Thesis Proposal:
Interaction Techniques for Sketch-based Design for Rapid Prototyping using Laser Cutters

Gabe Johnson
School of Architecture
Carnegie Mellon University

May 18, 2011

Committee

Mark D. Gross (Chair) — School of Architecture — CMU

Ellen Yi-Luen Do — College of Architecture & College of Computing — Georgia Tech

Jason I. Hong — Human Computer Interaction Institute — CMU

Contents

Abstract

Rapid prototyping machines such as laser cutters are becoming more common. Designers commonly make quick sketches to aid thinking about the problem and possible solutions. Sketching is a skill that even untrained people can perform to some useful degree. Unlike sketching, modeling software needed to design things that can be made with laser cutters poses a significant learning challenge for inexpert users and can be hard to use even for those with experience. Freehand sketches often capture critical elements of the design, such as object shapes, curvature qualities of boundaries, and numeric parameters like lengths—all of which must be communicated to a computer in order to produce on a laser cutter.

I propose to create a modeling system that allows for design of objects via iterative freehand sketching. Sketch-based interaction is a promising design paradigm, but unlike traditional applications there are no standard interaction techniques for sketch-based applications. Standardizing interaction techniques allow people to use other applications more easily. The proposed system will explore sketch-based interaction techniques for moderately complex projects involving the design of objects for production via laser cutter. Additionally, I will test the utility of this system against existing tools commonly used for designing objects for laser cutters.

1 Problem Statement

In recent years, computational support for sketching has focused mostly on the early phases of design. This is for good reason: paper and pencil sketching remain central to design practice despite the ubiquity of powerful computer-based design tools. Sketching allows people to quickly jot down ideas or exchange thoughts with others; they provide a medium through which designers think about problems and potential solutions. Freehand drawing is an activity that nearly all designers—professional and avocational alike—readily employ.

Ideas that begin as rough sketches may eventually be prototyped with rapid fabrication machinery such as 3D printers, CNC routers, or laser cutters. Over time, this sort of machinery will become more affordable, utilize a wider range of materials, and produce higher-quality output. Rapid fabrication machines offer the potential to enable people to take active control of the design of their world, rather than being passive consumers of products somebody else has made.

However, a gap remains between the technology used to design and the technology used to prototype physical items. Today, a typical design process might involve several paper sketches, followed by a session with a computer modeling system such as SketchUp, SolidWorks, or Rhino. When the designer is satisfied with the CAD model, the rapid fabrication machinery is put to work. The designer can then evaluate the output and decide what to do next: keep the design as it is, change the CAD model, or "go back to the drawing board" and sketch new ideas. It is common for people to physically sketch while making revisions or redesigning, because it is often easier and faster to sketch than working with existing CAD models. This is especially the case with the users targeted by this system: avocational designers who are not necessarily proficient with professional tools.

The software proposed here is in a category that does not yet exist outside of research labs. Consequently, it is important to build such software to explore the possibilities. I argue that it is beneficial to look beyond current user needs and invent tools for the future.

Kirsh and Maglio [30] distinguish between *pragmatic* and *epistemic* actions. Structured

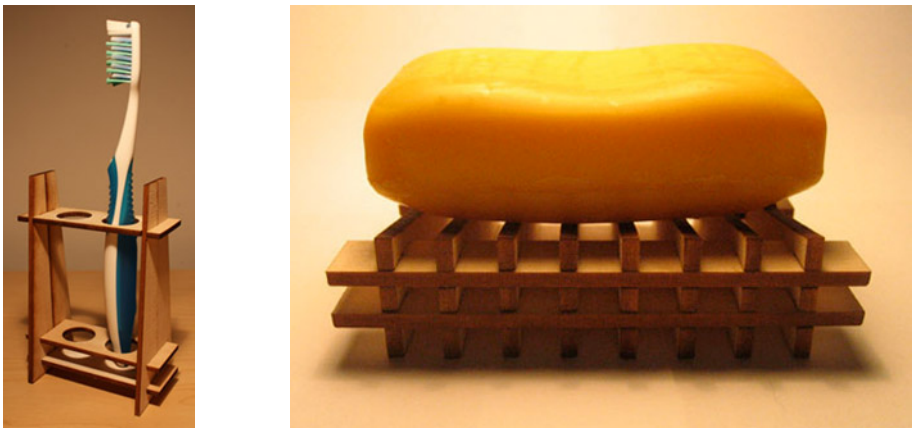
modeling tools enable users to manipulate domain elements: boundaries, vertices, textures, layers, light sources, and so on. These pragmatic actions transform the model. When people sketch, they frequently make marks that aid thinking but are not intended to be part of the model. These epistemic actions transform the designer's state of mind that potentially make it easier to think about how to proceed. Computer modeling tools tend to be very good at supporting structured, pragmatic actions. Unfortunately, computer tools are poor at supporting sketchy, epistemic actions.

Goel's studies found that designers using structured computer tools came up with fewer and less creative ideas than others who sketched [14]. Although this study is by now close to twenty years old, the nature of structured design tools has not changed (and sketching certainly has not). It is faster and easier to sketch an idea than it is to produce it in a modeling tool. This rapid idea generation leads to more ideas, giving a larger pool from which to draw new ideas. Ultimately sketching promotes better design by allowing designers to explore more ideas and variations within those ideas [14, 55].

While sketching is regarded as a highly effective technique for exploring design ideas, structured editors are seen as more appropriate for making incremental revisions to a single idea. Consider the following quote from [43]:

> *"The beginning of each step I'll do on paper. As soon as I feel like I'm going to be starting any design revisions, then I'll move to [an electronic tool]... because it's easier to make changes to these things."*

This proposal describes an approach that brings sketching and computer modeling activities together in the same tool. This tool belongs to area of sketch-based interaction and modeling (SBIM). These applications are sometimes called calligraphic systems or Computer Aided Sketching (CAS). The proposed tool lets users sketch as roughly or precisely as they like, iteratively and interactively making models that can be manufactured using rapid fabrication machines. Specifically, the system will support users to employ calligraphic interaction to model objects for production on laser cutters. Such objects are composed of



(a) (b)

Figure 1: Household objects made with a laser cutter.

flat, rigid parts. Figure 1 shows example household objects that could be designed using the proposed system and produced with a laser cutter.

This project's first motivation is to empower those who are not professional designers to create objects with rapid fabrication machines. It can not be assumed that these users have been taught to draw as an architect would have been, nor can it be assumed they have invested a great deal of time to learn the often complicated modeling software that was developed for professional use. Therefore, the project must enable people to design without presenting an intimidating learning curve.

The second motivation is to explore the space of sketch-based interaction as it applies to modeling physical objects. A success criterion is that the user should never need (or want) to set down their stylus in favor of keyboard or mouse input.

1.1 Conversational Sketch-based Interaction

The premise of interactive systems is that there is an ongoing "conversation" between the user and computer. In sketch-based systems, users "speak" with a stylus; computers "talk back" with graphics. Just as humans have social norms for engaging in conversation that allow us to talk to people we have not met before, there should be interaction norms

for working with sketch-based systems so our experience can be portable among various sketch-based applications.

Olsen [45] argues the dominant paradigm of "one display, one keyboard, one mouse" hinders progress in interactive systems that use alternate hardware configurations. This is because it is expedient to replicate mouse/keyboard interaction on new devices, even if the hardware is different. New systems architectures are needed to more appropriately take advantage of new hardware configurations. There is little agreement on interaction techniques for sketch-based systems, where hardware is a pen-sensitive display and stylus. In recent years, researchers have begun to give more attention to interaction aspects of sketch recognition-based user interfaces, but the extent of this work remains limited. This topic is discussed further in Section 5.2.

For each of the several technical challenges in sketch-based systems research there is a human component. Segmentation and recognition are concerned with perceptual and semantic interpretation, which is inevitably wrong on occasion. When does the system attempt to recognize input? How detailed must the recognition be in order to match the user's intentions? What salient aspects are to be recognized? How does the system tell the user what has been recognized? When should this occur? How can the user work with the system to recover from mistakes?

