final construction working is shown below. A green flash indicates a correctly read token, a red flash indicates that something was found, but it was not an identifiable token.

## 5.2 Live Decoding Software

### 5.2.1 Bit Detection

To begin detecting the encoding on these tokens, I started with the base code that was generated and explained in Section 4.2.2. Luckily, since there are only 'circular'

cavities on these tokens, the code can be simplified and does not need to differentiate between triangles, rectangles, and squares. The bits are detected by knowing the location of each bit in relation to some orientation marker. In this case, the orientation marker is the horizontal line below the '99' marks. Once the token is rotated (in software) to the correct position, the bit locations are identical between all tokens. This of course requires that the camera be perpendicular to the token face, so that the rotation of the token does not skew locations. This is taken into consideration in design of the final enclosure. An image showing the software identifying the potential location of bits is given in Figure 5-7.

### 5.2.2 Orientation

To correct for orientation I use the horizontal reference line that is inscribed below each of the '99' marks on the tokens. I detect this line by looking for the blob that has a radius greater than a certain set value (as determined through trial and error) and a width to height ratio that is greater than a set value. The line is unique on the token as one of the only features with a long, skinny profile, and this is used to the advantage of the code to detect it. Once the orientation line has been found, its endpoints are calculated and a line is fit to that. The slope is calculated, and the arctan is taken to find the angle from horizontal. Of course, the arctan may return a value that orients the token upside-down, with the orientation line being flat at the top of the image, rather than the bottom. To fix this, I do a check after the rotation to ensure that the line's center is below the middle pixel of the image, and if it isn't rotate by an additional 180 degrees. The rotation is performed every two seconds, to handle the introduction of new tokens with varying orientations.

The tokens and token reader were used at several events (and continue to be used in current 99 Friday events). The tokens are fabricated uniquely for each event and distributed a few days prior. A review of the various encoded tokens that have been generated are shown in Figure 5-9.

## 5.3 Collected Data

During one 99 Fridays event, data was collected by the token reader. The reader recorded the ID of each token read, as well as a timestamp of when the token was read. The token IDs were not connected with individual identities, so there was no concern of gathering potentially personal data. A total of 306 tokens-readings occurred over the course of the 6-hour event. No values were out of the expected range of inputs, so while there exist some permutations of bit shifting that would go unnoticed, there were no detectable errors in this set of data. There were some ideas that such information could be used to promote safe habits in an event such as this one. For example, if the system sees that an identical ID was scanned too many times or too frequently, indicating that a person was consuming too much alcohol too quickly, the box may be able to flash a distinct color or pattern, letting the user know that they should slow down.

### 5.3.1 User Feedback

Some of the most useful feedback from this event came in the form of observing people's behavior with the system. It was a useful tool for realizing which aspects of the tokens and the reader were intuitive and which were perhaps poorly designed. One recurring behavior that I noticed was that people tried to self-orient the tokens. Even though the reader can accept a token at any rotation, many users would often try to put the token right-side-up as they deemed from the depiction on the token (either the face or numbers). While this didn't hinder the system in any way, it was additional work by the user that was unnecessary (and perhaps a point of confusion about how the system worked). Another behavior that I noticed a couple users displaying was the tendency to continually rotate the token once it had been placed down in the slot. I assume their motivation was that the continually rotating the token would eventually land it in the right orientation. However, this actually would have the reverse effect given the current implementation of the software as the code re-orients each new token once, and then processing given that orientation. A continually rotating token

will likely not be correctly identified.

The typical response from users was generally positive. People seemed to enjoy the feedback of the flashing box upon successful reads, and many expressed interest in seeing the tokens used at future events. This suggests that many found the means of interaction to be not only a simple one, but an effective and enjoyable one, perhaps giving support to the idea that using physically tangible encoded objects is a valuable means of interaction.

## 5.4 Test Results

To provide further performance characterization, I ran a test using 10 tokens (each with a unique ID). Each token was scanned 10 times, that is placed on the token reader and then removed, and the results were collected. The token is left on the reader until a decision is made (as signified by green or red flashing). The token is randomly rotated on each placement. The 10 tokens were randomly sampled. The tokens varied in their quality, as chosen from the random sampling of the manufactured lot. Some of the tokens have slightly altered wood grain patterns while others seem to have burned a slight amount more, producing more discoloration. The table of results from this test is given in Table 5.1.

| Token ID | Correct Reads | Incorrect Reads |
|---|---|---|
| 128 | 7 | 3 |
| 66 | 4 | 6 |
| 68 | 6 | 4 |
| 93 | 9 | 1 |
| 57 | 5 | 5 |
| 60 | 0 | 10 |
| 75 | 10 | 0 |
| 153 | 10 | 0 |
| 67 | 1 | 9 |
| 225 | 5 | 5 |

Table 5.1: Token Testing Results.

A second identical test was run a few weeks later after a new set of tokens were

fabricated. These tokens incorporated new design features in response to the failure points of previous tokens. Notably, a smaller central image and smaller orientation line were used. The test was again a set of 10 tokens each scanned 10 times. The results for this second test are shown in Table 5.2.

| Token ID | Correct Reads | Incorrect Reads |
|---|---|---|
| 131 | 9 | 1 |
| 143 | 10 | 0 |
| 56 | 10 | 0 |
| 126 | 10 | 0 |
| 240 | 10 | 0 |
| 64 | 9 | 1 |
| 159 | 10 | 0 |
| 155 | 10 | 0 |
| 41 | 10 | 0 |
| 129 | 8 | 2 |

Table 5.2: Token Testing Results for a second set of tokens.

As is shown in the table of read results for the second test, the newly incorporated design features lead to much more robust decoding.

### 5.4.1 Read Error Rates

Qualitatively, inspecting the tokens as the results from Table 5.1 were generated, there seemed to be three main reasons for token-read failure: 1) the token had a large and strong wood grain discoloration across an important part of the encoding, 2) the token was fabricated slightly askew, resulting in a orientation line that was not entirely in the view of the camera, and 3) dust from the cutting process created slight discoloration across the orientation marker resulting in the thresholding not adequately isolating the marker. These mostly seem to be challenges that could be addressed in the fabrication process. Perhaps more delicate cutting and machining would result in more consistent tokens. Furthermore, regarding the wood grain, perhaps a more homogeneous material could be used for future tokens (or large wood grain marks could be avoided in favor of 'cleaner' sections of wood).

From the 100 scan samples, there was a resultant 57% read accuracy. While this seems to be relatively low, it should be noted that two tokens with strong discoloration led to a 100% and 90% read failure. This suggests that higher fabrication standards of quality would lead to significantly better read accuracies. Furthermore, the data also shows that only one token was completely unreadable, while the others could be successfully read on a separate placement with differing rotation.

From the second test, as documented in Table 5.2, the 100 sample reads had a resulting read accuracy of 96% – a dramatic increase from the test with the first set of tokens.


Figure 5-4: The fabricated tokens on the laser cutter bed after being manufactured. Each token has 32 bits if unique identifying information.

Figure 5-5: A close up showing the illuminated opening of the final token reader.

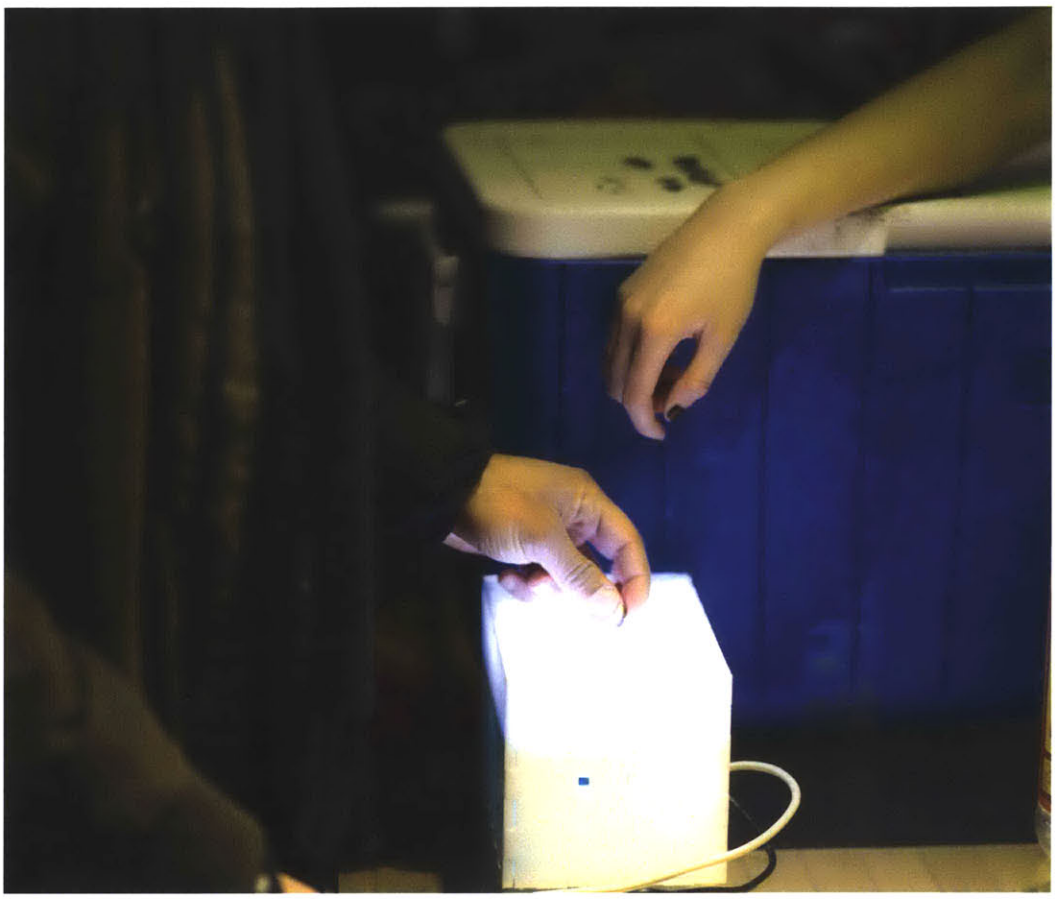

Figure 5-6: A user placing his token onto the token reader.


Figure 5-7: The software identifying potential bit locations in blue.

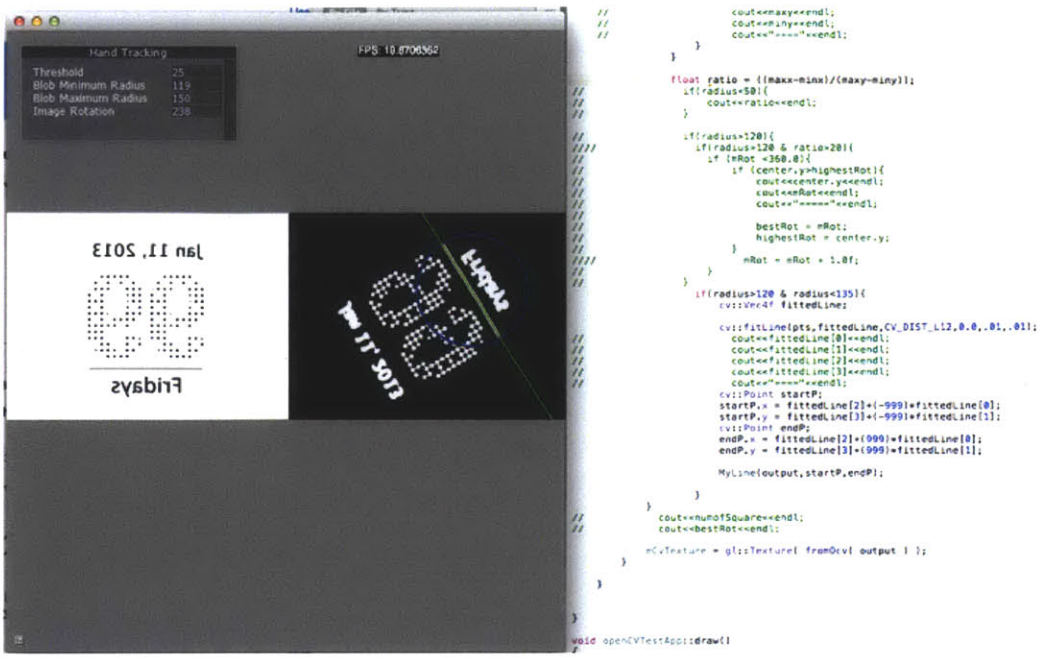


Figure 5-8: Software screen showing the orientation line being detected and the fit line being drawn.


Figure 5-9: A sample of the various encoded tokens that have been generated.

# Chapter 6

# Encoding 3-Dimensional Surfaces

This chapter explores the techniques and challenges of fabricating non-planar, 3-dimensional objects whose surfaces have been patterned with a texture encoding. Specifically, leveraging what was learned in the planar encoding chapter, these textured encodings will be fabricated from a series of deterministic cavities that will be optically read and decoded. The work presented here focuses on using 3D printers to fabricate the 3D object, or a 3D negative of the object which can later be cast with a rubber or plastic.

## 6.1 Simple CAD Encoding

To begin, I look at defining a workflow for simple 3D objects. I start with simple geometries to build some of the fundamental ideas and requirements. To start, the common 3D modeling software SolidWorks is used. One downside is that Solidworks does not allow for robust scripting, as do some CAD softwares like Rhino. This means that the beginning efforts will be mostly manual. However, beginning with a more manual process may be useful for giving more specific insights into the pros and cons of different techniques.

Beginning with the most basic 3D geometry, a cube, common extrude cut functions can be used to cut cavities into a surface as defined by a 2D sketch element. This is the most straightforward technique, but is still useful for some situations. A sample

cube has been cut with a cavity test pattern composed of circles and triangles and is shown in Figure 6-1.

