

THE V-SKETCH SYSTEM
MACHINE ASSISTED DESIGN EXPLORATION IN VIRTUAL REALITY

A Thesis
Presented to the Faculty of the Graduate School
of Cornell University
In Partial Fulfillment of the Requirements for the Degree of
Master of Science

by
James Michael Mahoney
August 2018

© 2018 James Michael Mahoney

ABSTRACT

Many critical decisions of any design process occur in the early exploratory stages when a design is in its most rough form. This project presents a process to enhance this early stage design exploration by making it easier and more intuitive to explore a wider range of possibilities. It's called V-Sketch. We employ digital computation to translate from intuitive sketching to 3D geometric elements¹. Unlike earlier work, we decouple the sketch analysis from the creation of the 3D forms allowing for a modular and therefore more flexible representation of the resulting elements. This is accomplished by introducing an intermediary, qualitative description of the sketches. "Sketch analysis" using machine learning techniques is employed to generate this intermediate description of the sketch data as the designer draws. A "reconstruction function" then translates the description into a resulting 3D form. The introduction of a reconstruction function allows an artist or designer to flexibly redefine the solution space of possible 3D forms from the same sketch as they desire without having to retrain the machine learning algorithms. Our methods are focused primarily on architectural design but potentially generalizes to other visual design problems such as, sculpture, virtual environments and industrial design.

This thesis presents a conceptual framework for the V-Sketch system. We present the interactive construction of an open source data set of labelled three-dimensional drawing data, and we show a prototype implementation of the system that demonstrates the approach and motivation for the project.

¹ Note: For clarity, we're using the word "model" in the context of a machine learned algorithm and avoiding it in reference to 3D geometric models that might be created in Maya for example.

BIOGRAPHICAL SKETCH

James Mahoney was born in West Roxbury, MA in April of 1960. His father, Joseph was an engineer at the Boston Naval Shipyard and his mother Alice was a telephone operator and homemaker. He has three wonderful older siblings, Joe, Janice and Jeanne. James graduated from Cornell with a bachelor's degree in architecture and a minor in engineering in 1984. He has since been at the forefront of digital media working at places such as Hanna-Barbera Productions and Microsoft Research. While at Cornell in the fall of 1983 James created an interactive paint program which included many novel features such as a soft air brush, smudge tool, gradient tool, masking and blend modes. Also in 1983, he and two other students were the very first at Cornell Architecture to use a 3D digital print of their project in a final presentation. At Hanna-Barbera James and a coworker produced the now famous 3D chrome swirling star logo ubiquitous in the late 1980's and early 1990's. James and Cindy Ball produced the animated short, "2D Life in a 3D World," presented in the Siggraph, Electronic Theater in 1997. James Mahoney's artwork has been featured at several prominent art galleries and museums in multiple countries. Twice his art work was awarded the coveted Los Angeles Weekly, "Art Pick of the Week." More recently James joined the faculty at Dartmouth as a Lecturer in the computer science department focusing on digital arts, generative design and augmented reality. James is married to his beloved Elaine and they have an amazing daughter named Sierra.

This thesis is dedicated with love to my amazing mother Alice, my wonderful daughter Sierra and my lovely companion Elaine.

ACKNOWLEDGMENTS

I would first like to thank my thesis advisor Dr. Donald Greenberg, the Jacob Gould Schurman Professor of Computer Graphics at Cornell University. He has been and remains a great friend, visionary mentor and teacher. He consistently allowed this paper to be my own work but steered me in the right direction whenever I needed it. I am grateful and forever indebted. I would also like to acknowledge Dr. Bart Selman of the department of computer science at Cornell University as the second advisor of this thesis, and I am so very gratefully to hm for his very valuable discussions and insights regarding this thesis. In addition, I would also like to acknowledge Dr. Michael Cohen the director of computational photography at Facebook as an ad-hoc advisor of this thesis. I am tremendously thankful for his gentle guidance, wisdom and friendship throughout this thesis project. I'd also like to thank the architecture department for providing a generous graduate teaching assistanceship as well as Pixar for generously supporting graphics research at Cornell. Thanks goes to Facebook for kindly supplying an Oculus Rift and Nvidia for providing the amazing Titan X graphics cards required for this work.

I would also like to thank my amazing friends at Cornell for their support and fellowship the entire way. Thank you Prof. Steven Marschner, Scott Wehrwein, Chris Morse, Leul Tesfaye, Henry Chen, Corey Torres, Ethan Arnowitz and Nathan Adara, as well as all the wonderful lab undergrads, I will always cherish our time together. Finally, I must express my very profound gratitude to my partner Elaine for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without her. Thank you.