The answer to these questions begins with "it depends on what the user is trying to achieve." In the proposed system, ideas are gradually refined from sketches into manufacturable models. The user's goals are different early in this process compared with latter stages. Early in this process the user's goal is to simply record ideas, possibly to share with other people. The early phase is usually about *big ideas*. Later, the user's goals might be to add specific details such as lengths, angles, and how different parts interact—this phase is about *details*. And the iterative, dynamic nature of many projects might see designers walking back their detailed work to try different big ideas. Ideally, the system should support designers to work at either level at any time.

1.2 Proposal Organization

The remainder of the proposal is organized as follows. First I give a brief overview of the thesis goals, scope, and methods. Next I give details on the design domain my system will support. To give a better sense of the system I intend to build, I then present a motivating scenario of how the system might be used, and relates many of the sketch-based interaction techniques. Previous related work on sketch recognition, sketch interaction, and rapid fabrication is then discussed. Next I detail my own efforts in sketching and design tools for fast fabrication. The proposal ends with the thesis contributions, a discussion of how the system will be evaluated, and a time line for project completion.

2 Thesis Summary

This section summarizes the purpose, scope, and methods of my dissertation.

2.1 Thesis Statement

This thesis holds that iterative sketch-recognition based interaction can be a useful and usable paradigm for design in the domain of laser-cut items.

2.2 Research Scope

The tool under consideration here supports inexpert designers in the fairly narrow domain of design for 2D laser-cutter fabrication. This domain is chosen because laser-cut parts typically have some aspects that require precise measurements.

The research scope of this project is to develop calligraphic techniques to facilitate the design of these 2D parts. I will evaluate these techniques in context of the laser-cutting fabrication design domain. This domain is chosen for several reasons. 2D environments are easier to develop than a 3D tool. I have experience making a laser cutter design tool (which is described later). Last, laser cutters are relatively quick when compared to 3D

printer fabrication (on the order of minutes compared with hours), so this domain will allow me to iterate more rapidly.

2.3 Methods

This dissertation will center on iteratively creating a calligraphic design tool. Development will be based on two sources:

1. Prior work: This includes the work of others (largely summarized in [27]) and my own past work.
2. Iterative evaluation: I will regularly test the system with CMU students and other volunteers.

2.4 Objectives Beyond This Thesis

The research involved in this thesis is part of a larger interest in helping to bring sketch-based interaction to people in areas beyond the laser cutter domain presented here. Sketching is done by designers of many kinds, from architecture to industrial design, and from software to civil engineering. Nor is freehand drawing only a tool for 'design': diagrams and mathematical graphs are commonly seen on whiteboards in classrooms and corporate meeting rooms. It is hoped that the current work will improve the body of knowledge related to sketch-based interactive systems.

3 New Makers, New Tools

There is a growing community of self-described *makers* who design and build many kinds of physical things [13]. Some are electronic or robotic gizmos, while others are composed of traditional material but designed or made with modern technology. The "new makers" [19] include mechanical engineers, industrial designers, artists, computer programmers, architects, and many others. It is common for these people to be relatively untrained in aspects

of their craft, such as using modeling software, designing or assembling electronics, or making visually appealing physical constructions. It is rare to find any individual person that is an expert in all of these topics, but it is common that a person knows something about a lot of them.

It is possible that we are beginning to see a shift from an economy based on mass-production (in factories) to one that includes mass-customization (in homes and community centers). Rapid fabrication machines continue to decline in price while improving in quality. It is conceivable that in the not-so-distant future, small manufacturing machines will be as common as desktop printers [60]. A new sector of small businesses use rapid fabrication to cater to the needs of hobbyist designers as well as people that need highly customized goods (e.g. prosthetic limbs) [42].

The surge of home manufacturing is made possible by a number of related factors. First, there is a trend away from a culture of consumption towards a culture of production. This is most visibly reflected in the democratization of modern media where the users become the dominant producers of content (e.g. social networking systems like Facebook or the game Minecraft). People are once again beginning to participate in the design of the world around them.

Second, there is a developing support structure for the new makers, including physical shared space, Internet resources, magazines, and conferences. *Hacker spaces* are physical locations where people can meet, share ideas, and work on projects. Hacker spaces commonly have various tools and supplies available to members. Online stores such as Sparkfun and model sharing sites like Thingiverse and 3D Warehouse give hackers easy access to supplies. There are several Internet-based companies (e.g. Ponoko) that offer fabrication services. O'Reilly's *Make Magazine* is a popular periodical catering to this community. O'Reilly also puts on several *Maker Faire* gatherings every year where physical hackers can meet.

Last, physical hacking is made accessible by the increasing availability and affordability of rapid fabrication machines like 3D printers, CNC mills, and laser cutters. This work

focuses on projects that use laser cutters. Figure 2 shows prices for a comparable 25-Watt, 16”x12” laser cutter model from Universal Laser Systems (these values were found on hobbyist web forums). While these data may not be exact, they do show the price of desktop laser cutting machines has been cut by almost half in the past ten years. While still out of reach for most people to afford, they are becoming inexpensive enough for schools and hacker spaces to own.

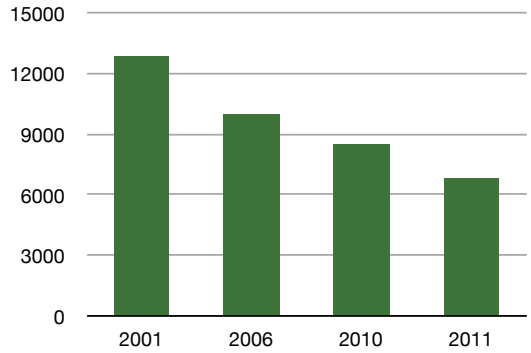


Figure 2: Declining prices of Universal Laser Systems 25-Watt 16x12 inch laser cutter (US Dollars).

3.1 Current Design Practices

Laser cutters (and other rapid fabrication machines) are a relatively new addition to the design studio. Further, many of the people using laser cutters are avocational designers that lack formal design education.

I am currently conducting interviews and observations to better understand how people think about, model, and make things using laser cutters.

3.2 Laser Cutters and Fabricated Output

A laser cutter can be thought of as a very fast, strong, and precise automated razor blade. They cut through flat material (paper, wood, plastic, metal, etc.) from directly above.

Some items can be made entirely with a laser cutter. These projects do not involve

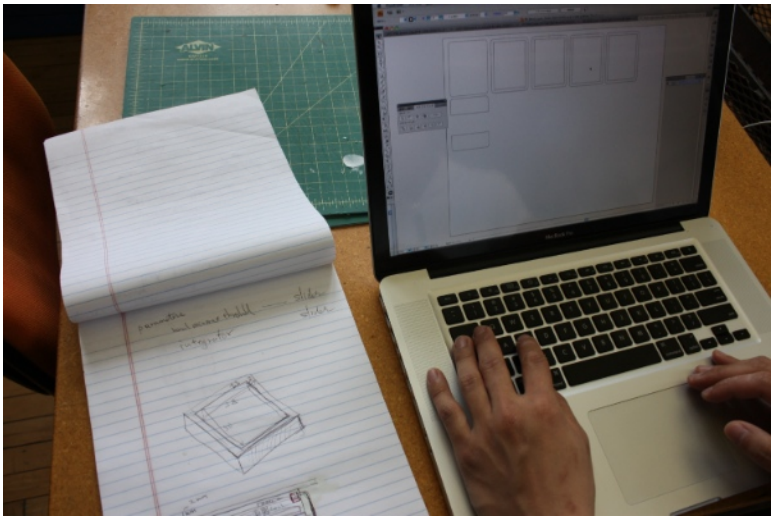


Figure 3: A common part of designing for laser cutters: translating a hand-made sketch to a computer modeling tool. The sketch includes a 3D depiction of the result, with important dimension values written down.

other machines or material, aside from the occasional screw or glue. Other projects might be made partly with a laser cutter, and partly with other machines. Sometimes an artifact is designed with to allow it to interact with another object that already exists. This places additional constraints on the design because it must conform to existing measurements. The car-mounted smart phone holder in Figure 4 is one such example. If we are given another similar device with slightly different dimensions, we might simply change some parameters and cut another holder.