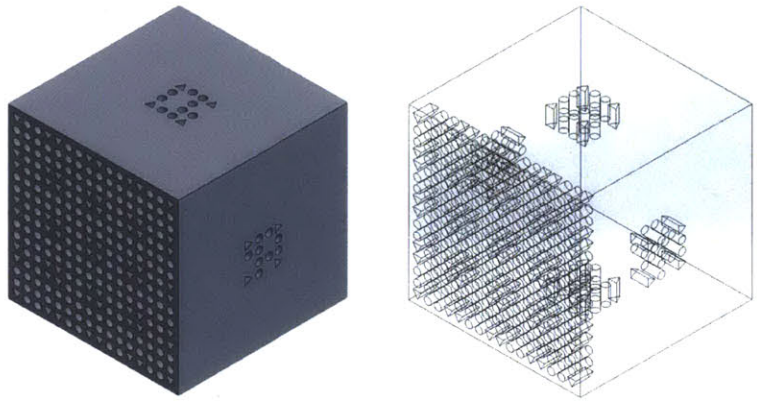

Figure 6-1: A simple cube with extruded cavity encodings.

For certain non-planar surfaces, SolidWorks provides a wrap function. The wrap feature lets you take a planar, cylindrical, or extruded surface and wrap a 2D sketch element around it. Furthermore, there is a deboss option that allows you to make direct cavities from this wrapped feature. While this achieves the exact functionality needed, it is only suited for a limited domain of elements. The wrap feature does not work on arbitrary surface elements or spherical elements. This means that additional techniques must be implemented to achieve the generality I am seeking. A sample cylinder model has been modeled using the wrap feature to apply four differing cavity codes to the surface. An image of the cylinder example is given in Figure 6-2.

As mentioned above, the Wrap feature doesn't work on spherical objects. One possible option is to perform extrusion cuts and selecting to do them from the surface of a particular face. This essentially projects a 2D sketch pattern onto a face and performs the extrusion from that point inwards. This has a large downside in that the projection does not conform to the contours of the object, but if the codes are sufficiently small in comparison to the contour, this may be acceptable on a large scale. This is probably the biggest weakness in using a tool like SolidWorks and should be taken as a sign that alternative methods will be required for more complex geometries. A sample of this technique, projected onto a sphere is shown in Figure

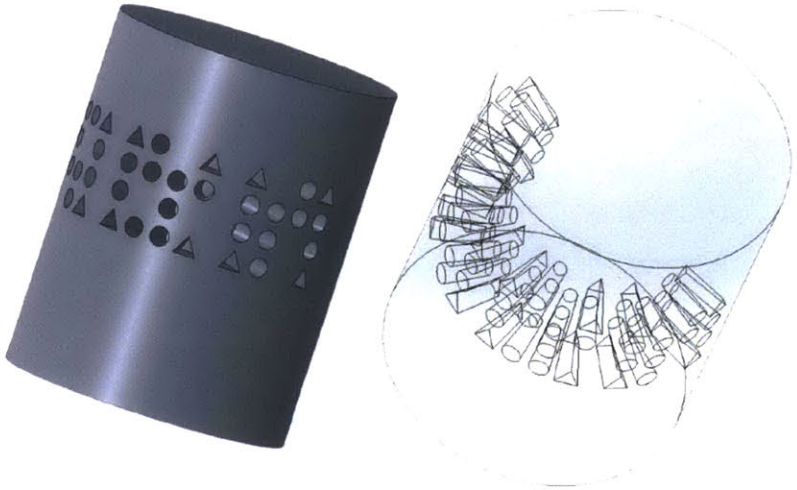

Figure 6-2: A simple cylinder with wrapped cavity encodings.

6-3.

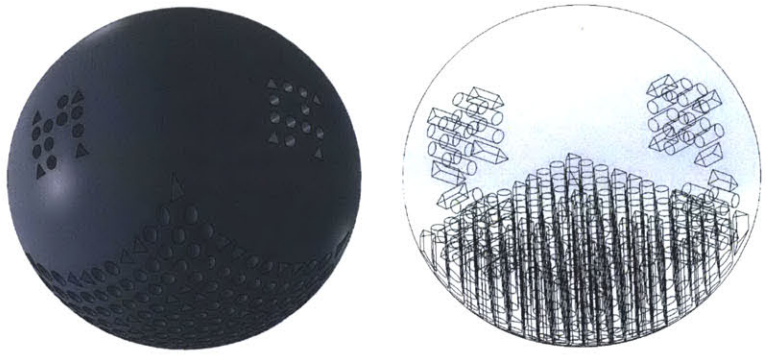

Figure 6-3: A simple sphere with extruded cavity encodings.

### 6.1.1 Simple CAD Fabrication

Despite the issues with modeling more complex models with the above technique, I printed the test cylinder and test cube that are shown above. I created three versions of each shape, each version at a varying scale and size. I created a 1", .75" and .5" diameter version of each of the shapes. The result is shown in Figure 6-4.

While the 3D printer was able to accurately print all of the small features as

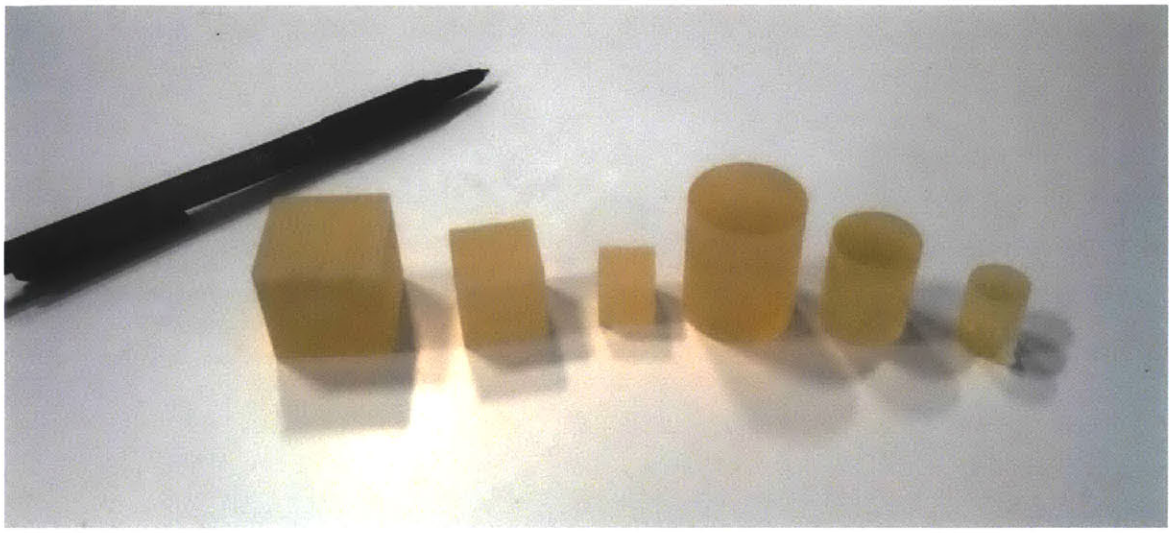

Figure 6-4: 1", 0.75", and 0.5" diameter cubes and cylinders 3D printed to test the printer resolution. A pen is shown at top for scale.

defined in the CAD models, the fabricated objects did suffer from the minuteness in a different way. The 3D printer used produces wax support material around the printed object to provide structure to any component that may be hanging or 'floating' during fabrication. This wax is usually melted off in a warm oven and then dissolved in a solvent bath, thus removing the wax support and leaving the printed object clean. Unfortunately, the wax within the small cavities of these models did not melt away as desired. The adhesive forces between the very narrow walls of the object seem to be sufficiently strong as to hold the wax in place even once melted. Furthermore, as previously suggested, the resultant model is far too translucent to robustly detect encoded cavities. A solution to this problem would be to 3D print the negative of whatever object is desired, and then to cast rubber or plastic to produce the final object,

## 6.2 Encoding Complex Geometries

While it is straightforward to map a 2D pattern to 3D objects of revolution, such as cylinders and cones, spheres and more complex bodies do not have as straightforward

of a method. While these simpler objects have a proper mapping of 2D codes, more complex objects simply had a flat projection that produce less than desirable results. As a result, it is necessary to explore a different workflow and figure out a solution more generally applicable. Displacement maps can serve as a solution.

### 6.2.1 Displacement Maps

Bump maps are a familiar and often used tool in the field of computer graphics. Bump maps effect the lighting that falls on a model to produce the appearance of textures or bumps. That is, a bump map doesn't actually change the mesh geometry of the object, rather just the behavior of the light that falls on it. Displacement maps are similar to bump maps, in that they can be applied as a 'texture' to the object, but critically, they actually deform the geometry of the model. An open source and powerful tool that is capable of displacement mapping objects is Blender.

A key component to displacement mapping is performing what is called UV unwrapping, or UV mapping. This process entails 'unwrapping' the 3D model to a 2D surface, defining a 2D bitmap and overlaying the two. The magic comes in the ability to define seams along the 3D model where the unwrapping should take place. This lets you define boundaries for complex 3D objects that otherwise would not have a trivial 2D unwrap. Blender, being a well developed piece of software with a vibrant community, provides many tutorials and examples on how such techniques can be performed.

Displacement maps work by physically moving mesh faces up or down (as defined by an axis or by the normal of that face). The result, is that the displacement mapping is only as accurate as the underlying mesh. You cannot have small surface displacements represented on a low-polygon model as the entire face is moved as a single unit. To work around such issues, it is therefore important to subdivide any many that is being used to an appropriate resolution. The downside of such a technique is that to achieve the resolution and accuracy desired, CAD models with many millions of faces are often needed. Even with modern computing power, manipulating such large mesh datasets can be very slow and tedious. Thus, it is

important to weigh the tradeoffs of higher accuracy versus modeling capability and to find the proper balance.

To document the process of applying a displacement map, I have created a step-by-step guide that explains the workflow in Blender. The workflow is subject to change based on the given geometry and other detailed considerations, but the provided process is a good baseline. The steps are listed below and associated images are given in Figure 6-5.

1. Import .STL File
2. Split screen and choose to make one display UV view
3. Either make new image or open existing image
4. Select model, enter Edit Mode and select Vertex selection
5. Shift+select a line that will define a seam. Cntrl+e and select 'mark seam'
6. Press 'a' twice to deselect all and then select model. Click u and select unwrap
7. Cntrl+a, to normalize unwrapped pieces, Cntrl+p to fit to image
8. Create material, Textures
9. Add Displacement and SubSurface Modifiers
10. Apply and export STL

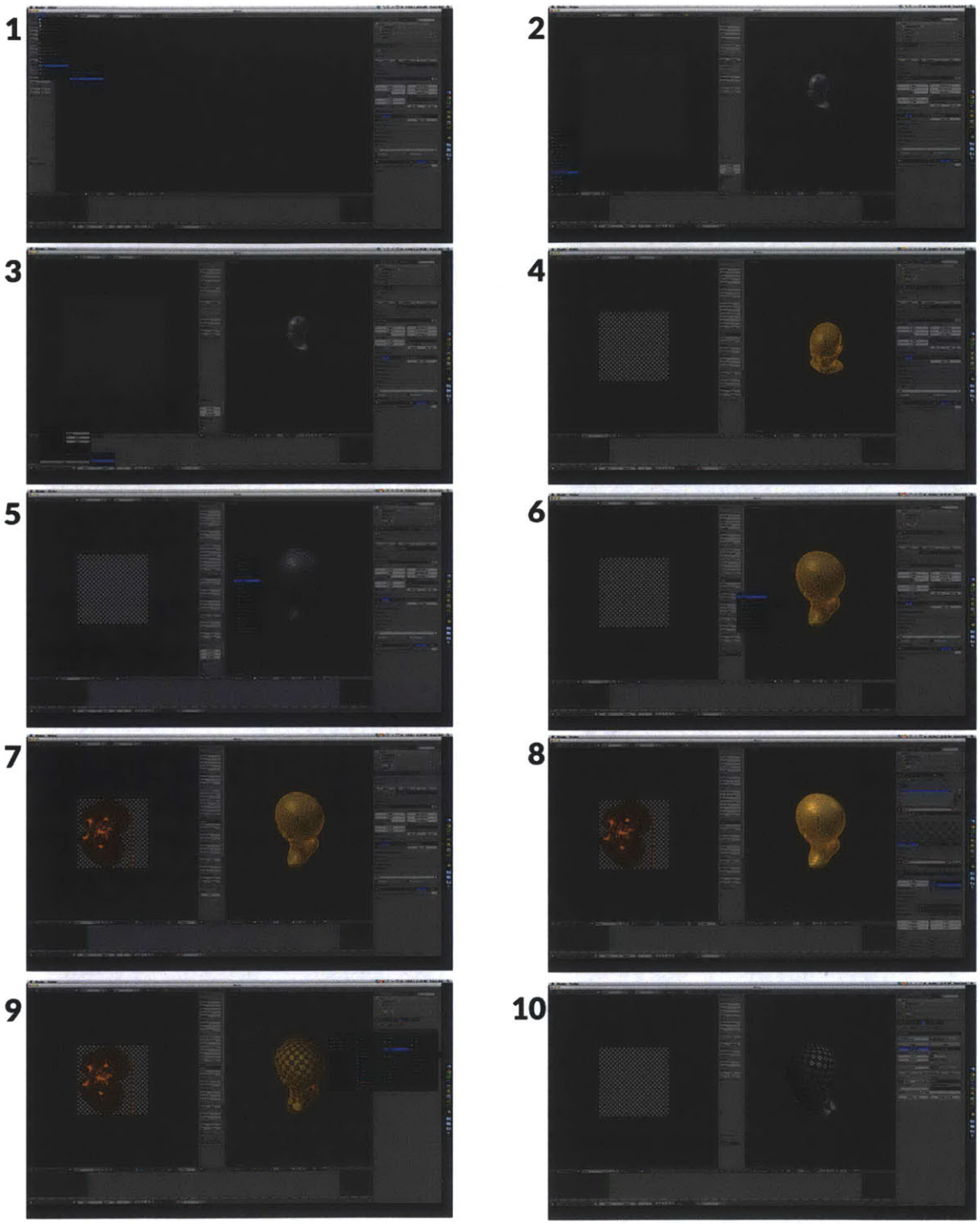


Figure 6-5: Steps for applying a displacement map in Blender.