James Mahoney

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION.....	1
CHAPTER 2: RELATED WORK.....	8
CHAPTER 3: DEFINITION OF TERMS.....	18
CHAPTER 4: MACHINE ASSISTED DESIGN EXPLORATION.....	24
CHAPTER 5: TRAINING DATA SET.....	44
CHAPTER 6: V-SKETCH PROTOTPYE.....	62
CHAPTER 7: CONCLUSION.....	76
REFERENCES.....	80

CHAPTER 1

INTRODUCTION

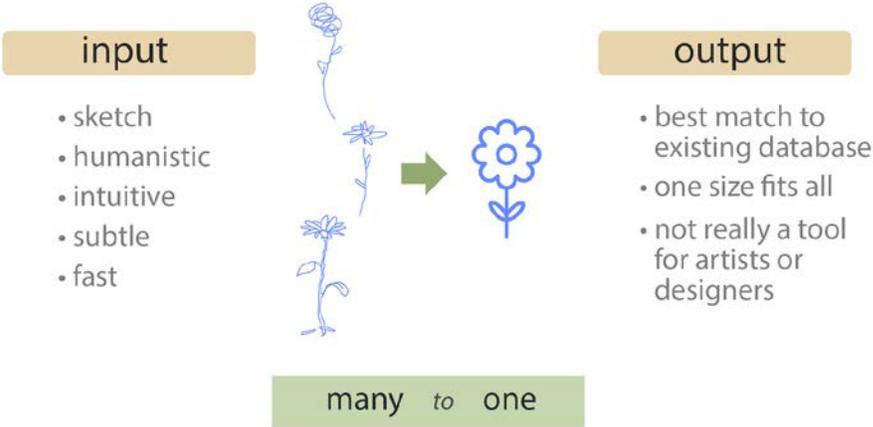
We live in an age of rapid technological change and disruption. It is quite clear that artificial intelligence, or A.I., will be employed in various ways to help design many things, including buildings, industrial objects, pop songs and pulp fiction. In some cases, machines may even become the sole designers. Alternatively, we believe that humans working collaboratively with machines will ultimately be more effective than either working separately. This thesis is an exploration of ways to enhance human involvement, often referred to as “human agency,” in a human-machine symbiotic system. Our objective is to help designers employ emerging technologies, such as “virtual reality” (VR) and artificial intelligence to become better, more effective designers by working collaboratively with digital media and algorithms.

“The defining art-making technology of our era will be AI. But this won’t be the kind of artificial intelligence of our past imagination—it’ll be the augmented intelligence of the present.”

- Rama Allen

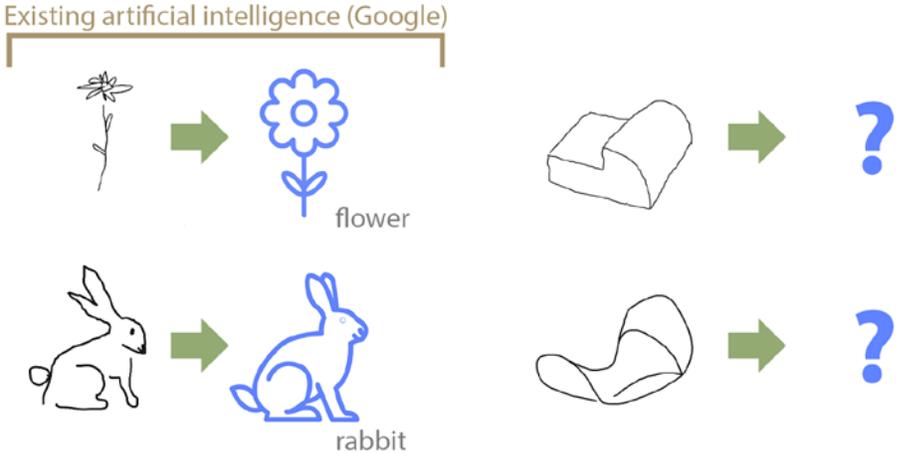
A critical aspect of this is to determine the optimal way to interact with the machine. What is the best way for a human designer to communicate quickly and intuitively? We propose, it is the “**sketch.**” Sketching in early design exploration allows an artist or designer to more easily tap into their intuition and achieve a state of mental flow that might otherwise be difficult to achieve. Many existing efforts are

aimed at capturing the semantics of an input sketch, "is this a sheep?", as for example with Google's *Autodraw* or *Quick Draw!*. (Figure 1)