3.3 Current Tools, Current Problems

Rapid fabrication hardware requires software. Currently, modeling tools for rapid fabrication are almost exclusively designed for professional, expert users. If rapid fabrication is to become common, software modeling tools must be made accessible by ordinary users [35].

Today, designers can choose among several modeling tools for laser cutter projects. As a minimum requirement, the tool must be able to produce a vector graphics file, where colors indicate which action the laser takes (e.g. cut through material or etch into it). Adobe Illustrator is the most commonly used package. Others include Rhino, InkScape,

Figure 4: A laser-cut smart phone holder designed to fit around a particular device and mounted on the dash of a car (by Davide Prato).

AutoCAD, and SolidWorks.

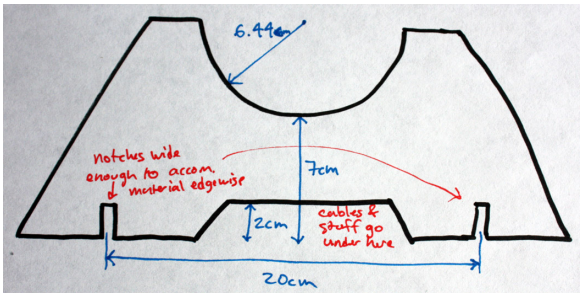
Illustrator is a general-purpose vector graphics tool. Specialized design tools like Rhino or SolidWorks are perhaps more appropriate for this kind of modeling, but many people are already familiar with Illustrator from other 2D vector graphics editors.

I interviewed designers to find out more about their work practices and better understand how they use their tools[1]. They have substantially different backgrounds: all have training in some form of design, ranging from mechanical engineering to graphic design to architecture. I began by asking each person to describe how they work. Each designer was able to show me sketches or videos of their work. While there are subtle (and some substantial) differences in their processes, each followed the following pattern.

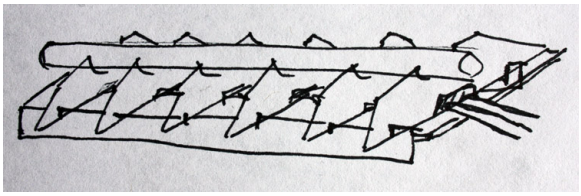
They begin by thinking about a problem and making drawings by hand. Some sketches are made to think about how to frame the project (what is it for), while others help reason about how to make it (how it works, how it fits together). Some designers explicitly noted that sketching is a necessary part of the process; it would not be possible to move forward without making freehand drawings. When the idea is reasonably well-formed they will implement the model with a software tool. It is common for this to involve translating a hand-made sketch to a computer model (Figure 3).

[1] At the time of this writing I have conducted four interviews. More are planned.

Following the interview about their work practices, I gave participants a sketch (Figure 5) and asked them to implement it using their software tool of choice. The purpose of this task was to learn what problems people met when executing the common task of translating a sketch to a computer model.



(a) The part users set out to replicate.



(b) Drawing of how the part is used in context.

Figure 5: Sketches given to participants to implement in modeling software.

To design an object suitable for laser cutting, participants used Illustrator (3 users) or Rhino (1 user). In all cases, a designer's strategy involved a number of common activities: creating or editing boundaries, aligning or snapping items, using guide lines or reference points, measuring distances, specifying or changing lengths and angles, and creating finished "cut files" to send to the laser cutter. These tasks are in addition to selecting/deselecting, zooming, panning, and so on.

Participants in this experiment spent a good deal of time on operating overhead. Overhead includes (1) trying to find the appropriate tool for the next task, and (2) recovering from errors made when the wrong tool was selected, or (3) when the tool did not behave consistently with the user's intention. I have not performed an analysis of the low-level actions people performed, however I estimate that in my experiments, more than 50% of a designers time is spent this way.

For example, one user was aware of Illustrator's "Path Finder" tool and wanted to use it. This user searched the program's menu structure and hovered over buttons to read tool tips before finding it. Next, the designer invoked various functions of the Path Finder, using the keyboard shortcut to undo after each attempt, as he searched for the correct mode within the Path Finder's subcommand palette. This process lasted approximately 80 seconds before finally being able to continue.

Participants used the Undo function routinely. The dominant use of this was to revert after making a single failed attempt to edit some element. There were a few instances where the designer believed their current approach was flawed in such a way that it would be better to back up several dozen operations because a single decision early on was now observed to be problematic. This is consistent with the study by Akers *et. al* on the use of undo [1].

One may argue that the problems associated with operating overhead can be solved by experience or by addressing particular usability problems. Obviously, masterful Illustrator or Rhino users exist and would have no problem executing the design task from my experiment. However, the target audience for this research is avocational users, not experts. Cooper argues that the set of people that are truly expert users is not only quite small, it also changes over time [9]. A person might maintain expert status for some time, but eventually stops using the program as often, and eventually becomes an intermediate user, and will generally remain an intermediate user.

This notion of *perpetual intermediates* is evident in the designers I interviewed. One participant was a trained graphic designer and (according to the user's peers) was an Illustrator expert. However, this designer used some rather strange strategies during the translation task. In order to remove an unwanted line, this designer chose to create an opaque white rectangle to obscure it, rather than erase it. ("Don't tell anyone I did this", he said at the time).

Similar episodes are common: a person *should* know the 'correct' action, but takes an alternate approach. The alternate way may achieve the intended effect, but it might be

less efficient (more operations, longer execution time) or it might introduce unwanted complexity (e.g. invisible white objects in the model).

3.4 Implications for Design

The design observations described above account for a relatively small slice of hackers designing for laser cutters. It would be unwise to base an entire research agenda on these few interviews. However, the current proposal is based on much more than these observations. I have several years of experience working in an environment where both undergraduate and graduate students made things with rapid fabrication machines (including a laser cutter). The recently conducted interviews validate the observations I have been making for years: current design tools are hard to learn, provide poor interaction for novice or occasional users, and do not support idea exploration or informal representation that is so important during early design.

HCI practitioners often conduct usability studies to uncover particular problems in applications. The issues uncovered by these studies are then addressed by changing the software. This iterative process continues over years, leading to incremental improvements [6]. The interviews and observations from my study confirm that using current tools involve a great deal of needless overhead for the target population. However, rather than use this as a basis for incremental improvement to existing tools, I am interested in developing new tools the explore an alternate interaction paradigm that is easier and more effective for typical non-expert designers.

The ultimate goal of any laser cutter project is to make a final assembly that is made from laser-cut parts. Based on my interviews with the designers and my own experience, I identify several themes of user goals and needs.

3.4.1 Theme: Rough Representation

Designers need strategies to help think about (1) what the whole assembly should work and look like, (2) what individual pieces should be made, and (3) how those individual pieces

fit together. It is exceedingly difficult for designers to compose nontrivial plans without external aid. This is most commonly done via sketching on paper. Some designers make rough physical models as well. Sketching is important for transformational thinking. But sketching is also a physical activity involving incremental production of visual elements to support cognitive processing. The activity of sketching can be understood as an epistemic action; the outcome—a sketch—is the result of a sequence of pragmatic actions.

Current tools do not make any significant allowance for this goal. The system proposed here is based on sketching as the primary input method. This will let users be as rough, abstract, and imprecise as they like.

3.4.2 Theme: Precision

Designers of laser cut items would like to be able to specify some (but not necessarily all) dimensions. This is the main purpose of most current modeling tools, though it is often difficult for people to do it effectively. For example, in the translate task, there are two notches that are specified to be 20cm apart (refer to Figure 5a). Several participants created temporary geometry (rectangles and lines) that were set to 20cm long and placed on-screen. This served as a ruler that the designer could then use to position the notches. The ruler was then erased.