Instead of using a testing geometry bitmap, we can now apply an encoded pattern as a displacement map. If we take a series of circular primitives as the baseline for

our encoded pattern, the result can be something as is shown in Figure 6-6. Note in this figure, that the mapping is not always perfect. The encoded pattern consists of circles of identical size, yet not all circular cavities are of the same diameter. In some instances, the mapping is not completely perfect, and results in skewed or stretched features.

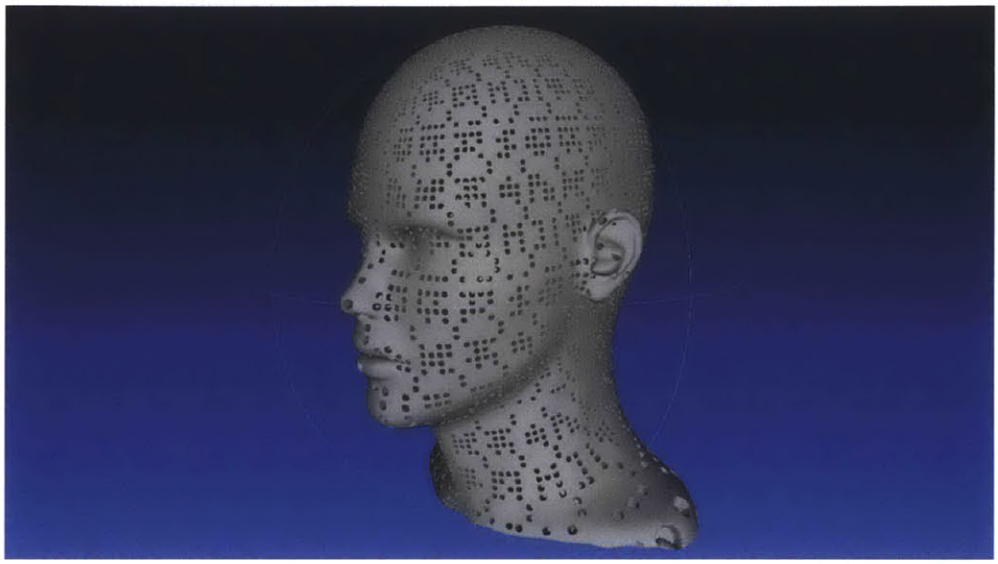

Figure 6-6: STL model of head with a circular encoded pattern applied as a displacement map.

### 6.2.2 Subdividing Meshes

One feature that I think is important to note concerns the effectiveness of subdividing a surface mesh. While there are many algorithms for subdividing a surface (some provide a smoothing function, others are just splitting triangles in two), for the sake of displacement mapping, it is common to apply a very simple subdivide. That is, we do not wish to alter the geometry in any way, we just want more faces. As a result, Blender will simply chop the mesh triangle into two in what it deems simplest.

However, depending on the characteristics of the mesh, this may not always provide the best results. If we take for example a simple contoured surface (as shown in Figure 6-7), sometimes the simplest way to model this with triangles is to use long

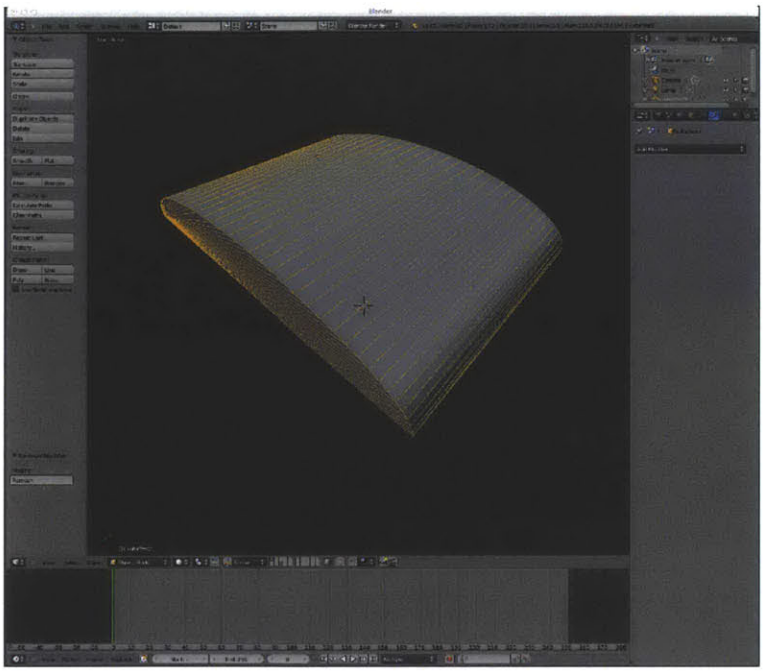

Figure 6-7: Low-poly mesh of a contoured surface. Note the long stretching triangle faces.

triangles that extend across the entire surface. When subdividing a mesh like this, what can often happen is that the triangle will be dividing lengthwise as opposed widthwise. This results in two long skinny triangles, as opposed to two medium-length medium-width triangles. This is not the desired effect. Figure 6-8 shows the effect of taking such subdivisions through several iterations.

These resulting polygons are not well suited for displacements because they effect a very wide area, as opposed to a small one. To fix this situation, one can simply first re-mesh a surface, export to STL, and then re-import the new STL. The result is that Blender will re-mesh a surface by creating smaller and smaller square (or rectangular) mesh faces. Upon export, these faces are converted to near-equilateral triangles. Thus, once the model is re-imported, rather than the long stretching mesh faces, one has small, roughly-square faces. Now, upon subdivision, each face becomes more and more point-like, which is the ideal situation for applying a displacement map. See Figure 6-9

A finalized mesh that has been created and will be used in final application use

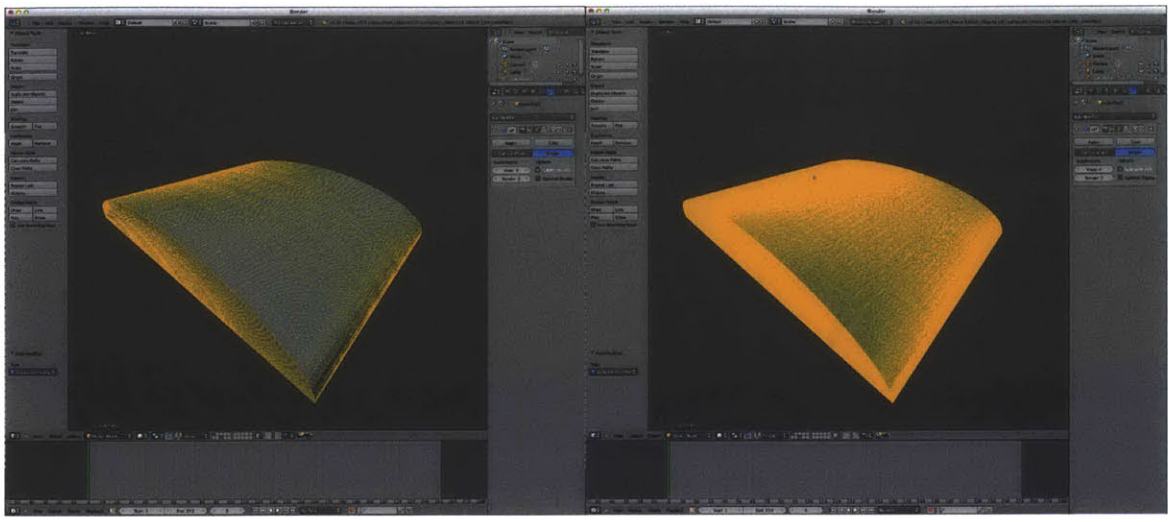

Figure 6-8: Low-poly mesh of a contoured surface after multiple subdivisions (left fewer subdivisions, right more subdivisions). Note the long stretching triangle faces.

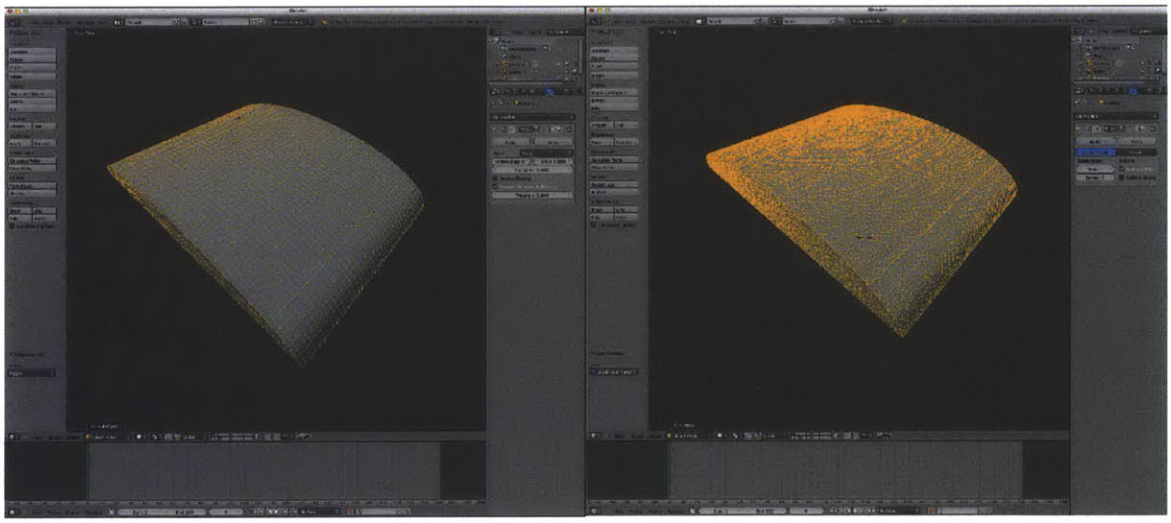

Figure 6-9: Re-meshed surface. Left is after the base re-meshing, right is after subdividing the re-meshed surface. Note the nice, point-like triangle faces that contrast the faces seen in Figure 6-8.

cases is shown in Figure 6-10. This mesh of a race car was produced by taking a low-poly model, selectively subdividing the surface (that is, only subdividing the regions where surface encodings will be applied), applying a displacement map, and then subtracting that entire mesh from a rectangle. The result is a negative that can be used as a cast after 3D printing. The mesh features are small triangles that are

placed on a grid, and then randomly shifted vertically and horizontally, and rotated. Note the loss of fidelity in the edges of the triangles as a result of the mesh not being sufficiently subdivided. The decision to finalize the mesh in this state came from analyzing the tradeoff between hi-poly meshes and sufficient modeling performance. Subdividing any further for more defined triangular features renders the model in a state that is nearly unworkable.

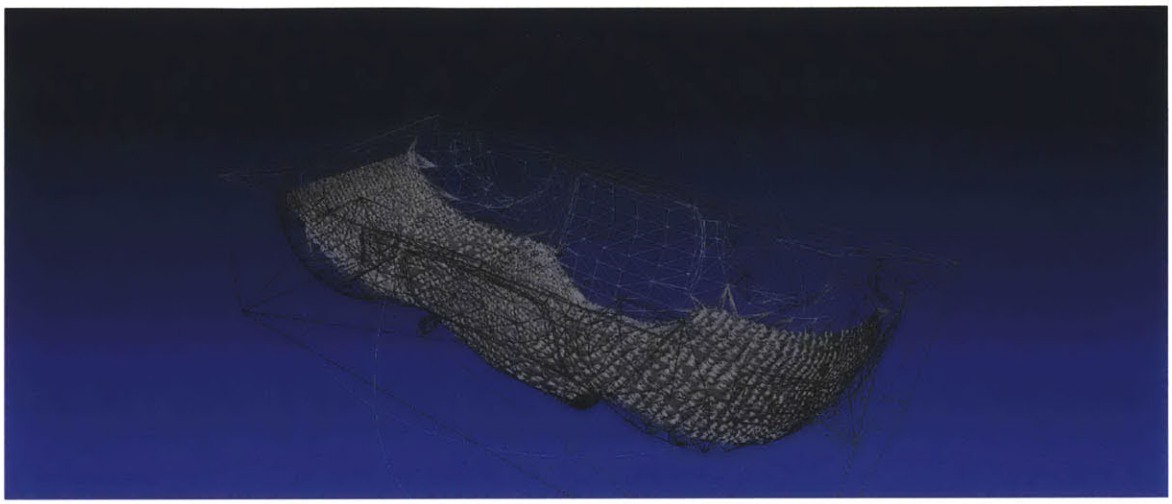

Figure 6-10: Mesh representing the negative of a surface encoded race car.

## 6.3 Fabricating Complex Geometries

Fabricating a 3D model once the STL has been generated is typically straight forward with 3D printers. The typical workflow is that the 3D STL model is imported into the 3D printer software, analyzed, printed, and then a final step of removing whatever support material that exists finishes the procedure. However, due to the peculiarity and sharp features that are introduced in many surface encoding iterations, the 3D printer software does not always operate how one would expect.

Specifically, issues with inverted normals (all faces must have an associated normal component that defines the outside of the object), holes in the mesh, and other geometric peculiarities can send the 3D printer into a job that does not finish in a way that resembles the graphical rendering. An attempt to print the 3D head model,

shown in Figure 6-6, produced the result shown in Figure 6-11. The problem was that the mesh was actually generated as two sets of 'shells' rather than a single mesh surface. The result is that there was a very small discontinuity between that caused a layer of unwanted support material to be printed through the middle of the head model. Such errors can typically only be cause once the model has been printed, leaving a certain degree of chance to the entire fabrication process when using a new model.

Figure 6-11: 3D printed result of using a bad mesh model.