Existing examples convert many sketches to a single iconic image Figure 1

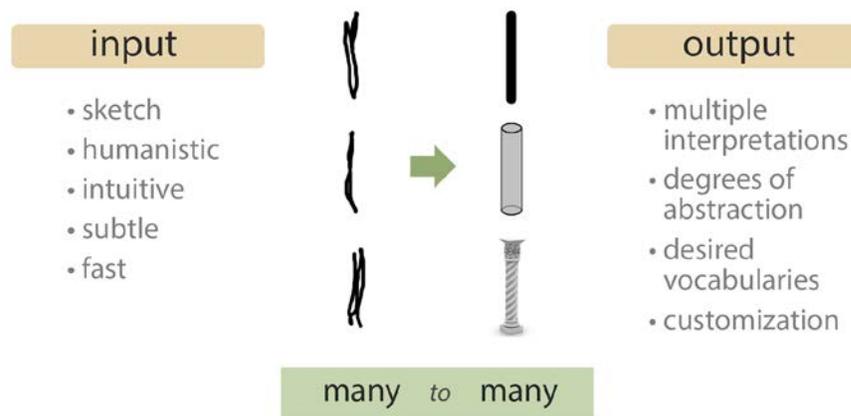
This is essentially a classification problem where machine learning is employed to determine what is the thing being sketched. This technique presents unique problems when used to classify abstract forms needed for many visual design tasks. (Figure 2)



How does one classify an abstract form? Figure 2

V-Sketch uses machine learning to capture the abstract qualities of the sketch. This inferred numerical description of the sketch can then be used as input to a separate

function (the **reconstruction function**) that interprets the description and returns a 3D object that best fits the description. This approach enables an artist or designer to leverage machine learning while retaining a more flexible, modular interpretation. This approach improves expressiveness and can produce a more customized 3D interpretation of a sketch. (Figure 3)



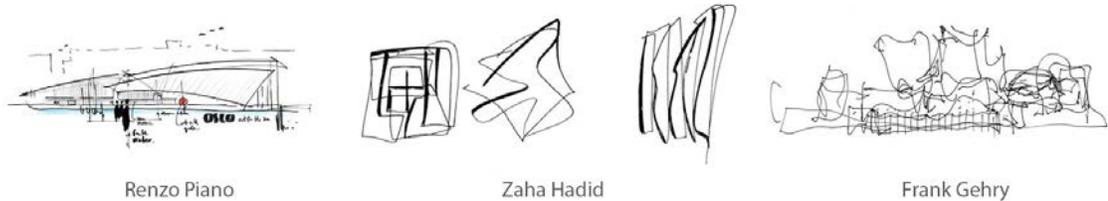
Goal is to convert input sketches to various output forms Figure 3

So, what is a sketch? The design process is much like a funnel, where the wide mouth represents the early stages where there are still many possibilities. These are explored and evaluated, and the range is reduced as the ideas solidify and the design become more specific and concrete. The earliest phase of design is exemplified by **the sketch** while the later stages are defined more by drawings and mockups. The process leads ultimately to detailed and precise design documents usually needed to construct or make whatever it is.

The cornerstone of this thesis project rests on two pillars.

- Sketching to interactively engage with the machine in the design process has significant benefits over selecting elements of a design from a list or table.

- Having the ability to create and swap libraries easily, without retraining the algorithms, will allow more customization and expressivity.



Examples of professional architectural sketches Figure 4

Let us now examine these more closely.

Why is sketching so common a starting point for many forms of design, architecture, sculpture and painting? (Figure 4) Here are some essential things our research has revealed about sketching.

- A sketch is **fast** and therefore not too “precious” (the process has **immediacy**)
- This speed reflects a desire to **explore** a wide array of possibilities.
- Sketching has a minimal separation of action (and artifact) from thought.
- As such, sketching often allows **subconscious intuition** to find expression
- In this way, as the sketch unfolds, there’s a **dialog** between the drawn marks and the just forming thoughts and intentions of the person sketching. It’s almost like the act of sketching helps to pull ideas from the mind, transforming vagaries into a physical form.
- This freely moving creativity has been described as a mental state of “**flow**.”²
- The early design process is more about **discovery** than communication.