The proposed tool is somewhat unique in the area of sketch-based design tools in that it will give users the ability to be quite precise in identifying dimensions and constraints. It is hoped that using sketch recognition can allow people to state their intentions (regarding precision) substantially faster than with structured tools.

3.4.3 Theme: Rapid Editing and Manufacture

Participants reported they use a rapid *model-make-evaluate* loop. They produce several versions of objects with the laser cutter before it is finished. It is important to be able hold laser cut output in their hands to determine if they made an execution error. Equally as as often, the physical "rough draft" is necessary to evaluate their idea. Laser cutter projects

tend to be fairly quick and inexpensive in comparison to some other fabrication machinery like 3D printing.

Current tools impose clumsy requirements on designers to prepare models for laser cutting. For example, the cut line color need to have the RGB tuple $(0, 0, 255)$, and line thickness has to be "hair line" at 0.01mm. Managing this overhead is time-consuming. The proposed tool will handle this overhead on the user's behalf. Another laser-cutter specific problem concerns arranging the objects to cut. All objects must be inside a bounding rectangle that represents the stock material that will lay on the laser cutter bed.

A common problem exposed by making a physical draft is the discovery that pieces do not fit together properly. Some dimensions (such as notch widths) must then change. However, changing one dimension might have unwanted side-effects, causing a cascade of issues the designer must identify and fix, while trying to avoid introducing new problems.

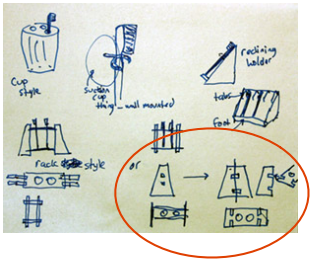
The proposed modeling tool assumes that the user might change the shape or parameterization of models later on, and facilitate these changes without creating more problems along the way.

4 Motivating Scenario

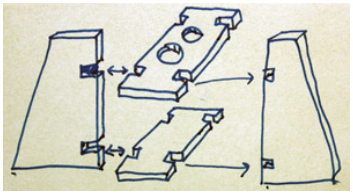
This section illustrates how a user might design the toothbrush holder shown in Figure 1a using the proposed system. There is also a video of this process as well [2]. This is included as a narrative to help the reader understand some of the general qualities the resulting system will include. It is not an exhaustive list of interaction techniques that will necessarily be implemented. The interaction described here is intended to allow people to *quickly* and *iteratively* use sketch input to gradually create precise models.

This narrative is admittedly contrived, because the user is 'designing' a part that has already been completed. It is included in order to present several proposed interaction techniques as they might be used together. The scenario in this section does not address

[2] http://vimeo.com/17997357



(a) Initial sketches of possible designs (preferred idea is circled).



(b) Sketch of the user's preferred design in greater detail.

(c) Final physical output made with a laser cutter.

Figure 6: Initial sketches and photograph of one part of the physical output.

the important interaction techniques that might lead a user to create several quick drawings used to think about the range of possibilities (as shown in Figure 6).

Some toothbrush holders rest on the sink while others are attached to the wall in some way; some hold a single toothbrush while others support up to four; some are similar to cups while others resemble racks. The user makes a variety of drawings (see Figure 6a) to help think about particular needs and preferences, and decides to focus on a rack-style holder, shown in the lower right corner of the initial sketch. This design features two slats supported at each end by side pieces. Toothbrushes fit through holes in the top slat and rest on the solid bottom slat.

So far the user had simply been sketching without the benefit of computation. But now the user is ready to begin thinking about details of how the object will be made, and the system can let the user supply information about dimensions.

There are three unique parts used in this design, shown in Figure 6b. To illustrate the interaction techniques featured in this system, the remainder of this section focuses on

the design of the part with circular holes in it (shown in Figure 6c). Figures 7–13 depict the design process.

The user realizes that toothbrush holders come in many sizes, and it is not completely apparent what the "right" dimensions are. Some properties like width and depth should be parametric, which allow the model to produce objects in a variety of sizes. Much of the following activity is guided by the desire to retain flexibility in the model.

The user begins drawing the part as a rectangle. Even though the user knows the finished slat will have notches cut from several locations, it is easy to begin by drawing it as a rectangle and removing the material for the notches later. The user asks the system to interpret the drawing, and the system replaces the input with a beautified rectangle [49].

The slat features four notches, each of the same size. To make these the user begins by drawing just one. A pen stroke that begins and ends outside a part, but traverses a part boundary will remove material (see Figure 8). This cutting gesture has been used in earlier systems like Teddy [24].

Next the user would like to add dimension details to the notch. Because the notch is relatively small on the screen, it is helpful to zoom in (Figure 9). Zooming is performed by a double-circle gesture, borrowing the zooming technique from Lineogrammer [64]. Clockwise and counter-clockwise gestures zoom in and out respectively. For a few moments after zooming, a widget appears that lets the user pan to the desired location.

The user would like to parameterize the notch dimensions so they may be changed later by name. To parameterize a length, the user draws a line with arrows on both ends near the desired line segment, and labels it with writing (Figure 10).

The first notch's geometry should be replicated for the next three. The user can give the system a *hint* [39] by selecting existing elements. The selection is used as a recognition candidate for the next editing operations—if the user sketches something that resembles a recently formed selection, the likelihood the user is replicating the selection is boosted. The user selects the notch by tracing over it. The selection is graphically acknowledged. Next the three more notches are drawn at the appropriate places. The *nd* and *nw* parame-

ters are implicitly copied from the original to the other notches because they were part of the hint (see Figure 11).

The user proceeds to give names to the height and width lengths. In addition the width is given a value, expressed in terms of height (see Figure 12). This updates the drawing to reflect the new width as a function of current height.

Designers often use external tools such as straight edges, French curves, or stencils when precision is desired. Alternately, people might lightly draw *construction lines* that guide subsequent drawing activity [8]. The user would like to draw two circles equidistant from the center of the slat. To precisely indicate the center of the part, the user draws two construction lines through the midpoints of two sides (Figure 13a). The designer uses these guides to place "fat dots" in several locations.

The user now wonders if three holes might fit, so two additional circles are made without using guides. The user decides to stick with the two-hole design, so the two new holes are erased using a scribble-out gesture (Figure 14). It is inevitable that the system's interpretations will occasionally conflict with the user's intentions. Rather than aiming for perfect recognition (which is impossible due to ambiguity) this system aims to enable users to easily recover from such recognition conflicts by providing appropriate interaction techniques.

The user continues to work, using interaction techniques already described to parameterize the distance between the center of the part, and the center of the holes. The user also draws one point where one of the hole boundaries will be (Figure 15a).

The user can hide non-boundary elements like parameters and construction lines and view the drawing (Figure 15b) before making the part on a laser cutter. Compare this with the final output shown earlier in Figure 6c.

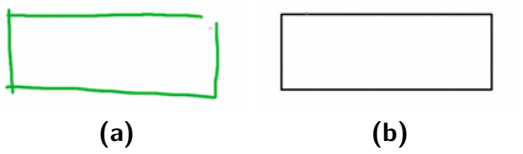


Figure 7: System identifies and rectifies closed shapes.

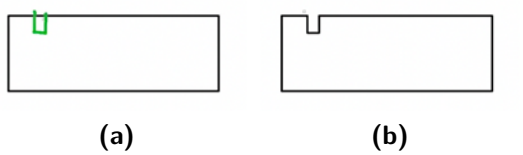


Figure 8: Remove (or add) material with gestures that cross the part boundary.