As further reference, the car model that is shown in Figure 6-10 went through over 10 revisions before a suitable mesh was generated. Issues with creating a negative model through boolean operations, displacement mapping very small features, and high polygon count caused for some very rough and undesirable meshes. Once a final mesh has been produced, though, the 3D printing process is straight forward. Moving forward with the race car example, once the 3D negative had been printed, I was able to simply cast Silicone rubber to generate the positive of a race car model with an encoded surface. The encoding used on this surface is composed of many 7x7 triangle arrays. Each array in the triangle is shifted by some random amount and also

rotated by a random number. The result is a deterministic pattern of high density and uniformity. The resultant car, and processing steps, can be seen in Figure 6-12.

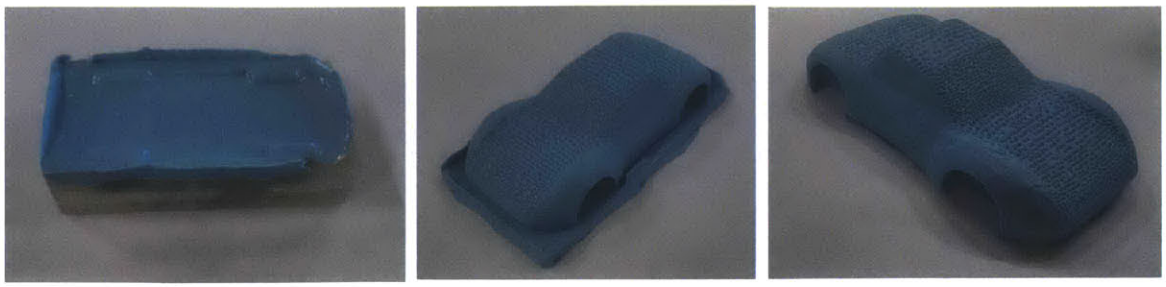

Figure 6-12: Process and result of fabricating the race car with encoded surface.

## 6.4 Decoding 3D surfaces

The important task of decoding the surface features is now approachable. The main dichotomy in techniques for doing this comes down to deciding whether a pattern matching approach (i.e. looking for known features and identifying) or raw decoding approach (i.e. looking for primitives and decoding information straight from those primitives) is to be used. A raw decoding approach is like that used in Chapter 5. While this approach worked very well in the planar case, given the fixed environment, there are several strong benefits to using a pattern matching technique once moving to 3D non-planar surfaces in a dynamic environment. The strongest case for pattern matching in such a scenario is that identifiers can be read even with the loss of bits. That is, if a pattern is seen to be 98% matched, for example, (the remaining 2% being lost to obstruction, damaged material, etc) a correct ID can still be made. This is not always the case in raw decoding, where a flipped bit can entirely change a message. Such errors can be approached with error-correcting codes and with check-sums, but such techniques are useful when a message can be resent. In the case of a physical encoding, if a piece of the material is broken or damaged, those bits can never be resent, so simply detecting an error will not be useful. Pattern matching does have the disadvantage that it relies on some known library of patterns, which increases the

data requirements and backend needed to successfully implement a scaled system. However, given the processing power, memory, and cloud access of many modern mobile devices these increased requirements are an acceptable cost. Thus, for this system, I have decided to implement a pattern matching library to decode 3D encoded surface objects. In this case, I am leveraging the freely available Qualcomm Vuforia toolkit.

Such an implementation comes with the further advantage in that any object of sufficiently high contrast can be easily identified and decoded. That is, a large range of 3D materials and things can fit into this self-descriptive world ecosystem. For example, stitch patterns can be used as encoding on fabrics, as can embroidered patches and screen printed decals. This opens the doors to creating a wide range of fabric based encodings.

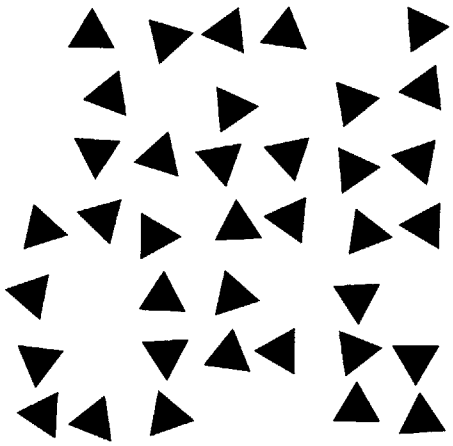

Figure 6-13: A sample triangle code used to encode a surface area.

Returning to the case of the 3D printed car, it is important to consider what types of surface encodings can be robustly detected and interpreted given the chosen pattern matching algorithms. The image processing routines work best with patterns of high contrast and sharp edges. Gradients, rounded edges, and smooth transitions are not well detected and should not be used to encode information. Given that, I base my encoding patterns on arrays of black triangles over a white surface. In the implemented system, the minimum code size is set to a 7x7 array of triangles, where

each triangle is individually shifted and rotated by a random amount. Figure 6-13 shows a sample code. To encode the 3D printed car, a large array consisting of these smaller 7x7 arrays is created. This large encoding array is then used as the image that dictates the displacement mapping applied to the model.

## 6.5 Characterization

### 6.5.1 Feasible Data Density

Given the resolution 3D printer being used and certain limitations in the CAD modeling (i.e. the workable resolution to which a surface can be subdivided), a minimum implementable bit density can be calculated. I have found that each 7x7 triangle array can be reliably produced on an area of 0.55"x0.55". This yields a surface area of 0.3025 square inches. If we simplify and assume the 7x7 array of triangles can only be encoded through the presence or absence of a triangle (this is not the case given each triangle can be shifted and rotated), this yields 49 bits per array, which when combined with the figure above, yields a bit density of over 161 $\frac{bits}{in^2}$. For comparison, a typical UPC barcode encodes around 50 bits on an area of 1.5" x 1". This results in a bit density of just over 33 $\frac{bits}{in^2}$. Furthermore, many RFID tags typically transmit a 32 bit ID code.

Promisingly, as 3D printers become capable of higher resolutions it will become possible to scale this bit density to larger numbers. Furthermore, if an alternative manufacturing technique is used, bit densities of much higher magnitude can be achieved. Assuming the same 7x7 triangle pattern, Figure 6-14 shows the relationship between code size and data density. At the extreme end, a physical encoding like that used in standard hard-disk drives represents the current physical limit of data density. While this work strives to land on something much more simply detectable (i.e. with a mobile device rather than with an ultra-precise magnetic head), this may be a useful benchmark by which to calibrate expectations.

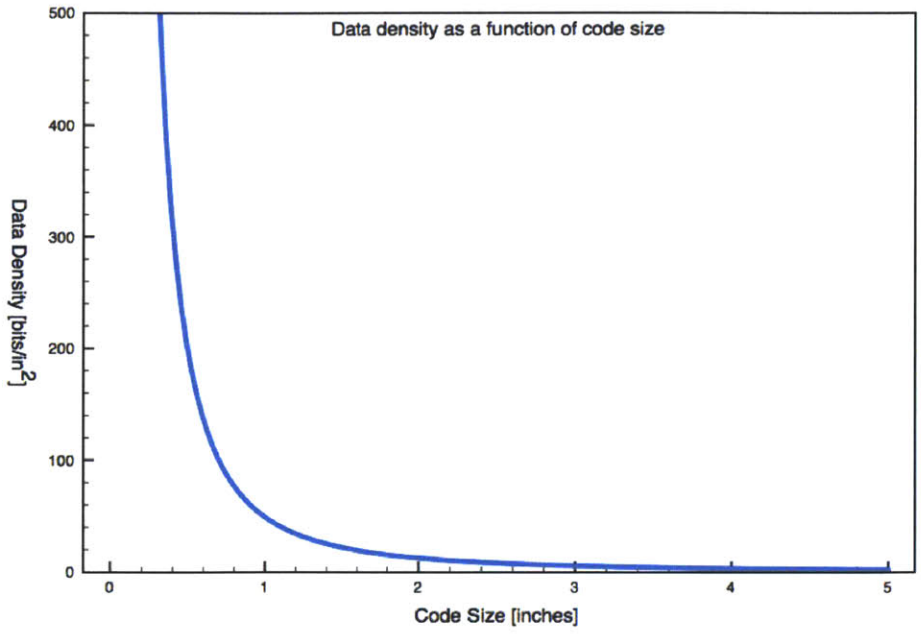


Figure 6-14: Data density as a function of the side-length of the fabricated code.

### 6.5.2 Optical Decoder Performance

The feasible data density factors in the performance of the modeling workflow, fabrication processes, and decoding mechanism. However, the bottleneck in this performance is most strongly driven by fabrication and modeling limitations. The optical decoder can perform at resolutions beyond what we've found to be workable from a fabrication angle. In this section, we provide further quantitative measurements on the performance of the optical decoder.

To test this a setup is built that allows the mobile device (camera) to be mounted a fixed distance from the code at consistent intervals. The code is a standard 7x7 triangle array printed on a piece of paper for simplicity. A mount is placed on top of this paper that raises the mobile device above the code. The base height of the mount is 0.45" and each successive height increase is 0.375". The distance is varied from a range of 0.45" to 5.7". The camera is positioned such that its field of view is center on the code laying beneath. Furthermore, a series of codes are printed at different sizes. The total code area is a square and the side lengths are varied from

0.159" to 0.55". A chart summarizing the distances at which certain sized codes can and cannot be read is given in Table 6.1.

| Code Size | Distance between camera and code [inches] | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | .45 | .825 | 1.2 | 1.57 | 1.95 | 2.32 | 2.7 | 3.08 | 3.45 | 3.83 | 4.2 | 4.58 | 4.95 | 5.32 | 5.7 |
| 0.55" | | | | | | | | | | | | | | | |
| 0.495" | | | | | | | | | | | | | | | |
| 0.446" | | | | | | | | | | | | | | | |
| 0.356" | | | | | | | | | | | | | | | |
| 0.285" | | | | | | | | | | | | | | | |
| 0.228" | | | | | | | | | | | | | | | |
| 0.159" | | | | | | | | | | | | | | | |

Table 6.1: Characterizing the read range for various code sizes. Green denotes the code can be read at that distance. Red denotes the code cannot be read at that distance.

Table 6.1 shows two read-range cutoffs. One at close distances and one at far distances. The cutoff at near distances is identical for all of the codes. This is a function of the fixed focal length of the camera that is being used. At a sufficiently close distance, the camera can no longer focus and the code is unreadable. At the other end, the far distance cutoff, the value varies between code sizes. This demonstrates the limited ability of the camera's resolution. Beyond a sufficiently far distance, the code becomes too small to be readable. It thus follows that the smaller codes thus have a nearer cutoff than the larger codes.

Also of note, the 0.285" code was the smallest readable code. The 0.228" code was also tested without the fixed structure support to verify that the code was too small to be read even at some intermediary distances not allowed by the step-wise test structure. Indeed, even when using freehand movement, the 0.228" code was too small to be read at any sufficient distance. To verify that this is a function of the camera and not of the code, a magnifying lens was placed over the code to produce a larger view. When this magnifying view was scanned, it was indeed read correctly. This proves that the limitation in the reading is due to the optics of the mobile device camera and not the printed resolution of the code.

# Chapter 7

# Interface and Backend Design

In creating a self-descriptive universe it is important to consider how a user will interact with their passive world. At a baseline, it has been established that, for this thesis, a mobile device will be used for decoding purposes. Given this, it seems straightforward and useful to have some UI elements contained on the screen of the mobile device itself. However, this poses certain limitations and it may not always be desirable to lock the entire interaction to a single mobile screen. For this reason, I have decided to implement a simultaneous browser-based UI that can work as a complement (if desired) to the mobile interface. This browser UI could be loaded on another mobile device or on a larger personal computer with a larger screen. This chapter will explore the features that are delivered between these two interface modalities as well as some of the earlier exploratory work that led to the final technical implementation.

One of the main challenges that will be addressed is the task of synchronizing the browser and mobile device. Selections and scans made by the camera of the mobile device should be reflected on the browser in near real-time. One solution would be to post data points to a database that is referenced from the browser. This database will store the points that have been 'scanned' on the physical model, and be used to render markers and annotations on other screens. Ideally, this interaction will happen in a manner such that the external displays will render the points read from the physical object in real time.