² The popular psychology book “Flow” by Mihaly Csikszentmihalyi was published in 1990

- Discovery is enhanced by the advantage of “productive **ambiguity**” and abstraction. This lack of specificity keeps the design concepts open to reinterpretation and the mind flows.
- Among other things, many people report an affection for the **physicality** of mark making; the speed, pressure, sequencing and even, the tactile experience of sketching.

With today’s digital technology the creation modes have substantially been expanded to include pen input on tablet computers or laptops. Sketching software allows for a great variety of sketching experiences including various simulated media (Adobe Sketch), drawing in three dimensions (“Cuttlefish” [GUTIERREZ]) or even drawing in space using AR or VR equipment (Tilt Brush, “vSpline” [ARNOWITZ]). With the inclusion of ever faster and more capable computers we find ourselves at the dawn of the age of thinking machines which ultimately leads to this question.

Is it not possible to employ today’s artificial intelligence techniques to enhance the sketching experience and early design process by improving the human-machine symbiotic interface?

Further, is it not possible to employ data analysis and machine learning to interpret sketches without constraining a designer to limited, fixed, predefined resulting forms?

This thesis presents a machine enhanced design process that augments the sketching part of early design exploration and leverages computation to move analysis and substantive evaluation closer to the spark of inspiration. This is achieved while maintaining unconstrained design potential by introducing flexibility in the pipeline

after the machine analysis of the sketch has been performed. The **reconstruction function** opens the definition of the possible resulting forms, or “solution space,” to re-imagining by the designer to explore whatever design problem is at hand unconstrained and with the most appropriate forms.

The general approach is to apply data science and machine learning techniques to train a system to interpret the desired intent of the author’s sketches. These algorithms produce an intermediary descriptor. This descriptor is ultimately represented as an array of attribute values and a set of keywords. These in turn can be used as input parameters to a reconstruction process to constitute a 3D object as an element of a composition. This can be done with a combination of formal logic and access to a library, or kit-of-parts, from which to find or assemble a resulting element in an appropriate vocabulary dictated by the designer. Human agency is captured most readily by the originating sketch itself but also somewhat easily in the creation of the library of elements from which the **reconstruction function** can use. The descriptor, or attribute space, is a layer of abstraction that stands between the user's input modality (a sketch in this instance) and the element created based on the machines interpretation of desire or intent. This allows for changes of vocabulary (for example, degree of abstraction) without the need for the laborious retraining of the algorithms. Human agency is also somewhat enabled, though requires more technical skill, by allowing a person to write their own code to procedurally reconstitute the descriptor attributes into a new element. Lastly, human agency is also found in the ability for anyone to replicate the process and to retrain the machine learning part to more precisely recognize their own input sketches.

These issues remain as primary concerns:

1. Can the machine be trained to recognize the subtle nuance of a hand sketch?
2. Will interpreting the sketch cause a disambiguation that denies the value of it?
3. Do the benefits outweigh the loss of the tactile experience?
4. Will the process maintain the sense of expressivity and immediacy of sketching?
5. Will this be useful to anyone besides the person who did the initial training?
6. Will the resulting images be sufficiently interesting to warrant the effort?

In summary, we present a complete conceptual design of a system that combines sketch analysis, machine learning and modular output using a virtual reality platform.

We present an interactive system for creating the training data required for the machine learning process. We've created an open source data set of over 5,000 drawing/attribute data samples. We've also created a working prototype of the complete system that illustrates the need for machine learning as well as the potential of such a system. The machine learned model is beyond the scope of this thesis.

In the next chapter we'll look at work that has been done in the past to enhance human-computer interaction and how this relates to our project.

CHAPTER 2

RELATED WORK

Early focus

In 1960, J. C. R. Licklider published a foundational paper entitled, “Man-Computer Symbiosis” [LICKLIDER] and presciently predicted a future age where humans work in cooperative systems with “*thinking*” machines, referring to this as the “*Mechanically Extended Man.*” He posited that, “*the symbiotic partnership will perform intellectual operations much more effectively than man alone.*” In that seminal paper, he lightly touched on the possibility of interacting with the computer via graphics and sketches. Sketch based computer interfaces and sketch-based computer modeling has since been more deeply explored, dating back as far as Ivan Sutherland’s famous “SketchPad: a man-machine graphical communication system”, incredibly as early as 1964. [SUTHERLAND] The literature is extensive and entails a landscape of approaches, goals and subproblems. Sutherland’s pioneering work was most notably followed up by the influential, Computer-Aided Design conferences in Brighton, England during the late 1970’s. More currently, sketch-based interfaces and modeling, referred to as “SBIM,” has its own conference, dating back to 2004, usually in conjunction with Eurographics or Siggraph. Two primary aspects of SBIM have been 1) **sketch recognition** (often a classification problem) and 2) **synthesis** - where the sketch is the input method for a following synthesis procedure to generate a new image [CHEN], 3D art assets[DELANOY] or scene [SHIN]. In some cases, the two are combined. These methods will be explored in the paragraphs below.