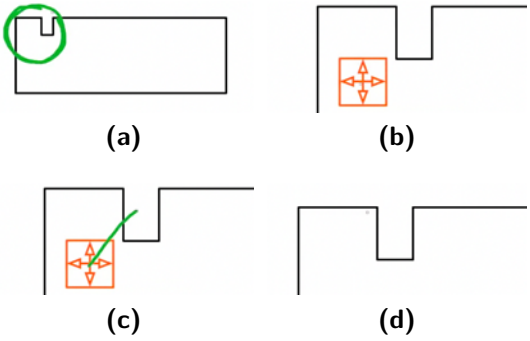


Figure 9: Double-spiral gesture zooms in (clockwise) or out (counter-clockwise) [64]. Pan widget appears after zoom gesture.

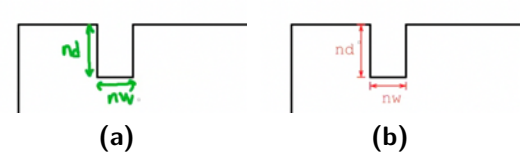


Figure 10: Make length parameters with double-headed arrows and text.

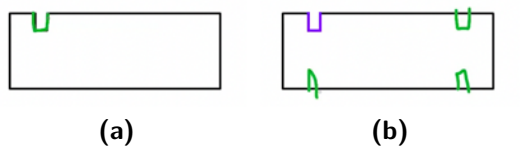


Figure 11: Make hints by selecting elements. The hint guides interpretation.

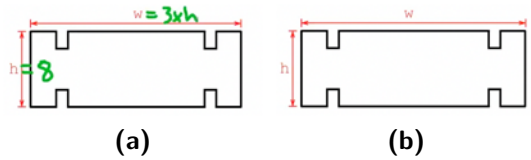


Figure 12: Set parameter values. Model updates to reflect new constraints.

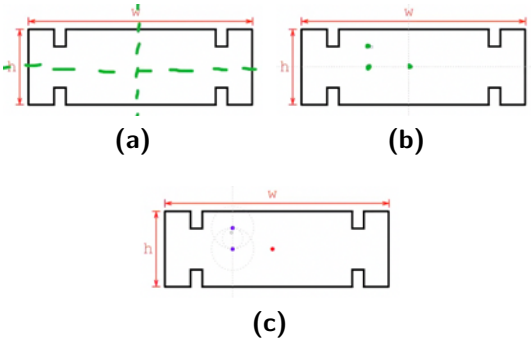


Figure 13: Guides facilitate accurate drawing. Guidelines (a) and reference points (b) drawn as indicated; circular guides (c) made by selecting reference points.

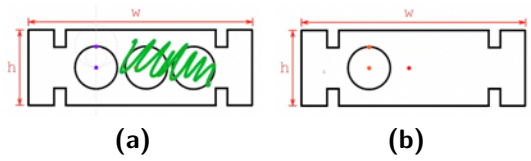


Figure 14: The user draws two holes but changes their mind and erases them with a scribble gesture.

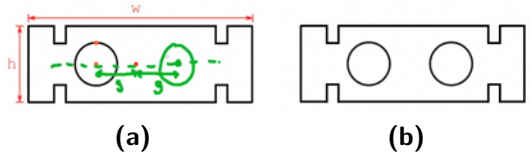


Figure 15: Combine techniques to finish part.

5 Related Work

While "the design process" has important differences from domain to domain, nearly all designers sketch. The ease of freehand drawing allows people to quickly externalize ideas to analyze their consequences, as well as to help see new solutions [32, 15]. Many design professions formalize the importance of sketching by making it central to the schooling: architects, industrial and interaction designers, and mechanical engineers are all explicitly taught to sketch. And even though sketching is not taught in other design domains (like software engineering), freehand drawing is still commonly done. Even people who consider themselves non-designers make quick drawings to help think through everyday problems like how the furniture in their house might be arranged. Freehand drawing is done by people of all ages [16] and experience levels [59].

The prior literature on technical aspects related to this project can be described in three categories: sketch recognition, calligraphic interaction, and rapid fabrication.

5.1 Sketch Recognition

Ideally, computers would understand sketches as readily as people. This is of course a very difficult problem of artificial intelligence. Therefore, researchers often restrict the problem by obliging users to draw shapes in prescribed ways [51, 62], or by limiting the user to work in specific domains with fairly small visual vocabularies (on the order of tens of meaningful primitives). Further improvements are made if the system can correctly identify contextual clues (often driven by domain semantics) to prune unlikely interpretations [18, 10].

Many sketch recognition approaches include at least two distinct tasks. The first is primitive ink parsing (also known as segmentation). This process interprets raw user input (time-ordered point sequences) into atoms such as straight lines, curves, intersections, and corners [48]. In systems that recognize multi-stroke elements, this step is also usually responsible for determining which atoms should be considered together. Segmenters often

leverage temporal data [56, 63], spatial data [28], or both [7].

The second task is to analyze these atoms to perform recognition. Recognition accuracy is dependent on the quality of the primitive ink parsing. To increase robustness and accuracy, some techniques supply multiple possible segmentations to the recognizer [2].

Design sketches from most domains combine diagrammatic ink and written language. It is helpful for the system to identify what is writing and what is not [57]. If done correctly, such meta-recognition can ease the recognition task by reducing the amount of input to interpret.

5.2 Calligraphic Interaction

Applications often interpret user input differently depending on which *mode* the program is in (e.g. line mode, circle mode, text mode, and so forth). This simplifies the computer's task because it is unambiguous how to interpret user input. But this complicates the user's task because they must manage modes while designing.

One way to make mode changing transparent is to infer the user's intention. The inferred mode protocol [54] analyzes user input to determine if user input is unambiguous enough for action to be taken. When ambiguity exists, the system might mediate by asking what was intended, or take no action and wait for additional input [36].

Another way of easing the problem of mode is to make it easier for users to change between them. This might involve buttons pressed with the non-dominant hand, or with pen gestures such as dwell or pigtail, or by using pressure[33].

Drawn gestures can be powerful, but users must first know of their existence. Some gestures seem easy enough that people do not necessarily need to "learn" them (e.g. scribble over something to erase it), but many gestures are hard to remember and might be nearly impossible to discover without assistance. GestureBar is a novel approach to let users discover and practice gestures [5].

Last, mode can be managed as it is in traditional applications, where people use on-screen widgets. These widgets may be present at all times [12], invoked via gesture [21, 31]

or by placing the pen near on-screen ink [37, 20].

Design sketches often involve a combination of text and pictures. In situations where the computer should recognize text (or at least recognize which ink specifies writing) it is necessary to distinguish between what is text and what is not. One approach is to automatically classify input as text or non-text based on a statistical analysis of its visual properties [47]. Another approach is to ask the user to specify what is text by performing a gesture. This was the approach taken by the developers of Lineogrammer [64].

The proposed work is similar to Lineogrammer and the ParSketch system [8, 41] in several key respects: they both offer the ability to create precise drawings by using sketch and gesture recognition, and they both eschew modal input when possible. The proposed work integrates many more sketch- and pen-based interaction techniques. Also, because it is in support of machined output, it is necessary that the drawing give enough detail that objects can be manufactured.

Need	**Technique**	**Example**
Select Item	Lasso Select	ScanScribe [53], Scriboli [22], Handle flags [20]
Delete	Scribble-out	GRAIL [11], Pegasus [23]
Undo/Redo	Gesture	Lineogrammer [64], Scriboli [22]
Zoom/Pan	Gesture	Lineogrammer [64]
Discern Text, Graphics	Recognizer, Gesture	Shilman *et. al.* [57]
Assign Constraint	Gesture, Mode	SketchPad [58], Stretch-a-sketch [17]
Remove/add from shape	Ink Recognition	Teddy [24]
Deform shape	Ink Recognition	SmoothSketch [29]
Create guides/rulers	?	?
Resize/move object	Context widgets	Hover widgets [21]
Hatch/fill region	Ink recognition	Thierjung *et. al.* [61]
Set pen properties	Crossing widgets	Crossy [3]
Train recognizer	Drawn example(s)	$1 Recogn. [62], Elect. Cocktail Napkin [18]
Invoke recognizer	Gesture, Button	Shadow button [37], Li *et. al.* [33]

Table 1: Interaction needs and techniques/systems that support and explore them.