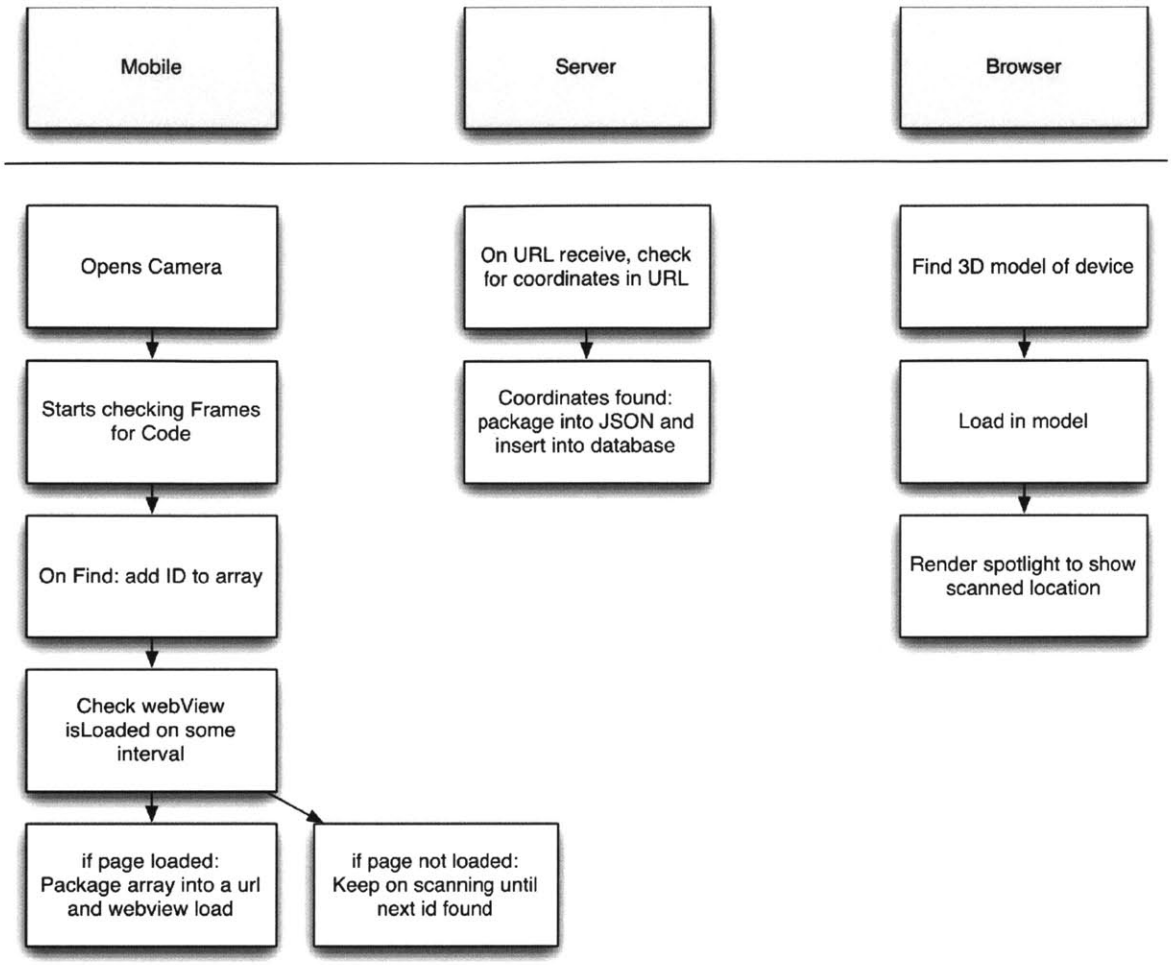


Figure 7-1: A preliminary chart describing the tasks and steps of the mobile device, server, and browser.

I initially thought of using Django. Django is a python-based MVC web framework that I've used many times in the past and am quite familiar with. I thought of a system where the mobile device would simply call a specified url, and the url handler (written in python) would make an entry to the database based on the URLs content, which would contain information about the scan (e.g. `www.example.com/?id=2463&locX=23&locY=39`). I quickly realized that I need a framework that allows me to refresh the browser display with each update of the database. I need a responsive, live interaction - similar to that seen in a collaborative browser applications, like Google docs. I began exploring Node.js and Socket.io to create a websocket style application that would maintain an open connection. Node.js and Socket.io are two

open-source free web frameworks that provide a set of predefined functionalities. Further libraries were also explored, such as Derby, Meteor, and Tornado. In the end I found Tornado to be the most flexible and appropriate platform form which to start development.

One important factor in choosing the right web framework is that I wish to render a 3D environment that can display the model CAD file and provide dynamic feedback on where the user is pointing their mobile device. For example, this rules out using the Meteor framework as, in its current state, it does not allow for arbitrary javascript packages to be cleanly integrated. Tornado on the other hand does. This allows me to use developed WebGL rendering libraries to create a dynamic 3D environment within a responsive web framework.

To describe in more detail the mechanisms and functionality of these UI elements, the following section describes exactly what I mean by a responsive framework.

## 7.1 Web Sockets

Web sockets are a recent browser-side technology that allows for persistent connections to be kept open between a server and browser client. This allows a server to push messages to a browser and have those messages reflected in the layout in real time. A common example of an application of this would be Google Docs, where multiple people can be remotely collaborating on a single word processing document, and all edits are displayed to everyone in near real-time. This is done by sending update messages to the server which are in turn pushed out to all open clients (browsers).

Such a technology can be used to synchronize the mobile device and browser layout. The mobile device can push messages to the server whenever a code is read and the server can then push a message to all open browser clients with the relevant information. Thus, in this way, the CAD model can be updated, annotations can be made, and other features showing which code was just read, all in near real-time.

The tornado framework previously introduced creates a simple interface for handling such web socket messages and provides a set of methods and function calls that

allow for easy socket manipulation and creation.

## 7.2 Mobile Platform

For the implementation of this thesis, I have chosen to develop the mobile architecture for the Android platform. I made this decision because of the relatively more open architecture that Android provides. While the base Android and iOS SDKs provide very similar functionality, Android provides lower-level access to an NDK (Native Development Kit) which exposes lower-level functionality that may be useful for this intensive application.

## 7.3 Basic Interface Functionality

The browser-side and mobile-side interfaces display a common set of data, but leverage different techniques to do so. The browser-side client uses web sockets to dynamically update the content on any given page. This allows for elements to be dynamically added or removed from a page to reflect the current set of data that is contained in the database.

The mobile side interface, however, does not have the capability of leveraging web sockets. This is due to the implementation of the mobile-side webView class that is used to render html elements within a native application. To work around this problem, the server that regularly sends web socket messages, simultaneously creates a new mobile site each time the data in the database is changed. This mobile site is then used to display the appropriate information. To display this updated information correctly, the mobile-side interface refreshes the webView element at appropriate intervals.

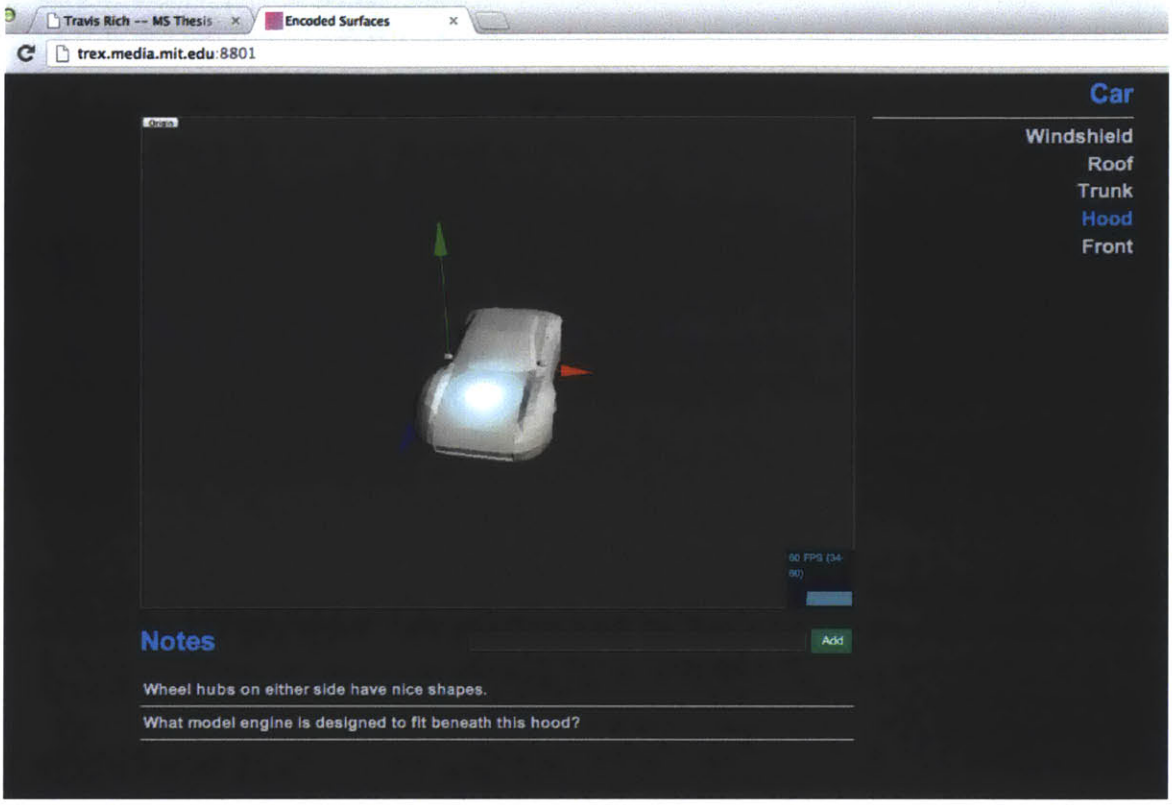


Figure 7-2: A screenshot of the browser UI being used to explore the car model.

## 7.4 Interface Features

This section will describe the features that are provided to the user and outline their implementation and functionality on both the browser and mobile side. All mobile-side features are developed and integrated into a single native application, while the browser side interface is accessible by a URL on all modern browsers.

### 7.4.1 Rendering STL Model

On the browser-side client, a 3D model of the most recently scanned object is displayed. This object is rendered in an interactive environment and the model can be rotated with the mouse and zoom-in/zoom-out functionality is available. This allows the user to see every aspect of the model of interest. An open source canvas library, THREE.js is used. THREE.js which is a javascript library for rending 3D scenes.

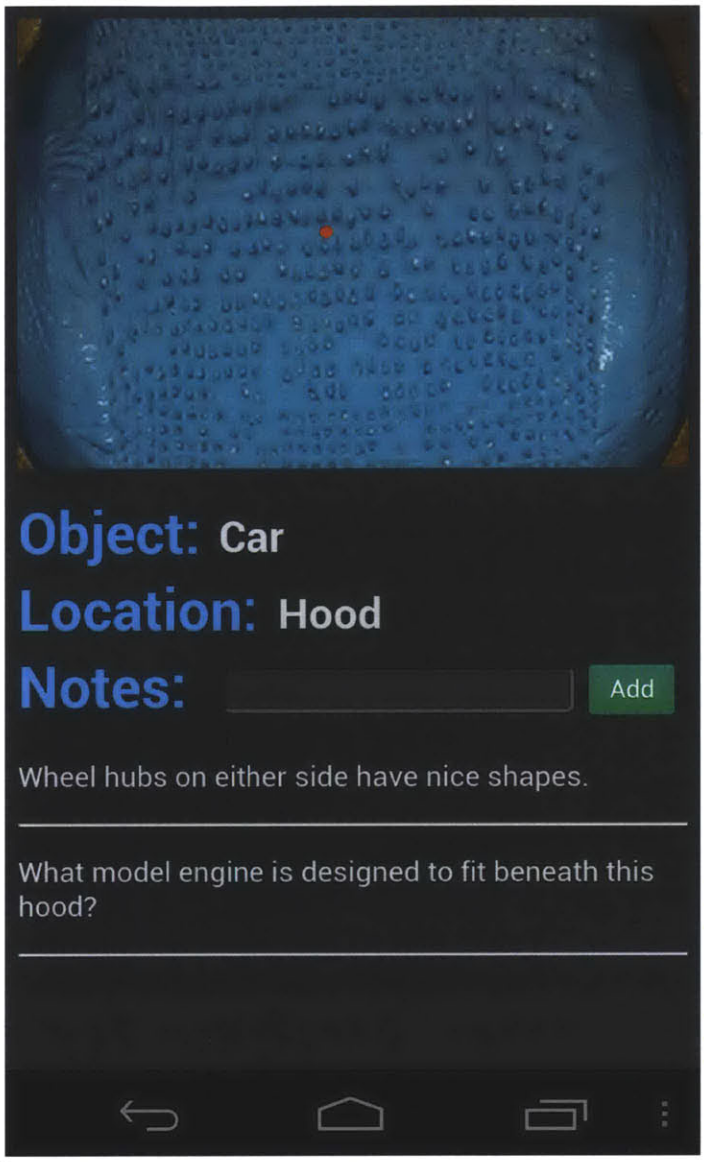


Figure 7-3: A screenshot of the mobile UI being used to explore the car model.

The library is capable of producing very impressive rendering with relative simplicity in the code.

One of the bigger hurdles was that THREE.js does not natively support the display of .STL files. Given that this is what most of my 3D CAD files are exported as, this is a problem. Luckily, there exists an example that provides simple loading of .STL objects. Looking at the source code of this example and copying over the necessary files, I was able to implement this on my own site. One minor caveat though is that

It only works on ASCII STL files (i.e. not binary STL files).

When a new object is scanned, the current mesh is removed and the new one is loaded. This can cause a slight lag on some computers (in my experience, a drop from 60fps rendering to 30fps rendering).

### 7.4.2 Spotlight Selection

Again on the browser client, in the same environment that the 3D model is rendered, a spotlight effect is created to depict which area of the object is current being scanned. This spotlight effect is dynamic in that is refreshes with each new code that is received. This can produce the illusion that the camera's field of view is represented by the spotlight in the browser-side rendering environment.

### 7.4.3 Leaving Annotations

Annotations can be left in both the browser and mobile interfaces. Annotations are associated with specific locations on the object. In the browser side, a field is present that allows users to comment on whichever part is currently highlighted within the browser-side environment (this is not necessarily the most recently scanned location, as there is a feature to browse all parts of an object in this interface). Upon clicking the 'add' button, the comment is added to the database and the server pushes the comment to all open sockets to have the note dynamically added to the page.

In the mobile side interface, a similar text field is presented for leaving comments. Comments are automatically associated with the most recently scanned location. Because the mobile interface is a static page (i.e. does not leverage web sockets), upon clicking the add button the comment is posted to the server, which dynamically recreates the mobile site, and the mobile interface is refreshed to reveal the newly generated content.

### 7.4.4 Selecting Parts of the Current Object

The browser side interface allows for different parts of the current object to be reviewed at any time. This feature was created with the intention of allowing many parties to use the browser interface remotely and to explore the object as a whole, along with any annotations that have been made. All currently scanned parts of an object are displayed in a list. The current location is highlighted in blue. Upon clicking any alternative location, the spotlight will reposition itself to reflect the change and any associated annotations will be displayed. This allows users to see both the location and the notes associated with that location by clicking through the list of locations.

### 7.4.5 Camera Viewfinder

The most important part of the mobile side interface is a viewfinder that shows the view of the camera. This view allows a user see what part of the object they are currently attempting to read. The viewfinder will overlay a red dot at the center of any code that has been successfully read to provide feedback to the user.