Sketch and AI

Recently, attempts are being made to merge sketch input with artificial intelligence in general and more specifically, machine learning [CAKMAK, EITZ YESILBEK]. We see this as a significant development of sketch input and creation. This merger also creates its own set of unique problems. Since sketches are fast and easy to create, they necessarily lack some information and specificity. Most current sketch recognition systems attempt to overcome this lack of information by combining machine learning techniques with a very large training set, usually labelled with categories to support supervised learning methods. Attempting to infer 3D information from a 2D sketch is a particularly challenging but useful subproblem. We avoid this issue by sketching directly in 3D.

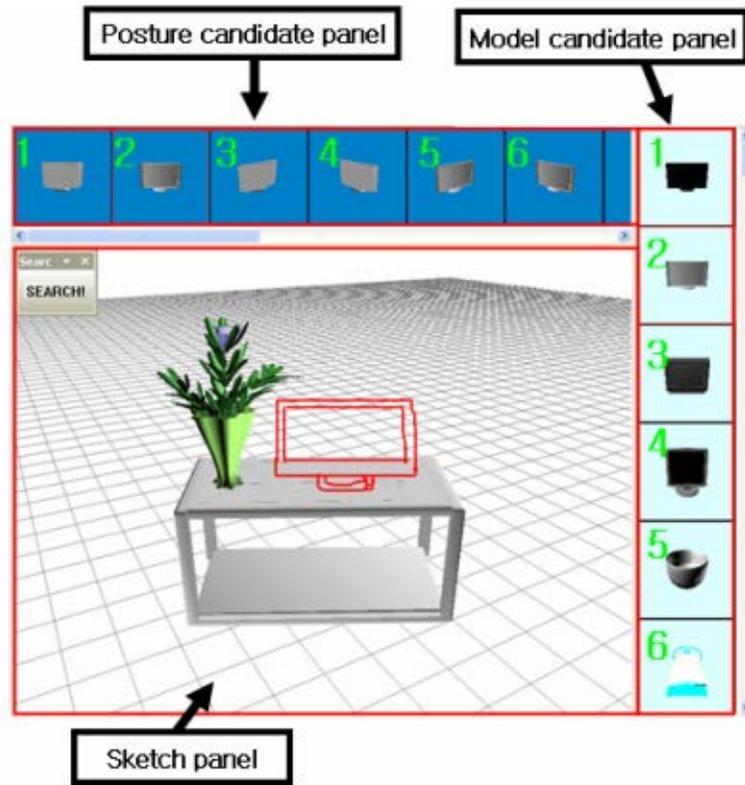
Gesture

Sketch input is most often treated as a static bitmap which lends itself most readily to the canonical machine learning process where features are extracted from a given set of examples. Classifiers are then trained on those features so new sketch inputs can be classified using the trained system. [EITZ] Less common in the literature are examples where the input sketch is based on “gesture.” Gesture and stroke data allows additional information to be preserved such as the stroke sequence, timing, speed and possibly pen pressure. [WOBBROCK] We capture more of the gesture data by using 3D stroke data instead of bitmaps for our machine learning and sketch analysis.

2D ambiguity

More recently, attempts have been made to provide additional information to the quick, but sparse, 2D sketch data by introducing the third dimension. 3D information

can readily disambiguate various shape and position possibilities. This extra level of specificity is very valuable for accurately describing physically tangible forms when that is the desired goal, as is often the case with architecture, sculpture and industrial design. Some of these systems allow designers to work on flat input devices while indicating the 3D location of strokes using various interface modalities [PICCOLOTTO, DORSEY, TOLBA, GUTIERREZ]. “Cuttlefish” [GUTIERREZ] for instance allows a user to specify an arbitrary plane or surface in three-dimensional space, and then to draw strokes on that plane. Others have demonstrated effective methods for using direct 3D sketch input using augmented or virtual reality gear and input devices [ARNOWITZ]. However, to our knowledge, no attempts have been made to combine 3D sketch information with artificial intelligence and machine learning. This combination inspires our efforts to explore ways to enhance early stage design exploration.