Table 1 serves as a map of some of the needs users might commonly have when using a sketch-based system. This list is by no means exhaustive. Instead, it illustrates the variety of the common tasks have been addressed for sketch-based systems. When developing my system I will borrow heavily from these and other systems.

5.3 Sketching for Rapid Fabrication

Sketching has long been a subject of study by makers of physical things. Leonardo da Vinci's legendary sketchbooks are an early example of the utility of sketching for both thinking and for specifying. Demonstrations of the earliest computer-based sketching system, Sketchpad [58], mostly focused on drawing machinable parts.

Until fairly recently, many systems for 3D modeling or fabrication have used the term "sketching" as shorthand for "quick" in comparison to traditional CAD modeling interfaces [4, 50, 65]. Google SketchUp is a commercial example of this.

Many researchers interested in sketch input for designing physical objects have been concerned with recognizing 3D drawings, e.g. [34, 38]. Such work focuses on interpreting the 3D geometric meaning of a finished 2D sketch. In a sense, that approach assumes that the designer's creative work has concluded and the job of the computer is to translate the drawing into a 3D model. This contrasts with interactive approaches that aim to support an ongoing conversation between the human and computer. A conversational system tacitly acknowledges that the user's ideas are not fully formed from the onset, but become progressively clearer and more defined as they work.

With the availability of prototyping machines, sketch-based systems for fabrication are beginning to receive attention. The Furniture Factory accepts 2D sketch input that is recognized as a 3D configuration of planes [44]. The user adds pieces incrementally, so there is no need to make a complete sketch all at once. The system automatically computes how these planes can be joined together and generates a series of 2D pieces that to be produced with a laser cutter.

Saul's Sketch Chair [52] is another sketch-based system for generating furniture. It lets users sketch the contours of a chair's seat and back rest, and using a different drawing mode, add legs. The system includes a sophisticated physical simulator to let the designer explore its physicality (for example to determine if it will remain upright), and if it will be comfortable. It also allows designers to change subtle properties of curves using on-screen control handles. My proposed system differs from Sketch Chair in two important

respects. First, Saul's system presumes the user is making a chair or chair-like object, and provides highly specialized software analysis tools. To borrow from Henry Ford, you can make anything you want, as long as it is a chair. My system is more general, so the range of possibilities with my tool is wider. Second, Sketch Chair (and many systems like it) constrain designers to using a small set of part junction types. My system does not propose to include pre-specified junction types. This is both liberating (because designers are free to chose whichever type they want) but also encumbering (because they must design the junctions themselves).

Plushie lets people make bulbous textile objects such as plush toys or balloons [40]. Users iteratively design objects by drawing object boundaries or giving editing commands by sketched gesture. The interaction is based on prior systems in the Teddy family tree [24].

6 My Prior Work

The Designosaur [44] is a prototype system for kids to design their own wooden dinosaur "skeletons". Users begin by drawing individual bone outlines and specify notch locations where the bone joins another. To support this, the user can name bones and apply those names to indicate how bones fit together.

We noticed that editing bone boundaries presented some challenges. First, dinosaur model bones are frequently symmetric (e.g. ribs or vertebrae). The system featured a simple "mirror mode" that helped users make the parts symmetric about the horizontal or vertical axes. Second, it is useful to be able to change the shape of bones easily. To tweak the bone curvature without redrawing, we invented a local editing technique called Flow selection.

Flow selection [26] is a modeless, time-based selection and editing technique for picking a region of a 2D boundary and changing its shape. It is especially useful in cases such as the Designosaur when organic curves are desired. The user sets their stylus down near a boundary and holds it still as a selection begins to form. An apt metaphor is that the

pen is "heating" the region, which is strongly selected near the pen, and progressively less selected as distance increases along the boundary path. Then the user moves the pen to begin reshaping the selected region—the more strongly selected the ink, the more it moves. The user may lift their pen up to end the operation, or hold it still once again to smooth the region. This is helpful in cases where the user input includes unwanted bumps or jagged lines.

Whereas the Designosaur is based on sketching, the follow-on system is based on programming. FlatCAD is a system for designing and manufacturing things on a laser cutter by writing programs in a language called FlatLang [25]. FlatLang features a "flying turtle"—a 3D version of the LOGO turtle—which lets the user compose 3D models of objects built from flat pieces. This helps users to specify and see how parts will fit together. FlatCAD made it easy to design everyday items using parameters, so the same model (FlatLang program) could be used to generate a variety of physical objects. The objects shown earlier (Figure 1) were made using FlatCAD.

My experience with the Designosaur sparked an interest in sketch-based interaction techniques, while FlatCAD kindled an interest in tools for making precise, machinable models. The Designosaur lacks the ability to make precise edits: make a notch *here* facing *this direction*; make *this* collinear with *that*; and so on. FlatCAD addressed this issue but introduced roughly the opposite problem: it is impossible to be non-specific or tentative when writing a FlatLang program. The current proposal is an effort to bridge the gap between the positive aspects of sketching (rough, tentative, fluid) with the positive aspects of FlatCAD-style modeling (precise, specific, parametric).

To better understand the space of sketch-based design tools, I co-authored an extensive survey with my thesis committee [27]. This survey was motivated by the question: *After forty years of research on computational support for sketching, why there are so few real world applications of this technology?* In my personal view, a significant reason is that we lack appropriate sketch-based interaction techniques. The proposed work seeks to establish a better understanding of what constitutes good sketch-based interaction. The proposed

system is a vehicle for this research.

7 Evaluation

Interaction techniques are challenging to evaluate. There is a tradition in human-computer interaction to evaluate novel interaction techniques in a laboratory setting. People perform simple tasks designed to evaluate the technique in isolation. While this evaluation approach has its benefits, it avoids the reality that people use interaction techniques *with other techniques* and *in service of a larger purpose*.

Therefore, the proposed system will be evaluated as a whole, rather than testing isolated pieces. The proposed system offers a set of techniques that are intended to work harmoniously together to support users in the realistic goal of designing household objects using flat material. To test if the system works as intended it will be compared with existing tools such as Illustrator, SolidWorks, SketchUp, Rhino, as well as plain pencil and paper.

The evaluation criteria in this "whole system" analysis focus on what users are able to do (or not do) that would be possible with traditional structured design software. What is possible to create using this system? What is *not* possible? What are the common interaction problems or errors? Do users tend to develop particular usage patterns that might be exploited? Or do usage patterns indicate underlying flaws with the system?

Iterative evaluation will be used throughout the development process. Therefore, the exact list of features (and perhaps evaluation criteria) must be flexible. Test participants will be asked to design and fabricate a functional object such as the toothbrush holder in Figure1a. Object choice will depend on the system's capabilities at the time of testing.

This whole-system approach will allow me to gauge several factors that compare the proposed system with existing WIMP-based tools:

1. Time taken for design and fabrication
2. Number of unique design ideas (as measured in [14])

3. Overall user satisfaction
4. User satisfaction with individual techniques
5. User satisfaction with certain combinations of techniques
6. How "learnable" the interaction style is
7. How system supports (or does not support) alternate working styles
8. How well tool helps people *think*, *create*, *explore*, *visualize*, *specify*, and *fabricate*
9. User feedback: criticism and suggestions

This evaluation rubric does not explicitly address technical factors traditionally found in work done on sketch based systems. These factors include recognition accuracy or computational complexity of various algorithms. Measures such as user satisfaction should account for such aspects: if the system misinterprets input but provides adequate methods for recovery, recognition accuracy is not a problem.

8 Contribution

Currently there are few interaction techniques for sketch-based interfaces, and even fewer examples of coherent sets of techniques. As stated earlier, it is desirable to provide a design system in which the user can do everything without setting down the stylus.