# Chapter 8

# Encoded Applications Beyond 3D Printing

Remembering the vision outlined in Chapter 1 for a self-descriptive universe, one will note that the universe is made up of more than simply 3D printed objects and 2D tokens. For that reason, I believe it is important to keep in mind that the ideas and processes laid out in this thesis are intended to speak to a bigger picture. A picture which cannot be entirely addressed in the scope of the thesis, but whose core concepts can be demonstrated. To this end, I have explored some alternatives beyond the work of 3D printed models and lasercut tokens. One of the driving ideas to keep in mind is that the encoding process, regardless of what is being encoded, should take place in the manufacturing of that object. If a post-processing step is required, little advantage is gained over traditional techniques of applying barcodes, printed stickers, or RFID tags.

## 8.1 Fabrics

One of the first forays into this new space was towards the domain of soft goods and fabrics. I began by looking at what the constituent parts of my clothing were for a baseline. The bias introduced here, is that I tend to where solid-color garments. The result of this is that the most 'encodable' aspects of the clothes I wear appears to

be the stitching. The fine thread that holds my pieces of apparel in place are not only an integral part of the manufacturing process, but importantly, could be altered in many ways while still providing identical functionality. This leads to the idea of encoding clothing through the use of deterministic stitch patterns.

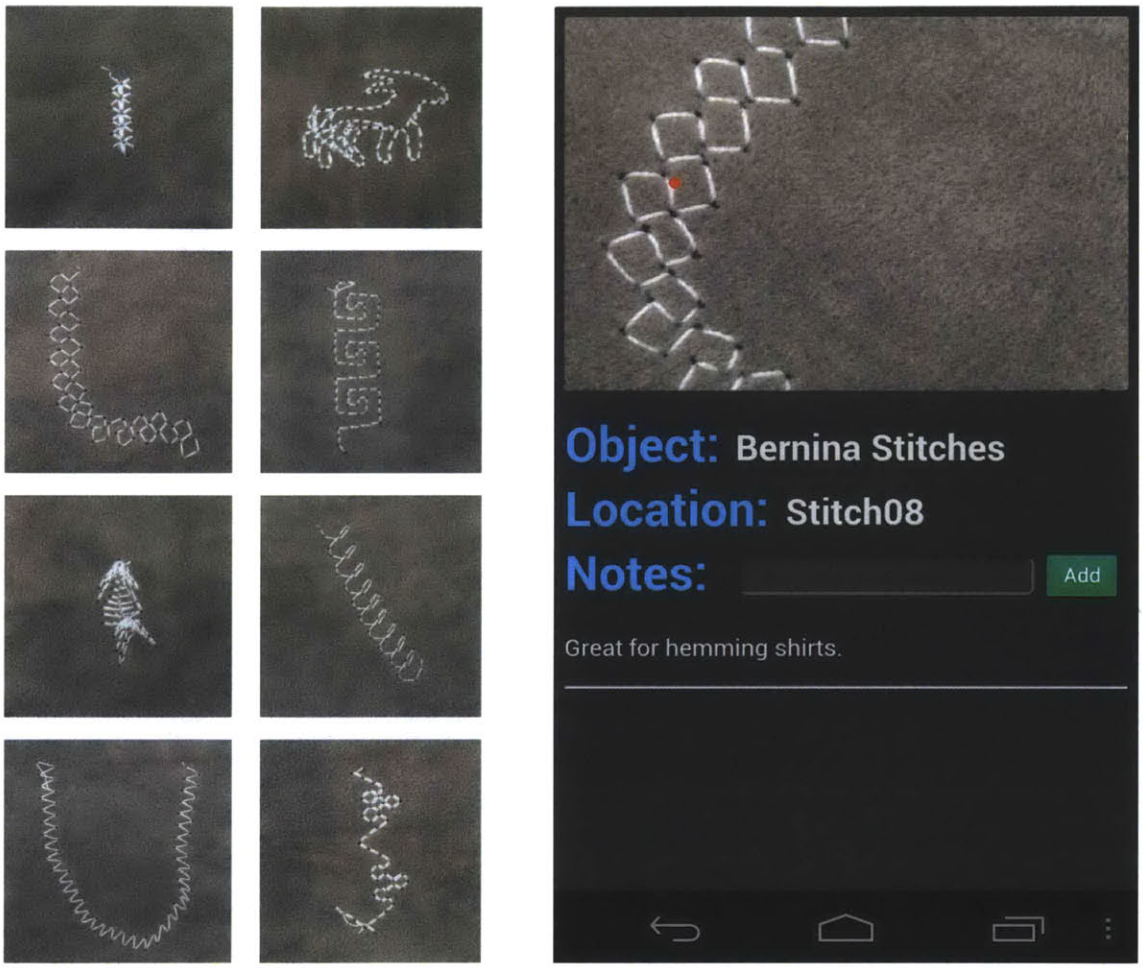


Figure 8-1: Decodable stitches and a screenshot of the mobile UI upon scanning.

To demonstrate this idea, I created an array of several different types of stitching. Each stitch, though composed of identical thread, is unique in the pattern, shape, and curvature it represents. I integrated these 'codes' into my mobile decoder application to create a simple tool that would allow you to identify the different types of stitches (and demonstrate the ability to encode identifiers into stitch patterns). In this case, the stitches were all sewn on a Bernina sewing machine, so the stitch's respective

Bernina ID number is retrieved upon scanning. Figure 8-1 shows the array of stitches that were fabricated and the mobile interface upon the successful scan of a code.

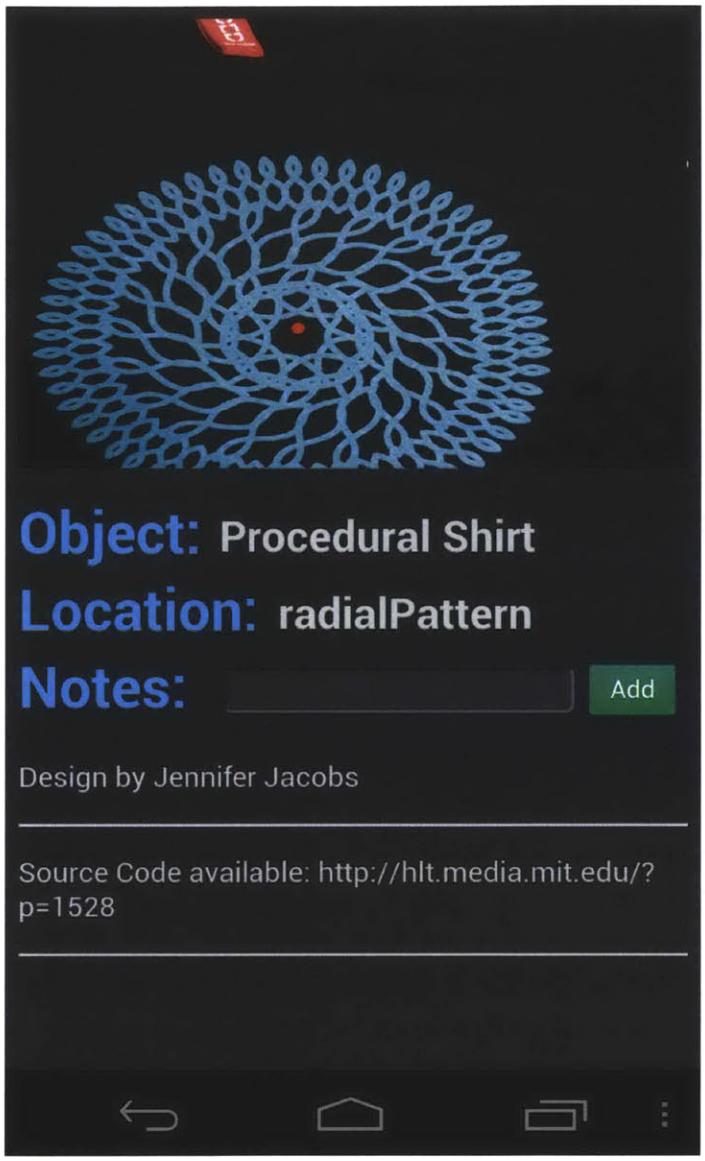


Figure 8-2: A screenshot of scanning a decodable shirt pattern, as produced as part of Jennifer Jacobs' thesis work.

I was also fortunate to be speaking with Jennifer Jacobs about her thesis around the time I was working on these stitch codes. Jennifer's thesis focuses on creating a set of tools and methods for creating algorithmically defined clothing. She is working on a software interface that creates simple to write, yet powerful, code for programmatically generating clothing patterns. This being an open source tool, she expressed

interest in somehow making her clothing patterns 'decodable' in a way that would allow wearers or people passing by to grab the original source code that generate that specific piece of clothing. Given that her software generates unique and varied patterns, it was a natural fit for my system.

By integrating the patterns Jennifer has created for her demonstrations (which have a specific representation in code) with my system, I am able to create a backend and interface that allows one to scan the piece of clothing and receive any information, source code, or attributions associated with the garment. The smooth integration of this with the exact same mechanism used for decoding the 3D printed bodies and stitches shows the value of using a pattern recognition approach. Figure 8-2 shows one of Jennifer's shirts and the mobile interface after a successful scan.

## 8.2 Food

Another interesting opportunity in the self-descriptive universe realm is to look at food. Techniques for producing food vary widely between cultures and individual dishes, so there is a wide array of techniques that could potentially be altered in a way such that the resultant food is encoded. One basic example of such an idea looks at the simple process of baking cookies.

Anyone who has baked cookies more than a few times has had the experience of making cookies that are of varying texture, color, or consistency. Given all the factors that go into baking cookies (e.g. cook temperature, cook time, cooking surface, etc), it is easy to have one small change create a slightly different outcome. A common example of this is slightly overcooking a batch of cookies and having their bottom surfaces come out darker or even burnt. Perhaps such a phenomenon could be leveraged to create an encoded cookie bottom. Such encoding may be able to deliver information regarding the ingredients or baker or perhaps instructions on how to make them yourself.

One option for trying to encode cookies would be to create a cookie sheet with a varying texture bottom. Little dimples in the cookie sheet could give rise to a textured

Figure 8-3: Failed results of the attempt to encode the bottom of cookies through layers of various cut materials.

cookie bottom which could be decoded. A first step along these lines is also to try a cooking surface of varying material. The idea here being that differing surfaces with conduct heat in different ways, resulting in a cookie bottom that similarly varies. To test out this hypothesis, I baked a batch of cookies with three different cooking surfaces: aluminum, parchment paper, and oil. A fourth set of cookies was also made that were placed flat on the cookie sheet to serve as the control group. The parchment paper and aluminum foil were cut in a way that small exposed holes were created. These holes would allow certain parts of the cookie bottom to lay flat on the cookie sheet, while the rest was insulated by the respective material. For the oil set of cookies, drops of cooking oil were placed around varying parts of the cookie bottom before being placed on the cookie sheet. The results (shown in Figure 8-3) were not very promising. The cookie bottoms did not show significant variation across any of the test cases. This could be due to the specific recipe of the cookie being used, or simply because there was not enough variation in heat transfer properties between the control group bottoms and test group bottoms.

An alternative technique, suggested by Janice Wang and Dan Novy was to use egg wash. Egg wash is typically used in baking to provide a nice browned color to the top of various items. Such a technique could instead be used to apply egg wash through a sort of stencil, resulting in a deterministic pattern of darkened areas. These dark areas could then be used as the primitive in encoding any baked good.

To demonstrate the idea, mechanical means were used to create a textured pattern on the bottom of a baked cookie. The cookie and resultant mobile interface are shown

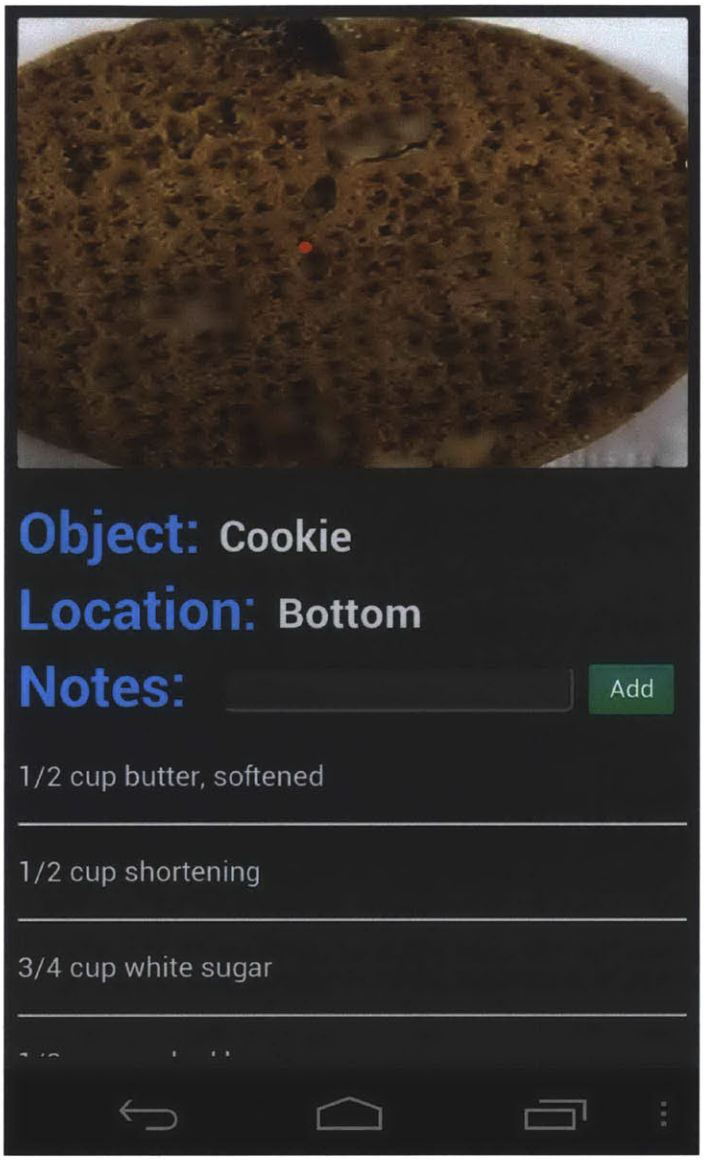


Figure 8-4: A screenshot of scanning a decodable cookie pattern.

in Figure 8-4.

While cookies and baked goods represent one possible domain of encoded food, there still remain many techniques that have yet to be explored. Grilling immediately comes to mind, as items such as steak branders have been used to sear a mark onto a steak before serving. Similar techniques could be used to grill food such that it is inherently seared with a pattern telling you about the food you are about to eat. More interestingly, perhaps, would be a technique where the darkness of the searing

is proportional to the cook time. In this way a correlation could be made between how well done a piece of meat is and the resultant encoding. Scanning the encoding could tell patrons if their steak was rare, well-done, etc.

## 8.3 Beyond

While a couple more instances of how a self-descriptive universe could be fabricated are discussed in this chapter, there are many more that will unfortunately go without noting. As fabrication technologies become more accurate and our mobile devices become more powerful, the opportunities and abilities of a self-descriptive world will only grow. The key in thinking about any self-descriptive object is to understand how it can be fabricated and what aspects of that fabrication process can be altered. Furthermore, it is useful to consider scenarios where the encoding has a secondary effect that leverages the physical nature of encoding. As an example, encoding the bottom of shoes by creating deterministic textured rubber patterns could provide the functionality of making the shoe self-descriptive in a way that it reveals what it is made out of, who wears it, etc, but it introduces a secondary effect in that it leaves tracks. The textured encoding, as the user walks through rain, mud, dirt, or other conditions will leave behind a texture pattern that could similarly be decoded. I believe this secondary nature is very powerful and in certain cases could provide be some wonderfully surprising and practical applications.

# Chapter 9

# Discussion and Future Work

## 9.1 Presented Opportunities

The system outlined in this thesis tries to solve the challenge of imbuing our passive physical object with information that makes them self-descriptive. That is, through the use of physical texture codes, systems are built to link object location IDs with collections of annotations about how the object works, where it was made, how it can be fixed, etc. While this thesis represents a first look at such systems, it attempts to open the doors to larger ideas and opportunities. That said, there do exist a handful of limitations and remaining challenges that are worth discussing.

## 9.2 Remaining Challenges

There exist some challenges that, while not necessarily important to the research aspect of such a domain, are important to the scalability and mass adoption of such ideas. For example, the system as presented is limited to working on Android devices only. No implementation for other operating systems or mobile devices is provided. While there are no foreseeable technical limits that would restrict such development, it would require a large amount of time to make user-friendly applications that exist across all platforms. Furthermore, there is an element of manual labor to the work that would not be ideal for a scaled system. For example, the STL models that are

generated and displayed in the browser interface are manually selected and labeled. There is no large backend system for handling the addition of new CAD models. It is foreseeable that one workable implementation would be to allow manufacturers and developers to upload models into a networked system directly, but such an interface does not exist. Without such an infrastructure, the system as a whole is limited to the domain of a prototype demo. Furthermore, and potentially more importantly, the process for encoding an object is quite manual and requires a custom approach for each new item.

### 9.2.1 Integration With Design Tools

One of the challenges of created 3D CAD models with an encoded surface is the manual process of generating a code, unwrapping the model, performing mesh subdivision, and then applying a displacement map. This process must be tweaked and performed in a custom fashion for each new object that is to be encoded. Such a process is not trivial and exact metrics for when a displacement map and resultant surface encoding is 'good enough' are not established. It would be important in future work to have this streamlined. The ideal scenario, in my mind, is one where in the export step of any CAD software there exists the option to 'apply surface encoding'. Checking this option and continuing to export would result in the desired CAD model with the appropriately textured encoded surface features. The algorithms for applying this generally are the topic of entire thesis on its own, but its value is understandable. Simple parameters could be included as advanced options, such as the ability to set the minimum resolution of the pending fabrication process or the bit density that is desired. A one-click approach such as this would certainly be challenging, but would likely lead to a much greater adoption of the idea of a self-descriptive universe.

### 9.2.2 Scalable Pattern Recognition

Another area that could be improved comes from the decoding process. To decode 3D objects, pattern recognition algorithms are used. These algorithms work by defining

a set of keypoints across the given image and checking to see if any known images in a database match that particular arrangement of key points. Key points are generated at locations of high contrast or sharp edges. While this technique works very well in the presented system, given its prototype scale, the challenge becomes a slightly different one if we are to work towards a scaled system with millions or more known codes. Fortunately, the growing speed and memory of cloud systems presents one possible future where large scaled databases would work, but whether it would be fast enough to provide the right experience is yet to be seen. Furthermore, relying on a cloud system would require that network connectivity is established, limiting the potential applications.

Moving away from pattern recognition to a system of raw data decoding is another option. While this option would be more challenging to implement, and perhaps more prone to errors and failures, it may be possible with advancing image processing techniques and mobile device hardware to provide a useable experience.

### 9.2.3 Characterizing the Robustness of Tags

A topic that is not addressed, but would be of high importance for commercial applications, would be to assess the robustness of the encoded textures after normal wear and tear. Longitudinal studies that measure the bit-error rate and read accuracy could be made at multi-month intervals on items that are handled and used regularly. This would provide insight into the robustness of specific encoding schemes given typical usage patterns. These measurements would surely depend on the material of the object. Silicone molds would perform differently over time as compared to a plastic 3D printed object or metal cast object.

## 9.3 Encoded Surfaces as a Means to Evolve

Looking to the future of how such a system may grow in use cases and application domains, it is exciting to consider possible advancements of the idea assuming the technical implementations are established. Returning to the analogy present in the

first line of Chapter 1 where an argument for creating a 'DNA' for physical objects is presented, it is interesting to carry that metaphor further. Initially the argument is that like biological DNA, which entirely describes a living being, there is value in creating a passive DNA encoded into the physical structure of our objects that can entirely describe the CAD file of that given object. However, one of the most important features of biological DNA is that it can mutate and, through these mutations, evolve. So let's take that metaphor to the similar domain.

Perhaps the surface encoding could also be used as a type of memory that describes the environmental conditions experienced by the object. A model that is weak at some point could show a deformation or change in the original surface encoding. This change could be decoded and analyzed to understand that future iterations of the object should be made stronger at that critical location. In this way, we allow objects to evolve as a function of the environmental conditions that they are subject to. This idea becomes even more powerful if coupled with the previous notion of connecting mass-produced items based on their shared surface encoding: a product line could evolve from generation to generation based on the reading of deformed or altered surface encodings. I think it would be an interesting experiment to create such an object, and every week 3D print whatever iteration is deemed current by the surface encoding that had been altered during that week. Slowly, over time you would begin to get objects that look entirely different than what you started with. Perhaps future soda bottles will have a lineage as rich, diverse, and unexpected as those that lead to present day life forms.

# Chapter 10

# Conclusion

This thesis presents and outlines a system for encoding physical passive objects with deterministic surface features that contain identifying information about that object. The goal of such work is to take steps towards a self-descriptive universe in which all objects contain within their physical structure hooks to information about how they can be used, how they can be fixed, what they're used for, who uses them, etc. By exploring modern manufacturing processes, several techniques for creating these deterministic textures are presented. Of high importance is the advancement of 3D printing technologies. By leveraging the rapid prototyping capabilities such machines offer, this thesis looks at how personalized objects and draft models may be encoded with data that allows annotations, ideas, and notes to be associated with physical points across that object.

A mobile application is provided that provides decoding capabilities and an interface for reading past annotations or writing new notes related to scanned texture codes. A browser interface is also described and built that allows for multiple remote parties to be annotating, exploring, and viewing the CAD file and associated data of physical objects.

The decoding mechanism relies on powerful pattern recognition algorithms that allow this system to extend beyond 3D printed objects and into other things that have within them a deterministic encoded pattern. This sets the ground work for scaling this idea to a wide set of domains that encompass many of the physical passive objects

that surround our everyday environments.

I hope that future researchers can use this work to explore their own set of ideas about what a self-descriptive universe would look like and the exciting new applications and opportunities it would create.

# Appendix A

# Source Code

## A.1 Primitive Generation

```
//Height and Width of window
int width = 400;
int height = 400;
//Define noise overlay image
PImage noiseImg;

void setup(){
 size(width,height);
 //background = black
 background(255);
 noiseImg = loadImage("noise.png");

//Function calls for reference
// drawTriangles(80,40);
// drawCircles(80,40);
// drawSquare(80,40);
// drawHorz(80,40);
// drawVert(80,40);

 for(int i=0; i< 256; i++){
  drawCircles(80,40); // Draw array of primitives
  tint(255, i); //Define image opacity (opacity decreases as i goes to 255)
  image(noiseImg, 0, 0); //Load noise image (will use previous opacity)
  String fileName = "images/vert_"+i+".png"; //Generate filename of image
      to save
  save(fileName); //Save primitive array as file
 }
```

```
}

void drawTriangles(int spacing, int length){
  for(int xdim = spacing; xdim<width; xdim = xdim+spacing){
     for(int ydim = spacing; ydim<height; ydim = ydim+spacing){
       fill(0);
       triangle(xdim, ydim+length/2, xdim+length/2, ydim-length/2,
           xdim-length/2, ydim-length/2);
     }
  }
}

void drawCircles(int spacing, int diameter){
    for(int xdim = spacing; xdim<width; xdim = xdim+spacing){
     for(int ydim = spacing; ydim<height; ydim = ydim+spacing){
       fill(0);
       stroke(0);
       ellipse(xdim, ydim, diameter, diameter);
     }
  }
}

void drawSquare(int spacing, int length){
    for(int xdim = spacing; xdim<width; xdim = xdim+spacing){
     for(int ydim = spacing; ydim<height; ydim = ydim+spacing){
       fill(0);
       rect(xdim-length/2, ydim-length/2, length, length);
     }
  }
}

void drawVert(int spacing, int length){
    for(int xdim = spacing; xdim<width; xdim = xdim+spacing){
     for(int ydim = spacing; ydim<height; ydim = ydim+spacing){
       fill(0);
       rect(xdim-length/2, ydim-length/2, length/3, length);
     }
  }
}

void drawHorz(int spacing, int length){
    for(int xdim = spacing; xdim<width; xdim = xdim+spacing){
     for(int ydim = spacing; ydim<height; ydim = ydim+spacing){
       fill(0);
       rect(xdim-length/2, ydim-length/2, length, length/3);
     }
```

```
  }
}
```

## A.2 Primitive Detection

```
#include "cinder/app/AppBasic.h"
#include "cinder/gl/gl.h"
#include "cinder/gl/Texture.h"
#include "CinderOpenCV.h"
#include "cinder/Rand.h"
#include "cinder/params/Params.h"


using namespace ci;
using namespace ci::app;
using namespace std;

class openCVTestApp : public AppBasic {
  public:
        void setup();
        void update();
        void draw();
    void prepareSettings( Settings* settings );

    params::InterfaceGl mParams;
    float mThreshold, mBlobMin, mBlobMax;
    gl::Texture mDepthTexture, mCvTexture;
    Surface mDepthSurface;
    int imageCounter;

    int numofCircles;
    int numofTri;
    int numofVert;
    int numofHorz;
    int numofSquare;
};

void openCVTestApp::prepareSettings( Settings* settings )
{
    settings->setWindowSize( 640, 720 );
}

void openCVTestApp::setup()
{
    mThreshold = 200.0f;
```

```
    mBlobMin = 5.0f;
    mBlobMax = 100.0f;

    mParams = params::InterfaceGl( "Hand Tracking", Vec2i( 10, 10 ) );
    mParams.addParam( "Threshold", &mThreshold, "min=0.0 max=255.0
        step=1.0 keyIncr=s keyDecr=w" );
    mParams.addParam( "Blob Minimum Radius", &mBlobMin, "min=1.0 max=200.0
        step=1.0 keyIncr=e keyDecr=d" );
    mParams.addParam( "Blob Maximum Radius", &mBlobMax, "min=1.0 max=200.0
        step=1.0 keyIncr=r keyDecr=f" );

    imageCounter=-1;
}

void openCVTestApp::update()
{
    imageCounter++;
    numofCircles = 0;
    numofTri = 0;
    numofVert =0;
    numofHorz =0;
    numofSquare=0;
    if(imageCounter<255){
//    if(1){ //Uncomment if you want to run just a single image.
        //Generate File name to pass to loadResource
        std::string label = "circ_";
        std::string post = ".png";
        std::string result = label +
            boost::lexical_cast<std::string>(imageCounter) + post;
        ci::Surface8u surface( loadImage( loadResource( result ) ) );
//        ci::Surface8u surface( loadImage( loadResource( "circ.png" ) ) );
    //Uncomment if you want to run just a single image

        gl::Texture depthImage = gl::Texture(surface);
        // make a texture to display
        mDepthTexture = depthImage;
        // make a surface for opencv
        mDepthSurface = surface;

        if(mDepthSurface){

            cv::Mat input_inv( toOcv( Channel8u( mDepthSurface ) ) ),
                blurred, thresholded, thresholded2, output;
            cv::Mat input;
            cv::bitwise_not(input_inv, input);

            cv::blur(input, blurred, cv::Size(10,10));
```

```
// make two thresholded images one to display and one
// to pass to find contours since its process alters the image
cv::threshold( blurred, thresholded, mThreshold, 255,
    CV_THRESH_BINARY);
cv::threshold( blurred, thresholded2, mThreshold, 255,
    CV_THRESH_BINARY);

// 2d vector to store the found contours
vector<vector<cv::Point> > contours;
// find em
cv::findContours(thresholded, contours, CV_RETR_EXTERNAL,
    CV_CHAIN_APPROX_SIMPLE);

// convert theshold image to color for output
// so we can draw blobs on it
cv::cvtColor( thresholded2, output, CV_GRAY2RGB );
int i = 0;
// loop the stored contours
for (vector<vector<cv::Point> >::iterator it=contours.begin() ;
    it < contours.end(); it++ ){
    i++;
    // center and radius for current blob
    cv::Point2f center;
    float radius;
    // convert the contour point to a matrix
    vector<cv::Point> pts = *it;
    cv::Mat pointsMatrix = cv::Mat(pts);

    // pass to min enclosing circle to make the blob
    cv::minEnclosingCircle(pointsMatrix, center, radius);

    int isTriangle = 0;
    int isCircle = 0;
    int isNothing = 1;

    vector<cv::Point> approx;
    cv::approxPolyDP(pts,approx,radius*.2,TRUE);
    if (approx.size()==3){
        isCircle = 0;
        isTriangle = 1;
        isNothing = 0;
        numofTri++;
    }

    if (approx.size()==4){
        isCircle = 1;
```

```
                isTriangle = 1;
                isNothing = 0;
                numofSquare++;
            }

            // calculate mean distance between provided edge points
            // and estimated circle's edge
            float meanDistance = 0;
            int ii;
            for (ii = 0; ii < pointsMatrix.rows; ii++ )
            {
                meanDistance += abs((float)
                    sqrt(pow(center.x-pts[ii].x,2)+pow(center.y-pts[ii].y,2))
                    - radius );
            }
            meanDistance /= ii;

            if(meanDistance<2 & approx.size() != 4){
                isCircle = 1;
                isTriangle = 0;
                isNothing = 0;
                numofCircles++;
            }

            cv::Scalar blobColor = CV_RGB( 255*isCircle, 255*isTriangle,
                255*isNothing);

            if (radius > mBlobMin && radius < mBlobMax) {
                // draw the blob if it's in range
                cv::circle(output, center, radius, blobColor,3);
            }
        }
        cout<<numofSquare<<endl;
        mCvTexture = gl::Texture( fromOcv( output ) );
    }
  }
}

void openCVTestApp::draw()
{
      gl::clear( Color( 0.5f, 0.5f, 0.5f ) );

  gl::disableDepthWrite();
  gl::disableDepthRead();

  glPushMatrix();
  gl::scale(Vec3f(-0.5, 0.5, 1));
```

```
    if( mDepthTexture )
        gl::draw( mDepthTexture,Vec2i( -640, 420 ));
    if ( mCvTexture ){
        gl::draw( mCvTexture,Vec2i( -1280, 420 ));
    }
    glPopMatrix();

    gl::enableDepthWrite();
    gl::enableDepthRead();


    params::InterfaceGl::draw();
    //Draw the FPS on screen
    gl::drawString(std::string("FPS: "+
        boost::lexical_cast<std::string>(getAverageFps())), Vec2f( 400.0f,
        20.0f ), Color::white() );


}

CINDER_APP_BASIC( openCVTestApp, RendererGl )
```