This thesis will explore the space of calligraphic interaction techniques to support researchers seeking to make sketch-based design systems in contexts beyond the laser cutter scenario described above.

This thesis will contribute the following:

1. A set of calligraphic interaction techniques. Some will be based on existing techniques (e.g. Lineogrammer's zoom), others will be novel. Here, *technique* is broadly defined to include interactive on-screen widgets, gestural phrases, and user interface conventions.
2. An analysis of *the contexts and use cases* when each technique is appropriate. For

example, rectification might be appropriate for latter stage design but distracting during early exploratory sketching.

3. An analysis of *which techniques* are complementary.
4. Recommendations for developing further techniques.
5. Software engineering implementation details.

The best interactive systems have discoverable, consistent interaction techniques working in concert. Companies such as Microsoft and Apple have user interaction guidelines for various platforms (Windows, Windows Mobile, OS X, iOS) that give developers standard advice for developing such systems. Pen- and touch-oriented interaction guides exist for TabletPCs and (recently) the iPad, but these devices are by no means sketch-based in that the interaction is still expected to be largely unambiguous, and requires little interpretation. No guide yet exists for sketch-based interactive systems. This thesis will provide a concrete step to providing one.

9 Time line

In this section I first describe a short schedule for developing the sketch-based system, followed by a longer-term schedule for completing the dissertation.

9.1 System Development

I have already written a good deal of code this project, so it should not take long to develop a rudimentary prototype. Existing code includes ink analysis and supported data structures (such as Masry's Angular Distribution Graph [38] among others) and Ouyang's PCA-based printed character recognizer [46]. External tools will be used whenever possible.

The following table presents a tentative schedule for system development. This schedule assumes I will implement the system as described in Section4. It will change if my user study indicates I should focus on something else.

Milestone	Num. Weeks	Topic
Milestone 1	1 week	Primitive stroke analysis to generate constraints
Milestone 2	1 week	Establish boundary model
Milestone 3	1 week	Recognize gestures to add/subtract material
Milestone 4	1/2 week	Erase unwanted ink with scribble gesture.
Milestone 5	1/2 week	Implement zoom and pan widgets.
Milestone 6	1 week	Indicate and name linear dimensions.
Milestone 7	1/2 week	Indicate and name angular and location dimensions.
Milestone 8	1/2 week	Indicate guides (points, lines, circles).
Milestone 9	1 week	Indicate hints and use them for interpretation.
Milestone 10	1 week	Indicate parameter values (scalar)
Milestone 11	1 week	Indicate parameter values (expression)
	9 weeks total	

Throughout this process I will conduct informal user studies to calibrate my work. Additionally, at each step in the process I will ensure that the system does what it should by using it to make items with a laser cutter.

A common heuristic project managers use to predict completion time is to multiply the predicted time by three—this brings the expected duration to 27 weeks, or about six months.

9.2 Dissertation Schedule

The schedule for completing and defending the dissertation follows. Because the completion time for system building has such a wide spread, best- and worst-case schedules are given.

Component	**Num. Weeks**	**Remark**
System Development	9–27 weeks	Schedule given above in section 9.1.
User Interviews	2–4 weeks	Understand current problems and processes.
System Evaluation	2–4 weeks	Includes modifications based on feedback. Concurrent with system development.
Dissertation draft	4–8 weeks	Borrow from literature survey [27] and this proposal.
Dissertation revisions	3–8 weeks	—
Defense preparation	2 weeks	—
Summary: 9–27 weeks development, 13–26 weeks other = *22–53 weeks total*		

References

[1] David Akers, Matthew Simpson, Robin Jeffries, and Terry Winograd. Undo and erase events as indicators of usability problems. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 659–668, New York, NY, USA, 2009. ACM.

[2] Christine Alvarado and Randall Davis. Dynamically constructed bayes nets for multi-domain sketch understanding. In *International Joint Conference on Artificial Intelligence*, 2005.

[3] Georg Apitz and François Guimbretière. CrossY: a crossing-based drawing application. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 3–12, New York, NY, USA, 2004. ACM.

[4] Mark Bloomenthal, Robert Zeleznik, Russell Fish, Loring Holden, Andrew Forsberg, Richard Riesenfeld, Matt Cutts, Samuel Drake, Henry Fuchs, and Elaine Cohen. SKETCH-N-MAKE: Automated machining of CAD sketches. In *Proceedings of ASME DETC'98*, pages 1–11, 1998.

[5] Andrew Bragdon, Robert Zeleznik, Brian Williamson, Timothy Miller, and Joseph J. LaViola, Jr. Gesturebar: improving the approachability of gesture-based interfaces. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 2269–2278, New York, NY, USA, 2009. ACM.

[6] Bill Buxton. *Sketching User Experiences*. Morgan Kaufmann Publishers, 2007.

[7] Sonya Cates. *Combining Representations for Improved Sketch Recognition*. PhD thesis, Massachusetts Institute of Technology, September 2009.

[8] Pedro Company, Manuel Contero, Peter Varley, Nuria Aleixos, and Ferran Naya. Computer-aided sketching as a tool to promote innovation in the new product development process. *Computers in Industry*, 60(8):592–603, 2009.

[9] Alan Cooper. *The inmates are running the asylum*. Sams, 1999.

[10] Ellen Yi-Luen Do. *The Right Tool at the Right Time: Investigation of Freehand Drawing as an Interface to Knowledge Based Design Tools*. PhD thesis, Georgia Institute of Technology, 1998.

[11] T. O. Ellis, J. F. Heafner, and W. L. Sibley. The GRAIL project: An experiment in man-machine communications. Technical Report RAND Memorandum RM-5999-ARPA, RAND Corporation, 1969.

[12] Kenneth D. Forbus, Jeffrey Usher, and Vernell Chapman. Sketching for military courses of action diagrams. In *Proceedings of Intelligent User Interfaces '03*, 2003.

[13] Neil Gershenfeld. *Fab: The coming revolution on your desktop—from personal computers to personal fabrication*. Basic Books, 2005.

[14] Vinod Goel. *Sketches of Thought*. MIT Press/A Bradford Book, Cambridge, MA, 1995.

[15] G. Goldschmidt. The dialectics of sketching. *Creativity Research Journal*, 4(2):123–143, 1991.

[16] G. Goldschmidt. The backtalk of self-generated sketches. In *Spatial and Visual Reasoning in Design*. Key Center of Design Computing, Syndey, Australia, 1999.

[17] Mark D. Gross. Stretch-a-sketch, a dynamic diagrammer. In *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 232–238, 1994.

[18] Mark D. Gross and Ellen Yi-Luen Do. Ambiguous intentions: A paper-like interface for creative design. In *UIST '04: ACM Conference on User Interface Software Technology*, pages 183–192, Seattle, WA, 1996.

[19] Mark D. Gross and Ellen Yi-Luen Do. Educating the new makers: Cross-disciplinary creativity. *Leonardo*, 42(3):210–215, 2009.

[20] Tovi Grossman, Patrick Baudisch, and Ken Hinckley. Handle flags: efficient and flexible selections for inking applications. In *Proceedings of Graphics Interface 2009*, GI '09, pages 167–174, Toronto, Ont., Canada, Canada, 2009. Canadian Information Processing Society.

[21] Tovi Grossman, Ken Hinckley, Patrick Baudisch, Maneesh Agrawala, and Ravin Balakrishnan. Hover widgets: using the tracking state to extend the capabilities of pen-operated devices. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 861–870, New York, NY, USA, 2006. ACM.

[22] Ken Hinckley, Patrick Baudisch, Gonzalo Ramos, and Francois Guimbretiere. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 451–460, New York, NY, USA, 2005. ACM.

[23] Takeo Igarashi, Satoshi Matsuoka, Sachiko Kawachiya, and Hidehiko Tanaka. Interactive beautification: a technique for rapid geometric design. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 105–114, New York, NY, USA, 1997. ACM.

[24] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. In *ACM SIGGRAPH'99*, pages 409–416, Los Angeles, California, 1999.

[25] Gabe Johnson. FlatCAD and FlatLang: Kits by code. In *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing*, 2008.

[26] Gabe Johnson, Mark D Gross, and Ellen Yi-Luen Do. Flow selection: A time-based selection and operation technique for sketching tools. In *2006 Conference on Advanced Visual Interfaces*, pages 83–86, Venice, Italy, 2006.

[27] Gabe Johnson, Mark D. Gross, Jason Hong, and Ellen Yi-Luen Do. Computational support for sketching in design: a review. *Foundations and Trends in Human-Computer Interaction*, 2(1):1–93, 2009.

[28] Levent Burak Kara and Thomas F. Stahovich. An image-based, trainable symbol recognizer for hand-drawn sketches. *Computers and Graphics*, 29(4):501–517, 2005.

[29] Olga A. Karpenko and John F. Hughes. SmoothSketch: 3D free-form shapes from complex sketches. *ACM Trans. Graph.*, 25(3):589–598, 2006. 1141928.

[30] David Kirsh and Paul Maglio. On distinguishing epistemic from pragmatic action. *Cognitive Science*, 18:513–549, 1994.

[31] Gordon Kurtenbach and William Buxton. Issues in combining marking menus and direct manipulation techniques. In *Symposium on User Interface Software and Technology*, pages 137–144. ACM, 1991.

[32] Bryan Lawson. *How Designers Think: The Design Process Demystified*. Architectural Press, 4th edition, 1997.

[33] Yang Li, Ken Hinckley, Zhiwei Guan, and James A. Landay. Experimental analysis of mode switching techniques in pen-based user interfaces. In *CHI 2005*, 2005.

[34] H. Lipson and M. Shpitalni. Correlation-based reconstruction of a 3D object from a single freehand sketch. In *AAAI 2002 Spring Symposium (Sketch Understanding Workshop)*, 2002.

[35] Hod Lipson and Melba Kurman. *Factory@Home: The emerging economy of personal manufacturing*. Report Commissioned by the Whitehouse Office of Science & Technology Policy, 2010.

[36] Jennifer Mankoff. Providing integrated toolkit-level support for ambiguity in recognition-based interfaces. In *CHI '00: CHI '00 extended abstracts on Human factors in computing systems*, pages 77–78, New York, NY, USA, 2000. ACM.

[37] Diane Marinkas, Robert C. Zeleznik, and Joseph J. LaViola, Jr. Shadow buttons: exposing wimp functionality while preserving the inking surface in sketch-based interfaces. In *SBIM '09: Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 159–164, New York, NY, USA, 2009. ACM.

[38] M. Masry, D. Kang, and Hod Lipson. A freehand sketching interface for progressive construction of 3D objects. *Computers and Graphics*, 29(4):563–575, 2005.

[39] Richard G. McDaniel and Brad A. Myers. Building applications using only demonstration. In *Proceedings of the 3rd international conference on Intelligent user interfaces*, IUI '98, pages 109–116, New York, NY, USA, 1998. ACM.

[40] Yuki Mori and Takeo Igarashi. Plushie: An interactive design system for plush toys. In *Proceedings of SIGGRAPH 2007*. ACM, 2007.

[41] Ferran Naya, Manuel Contero, Nuria Aleixos, and Pedro Company. Parsketch: a sketch-based interface for a 2d parametric geometry editor. In *Proceedings of the 12th international conference on Human-computer interaction: interaction platforms and techniques*, HCI'07, pages 115–124, Berlin, Heidelberg, 2007. Springer-Verlag.

[42] New York Times. 3-D printing spurs a manufacturing revolution, http://www.nytimes.com/2010/09/14/technology/14print.html, September 2010.

[43] Mark W. Newman and James A. Landay. Sitemaps, storyboards, and specifications: a sketch of web site design practice. In *DIS '00: Proceedings of the 3rd conference on Designing interactive systems*, pages 263–274, New York, NY, USA, 2000. ACM.

[44] Yeonjoo Oh, Gabe Johnson, Mark D. Gross, and Ellen Yi-Luen Do. The Designosaur and the Furniture Factory: Simple software for fast fabrication. In *2nd Int. Conf. on Design Computing and Cognition (DCC06)*, 2006.

[45] Dan R. Olsen, Jr. Evaluating user interface systems research. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 251–258, New York, NY, USA, 2007. ACM.

[46] Tom Y. Ouyang and Randall Davis. A visual approach to sketched symbol recognition. In *Proceedings of the 2009 International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[47] Rachel Patel, Beryl Plimmer, John Grundy, and Ross Ihaka. Ink features for diagram recognition. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, SBIM '07, pages 131–138, New York, NY, USA, 2007. ACM.

[48] Brandon Paulson and Tracy Hammond. Paleosketch: accurate primitive sketch recognition and beautification. In *Proceedings of the 13th international conference on Intelligent user interfaces*, IUI '08, pages 1–10, New York, NY, USA, 2008. ACM.

[49] Theo Pavlidis and Christopher J. Van Wyk. An automatic beautifier for drawings and illustrations. *SIGGRAPH Comput. Graph.*, 19(3):225–234, 1985.

[50] David Pugh. *Using Interactive Sketch Interpretations To Design Solid Objects*. PhD thesis, Carnegie Mellon University, Department of Computer Science, 1993.

[51] Dean Rubine. Specifying gestures by example. *SIGGRAPH Computer Graphics*, 25(4):329–337, 1991.

[52] Greg Saul, Manfred Lau, Jun Mitani, and Takeo Igarashi. SketchChair: An all-in-one chair design system for end users. In *Proceedings of the 5th International Conference on Tangible, Embedded and Embodied Interaction (TEI2011)*, 2011.

[53] Eric Saund, David Fleet, Daniel Larner, and James Mahoney. Perceptually-supported image editing of text and graphics. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 183–192, New York, NY, USA, 2003. ACM.

[54] Eric Saund and Edward Lank. Stylus input and editing without prior selection of mode. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 213–216, New York, NY, USA, 2003. ACM.

[55] D. A. Schon and G. Wiggins. Kinds of seeing and their functions in designing. *Design Studies*, 13(2):135–156, 1992.

[56] Tevfik Metin Sezgin. Feature point detection and curve approximation for early processing of free-hand sketches. Master's thesis, Massachusetts Institute of Technology, 2001.

[57] Michael Shilman, Zile Wei, Sashi Raghupathy, Patrice Simard, and David Jones. Discerning structure from freeform handwritten notes. In *Proceedings of International Conference on Document Analysis and Recognition (ICDAR) 2003*, 2003.

[58] Ivan Sutherland. SketchPad: A man-machine graphical communication system. In *Spring Joint Computer Conference*, pages 329–345, 1963.

[59] M. Suwa and B. Tversky. What do architects and students perceive in their design sketches? a protocol analysis. *Design Studies*, 18:385–403, 1997.

[60] The Economist. A factory on your desk. http://www.economist.com/node/14299512. September 2009.

[61] Raphael Thierjung, Florian Brieler, and Mark Minas. On-line recognition of hatched and filled regions in hand-drawings. In *Proceedings of IUI 2009 Workshop on Sketch Recognition*, 2009.

[62] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *UIST '07: Proceedings of ACM Symposium on User Interface Software and Technology*, pages 159–168, New York, NY, USA, 2007. ACM.

[63] Aaron Wolin, Brandon Paulson, and Tracy Hammond. Sort, merge, repeat: An algorithm for effectively finding corners in hand-sketched strokes. In *Proceedings of Eurographics 6th Annual Workshop on Sketch-Based Interfaces and Modeling*, 2009.

[64] Robert C. Zeleznik, Andrew Bragdon, Chu-Chi Liu, and Andrew Forsberg. Lineogrammer: creating diagrams by drawing. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 161–170, New York, NY, USA, 2008. ACM.

[65] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. SKETCH: An interface for sketching 3D scenes. In *SIGGRAPH 1996*, 1996.