## A.3 Coin Generation

```
int width = 9920;
int height = 9920;

//int width = 2000;
//int height = 2000;


int numWidth = 169;
int numHeight= 257;
PImage numtrans;

int[] xpos_bits1 = {179,215,251,161,269,161,269,179};
int[] ypos_bits1 = {202,202,202,238,238,274,274,310};

int[] xpos_bits2 = {233,269,197,269,161,269,197,233};
int[] ypos_bits2 = {310,310,328,346,382,382,400,400};

void setup(){
 size(width,height);
 background(255);
 numtrans = loadImage("99trans.png");
```

```
// Coins froms 0 to 255
 for(int ii = 0; ii < 16; ii++){
   for(int jj = 0; jj < 16; jj++){
     generateCoin(ii*16+jj,620*jj,620*ii);
   }
 }
 save("coinsFinal.png"); //Save primitive array as file
}

void drawCircles(int spacing, int diameter, int xstart, int ystart){
    for(int xdim = spacing; xdim<numWidth; xdim = xdim+spacing){
     for(int ydim = spacing; ydim<numHeight; ydim = ydim+spacing){
       fill(0);
       stroke(0);
       ellipse(xstart+xdim, ystart+ydim, diameter, diameter);
//       textSize(7);
//       fill(0, 102, 153);
//       text(ystart+ydim,xstart+xdim, ystart+ydim);
     }
  }
}

void generateCoin(int coinNum, int xstartCoin,int ystartCoin){
   String binStr = binary(coinNum,8);

   drawCircles(18,6, xstartCoin+125,ystartCoin+166); // Draw array of
      primitives
   drawCircles(18,6, xstartCoin+304,ystartCoin+166); // Draw array of
      primitives
   image(numtrans, xstartCoin+125, ystartCoin+166);
   image(numtrans, xstartCoin+304, ystartCoin+166);

   for(int i=0;i<8;i++){
    if(binStr.charAt(i) == '0'){ //if binary 0, create white circle to
       hide the bit
       fill(255);
       strokeWeight(2);
       stroke(255);
       ellipse(xstartCoin+xpos_bits1[i], ystartCoin+ypos_bits1[i], 6, 6);
          //first nine - top half
       ellipse(xstartCoin+xpos_bits2[i], ystartCoin+ypos_bits2[i], 6, 6);
          //first nine - bottom half

       ellipse(179+xstartCoin+xpos_bits1[i], ystartCoin+ypos_bits1[i], 6,
          6); //second nine - top half
```

```
        ellipse(179+xstartCoin+xpos_bits2[i], ystartCoin+ypos_bits2[i], 6,
            6); //second nine - bottom half
    }
   }

       textSize(50);
       fill(0);
       text("Jan 11, 2013",xstartCoin+146, ystartCoin+125);

       textSize(60);
       fill(0);
       text("Fridays",xstartCoin+200, ystartCoin+525);

       stroke(0);
       line(xstartCoin+143,ystartCoin+454,xstartCoin+466,ystartCoin+454);



   //Comment out for final version for lasercutter
   stroke(0);
   strokeWeight(1);
   noFill();
//   ellipse(xstartCoin+300,ystartCoin+300,600,600);
   //
}
```

# Bibliography


[1] M.R. Ackermann, P.A. Cahill, T.J. Drummond, and J.P. Wilcoxon. A brief examination of optical tagging technologies. *Sandia National Laboratories Albuquerque, New Mexico*, 87185, 2003.

[2] K.J. Brodie. Global positioning system tag system, July 30 2002. US Patent 6,427,121.

[3] Chee Kai Chua, Kah Fai Leong, and Chu Sing Lim. *Rapid prototyping: principles and applications*. World Scientific Publishing Company Incorporated, 2010.

[4] Klaus Cicha, Zhiquan Li, Klaus Stadlmann, Aleksandr Ovsianikov, Ruth Markut-Kohl, Robert Liska, and Jurgen Stampfl. Evaluation of 3d structures fabricated with two-photon-photopolymerization by using ftir spectroscopy. *Journal of Applied Physics*, 110(6):064911–064911, 2011.

[5] Patrick Dallaire, Daniel Emond, Philippe Giguere, and Brahim Chaib-Draa. Artificial tactile perception for surface identification using a triple axis accelerometer probe. In *Robotic and Sensors Environments (ROSE), 2011 IEEE International Symposium on*, pages 101–106. IEEE, 2011.

[6] Andrew Nicholas Dames. Multi-bit magnetic tag and its method of manufacture, April 16 1998. WO Patent WO/1998/015,853.

[7] K. Finkenzeller. *RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication*. Wiley, 2010.

[8] R.J. Gambino, A.G. Schrott, and R.J. Von Gutfeld. Magnetic tag using acoustic or magnetic interrogation, October 15 1996. US Patent 5,565,847.

[9] Neil Gershenfeld. *Fab: The coming revolution on your desktop–from personal computers to personal fabrication*. Basic Books, 2007.

[10] D.R. Griffin. Radioactive tagging of animals under natural conditions. *Ecology*, 33(3):329–335, 1952.

[11] F. Guimbretière. Paper augmented digital documents. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 51–60. ACM, 2003.

[12] F.A. Hansen. Ubiquitous annotation systems: technologies and challenges. In *Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 121–132. ACM, 2006.

[13] C. Harrison, R. Xiao, and S. Hudson. Acoustic barcodes: passive, durable and inexpensive notched identification tags. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 563–568. ACM, 2012.

[14] J.F. Heanue, M.C. Bashaw, L. Hesselink, et al. Volume holographic storage and retrieval of digital data. *Science (New York, NY)*, 265(5173):749, 1994.

[15] T. Higo, Y. Matsushita, N. Joshi, and K. Ikeuchi. A hand-held photometric stereo camera for 3-d modeling. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1234–1241. IEEE, 2009.

[16] B. Holat. Method of a making and applying a holographic identifier for garments, March 14 2000. US Patent 6,036,810.

[17] Visualant Incorporated. *Technology Overview*, 2012. http://www.visualant.net/overview.html.

[18] M.K. Johnson and E.H. Adelson. Retrographic sensing for the measurement of surface texture and shape. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1070–1077. IEEE, 2009.

[19] M.K. Johnson, F. Cole, A. Raj, E.H. Adelson, et al. Microgeometry capture using an elastomeric sensor. *ACM Transactions on Graphics (TOG)*, 30(4):46, 2011.

[20] Thomas Jung, Mark D Gross, and Ellen Yi-Luen Do. Annotating and sketching on 3d web models. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 95–102. ACM, 2002.

[21] Shunichi Kasahara, Valentin Heun, Austin S Lee, and Hiroshi Ishii. Second surface: multi-user spatial collaboration system based on augmented reality. In *SIGGRAPH Asia 2012 Emerging Technologies*, page 20. ACM, 2012.

[22] D. Merrill and P. Maes. Invisible media: Attention-sensitive informational augmentation for physical objects. In *Seventh International Conference on Ubiquitous Computing (UbiComp05)*, 2005.

[23] J.J. Mihalov, A. Der Marderosian, and J.C. Pierce. Dna identification of commercial ginseng samples. *Journal of agricultural and food chemistry*, 48(8):3744–3752, 2000.

[24] RB Mitson and TJ Storeton-West. A transponding acoustic fish tag. *Radio and Electronic Engineer*, 41(11):483–489, 1971.

[25] A. Mohan, G. Woo, S. Hiura, Q. Smithwick, and R. Raskar. Bokode: imperceptible visual tags for camera based interaction from a distance. *ACM Transactions on Graphics (TOG)*, 28(3):98, 2009.

[26] Catarina Mota. The rise of personal fabrication. In *Proceedings of the 8th ACM conference on Creativity and cognition*, pages 279–288. ACM, 2011.

[27] Alexander Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.

[28] J.M. Prober, G.L. Trainor, R.J. Dam, F.W. Hobbs, C.W. Robertson, R.J. Zagursky, A.J. Cocuzza, M.A. Jensen, K. Baumeister, et al. A system for rapid dna sequencing with fluorescent chain-terminating dideoxynucleotides. *Science (New York, NY)*, 238(4825):336, 1987.

[29] Bradley J Rhodes. The wearable remembrance agent: A system for augmented memory. *Personal and Ubiquitous Computing*, 1(4):218–224, 1997.

[30] Bradley James Rhodes and Pattie Maes. Just-in-time information retrieval agents. *IBM Systems Journal*, 39(3.4):685–704, 2000.

[31] J. Rouillard. Contextual qr codes. In *Computing in the Global Information Technology, 2008. ICCGI'08. The Third International Multi-Conference on*, pages 50–55. IEEE, 2008.

[32] Utpal Roy and Yaoxian Xu. 3-d object decomposition with extended octree model and its application in geometric simulation of nc machining. *Robotics and Computer-Integrated Manufacturing*, 14(4):317 – 327, 1998.

[33] Sculpteo. *Sculpteo — Your 3D design turns into reality with 3D printing*, 2013. http://www.sculpteo.com/en/.

[34] Inc. Shapeways. *Shapeways - Make, buy, and sell products with 3D Printing.*, 2013. http://www.shapeways.com/.

[35] Jivko Sinapov, Vladimir Sukhoy, Ritika Sahai, and Alexander Stoytchev. Vibrotactile recognition and categorization of surfaces by a humanoid robot. *Robotics, IEEE Transactions on*, 27(3):488–497, 2011.

[36] H. Song, F. Guimbretière, C. Hu, and H. Lipson. Modelcraft: capturing freehand annotations and edits on physical 3d models. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 13–22. ACM, 2006.

[37] Ivan E Sutherland. Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE design automation workshop*, pages 6–329. ACM, 1964.

[38] R. Want, K.P. Fishkin, A. Gujar, and B.L. Harrison. Bridging physical and virtual worlds with electronic tags. In *Conference on Human Factors in Computing Systems: Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, volume 15, pages 370–377, 1999.

[39] R. Want and D.M. Russell. Ubiquitous electronic tagging. *Distributed Systems Online, IEEE. http://www. computer. org/dsonline/articles/ds2wan. htm*, 2000.

[40] J. Werb, K. Underriner, and M. Long. Asset and personnel tagging system utilizing gps, March 2 2004. US Patent 6,700,533.

[41] C. Wu. Tagged out. *Science News, in Taggants: Barcodes for Bombs*, 1996.

[42] Li Yang, Amin Rida, Rushi Vyas, and Manos M Tentzeris. Rfid tag and rf structures on a paper substrate using inkjet-printing technology. *Microwave Theory and Techniques, IEEE Transactions on*, 55(12):2894–2901, 2007.

[43] Ibrahim Zeid. *CAD/CAM theory and practice*. McGraw-Hill Higher Education, 1991.

[44] Linlin Zheng, Saul Rodriguez, Lu Zhang, Botao Shao, and Li-Rong Zheng. Design and implementation of a fully reconfigurable chipless rfid tag using inkjet printing technology. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 1524–1527. IEEE, 2008.