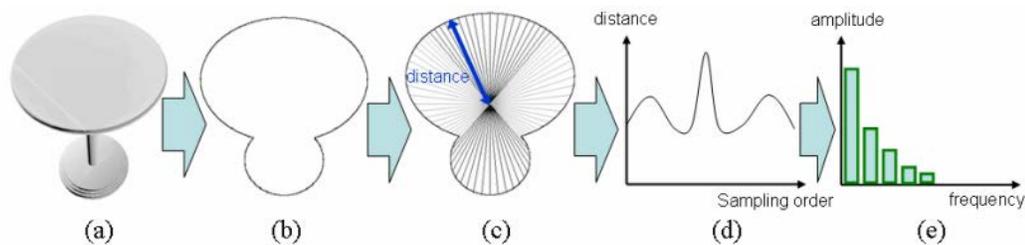
Magic Canvas model retrieval system [SHIN]

Figure 5

Retrieval

Many examples of using sketch input involve the use of the sketch as some form of retrieval or search based upon sketch recognition. (Figure 5) This is particularly true for the design professions. Using sketch for retrieval is a very natural idea to replace complex text descriptions or other interfaces with a simple and intuitive sketch to describe more casually and intuitively what is desired by the user. Many of these techniques use the sketch to index into a library of elements that have been pre-processed and analyzed to make it easier to match the user input sketch to an archetypal form from the library. The retrieved item can be 2D shapes [YESILBEK] or photographs [CHEN] or in some cases a 3D art asset [FUNKHOUSER]. This matching from sketch data to photographs or 3D art assets is often referred to as

“cross-domain classification”. For 3D object retrieval, a few approaches use the contour information of the 3D forms by analyzing the object from many different angles and then matching that to an input sketch. In the project, “magic canvas” by Shin and Igarashi [SHIN] the authors used an interesting technique of sampling the contour of the shape as an array of distances from a centroid. (related to earlier work [WITKIN]) These contour arrays, they call “feature vectors” are created for the object as it appears from 16 different viewing angles with a preference for three-quarter view angles. Each of these sets of contour arrays becomes a fingerprint of sorts for the object. The assumption is that the sketch will likely be a three-quarter contour sketch and can therefore be used as an index by comparing the feature vector generated from the sketch to that of the various objects in the database. (Figure 6)



Magic Canvas contour sampling classification [SHIN]

Figure 6

One subset of sketch recognition involves “cleanup” where a user might draw a crude circle and the machine recognizes this as a circle and converts it to a mathematically precise version [EGGLI]. This is most useful for systems with a limited and fixed vocabulary or alphabet such as electrical circuit figures, flow charts, etc. This is also essentially a form of sketch-based retrieval, where the input sketch is classified to a small set of iconic standard elements or shapes (circles, arrows, squares, etc.). Handwriting recognition is related to this, such that a myriad of possible

sketches can be mapped to an alphabet of standard forms. Besides cleaning up canonical shapes, input sketches can also be used symbolically in place of user interface buttons to invoke actions or perform functions. For example, a scribble gesture might mean “erase.” A good early example of this is the work done by Zelesnick et. al [ZELESNICK] presented at Siggraph in 1996 before the explosion of neural networks. Using only formal logic they interpret 2D sketches based on the way the strokes were made (sequence and direction was factored in) to map to a fixed set of predefined symbols. For example, three perpendicular lines would be transcribed into a cube. This however requires a user to learn the vocabulary defined by the authors and does not interpret random sketches. Along similar lines but using more contemporary methods, in 2017 “Sketch recognition with few examples,” Yesilbek, et. al. employed feature detection, “bagging” and machine learning to map 2D symbols into a constrained domain of about twenty symbols [YESILBEK]. Bagging is a common ensemble technique used to resample a dataset and average the results to decrease instability caused by outliers, and to reduce overfitting.

Effective image retrieval based on sketch input opens up numerous application possibilities. As an example, Lee et. al. [LEE] demonstrated a real time feedback assisted drawing method called “Shadow Draw”. This application dynamically maps user input sketches to a few dozen object categories using a nearest neighbor approach for finding geometrically similar objects. These objects are then faintly displayed under the sketch to gently guide the user towards an improved drawing. Another application is to map from crude amateur sketches to professionally made illustrations [Google *Autodraw*] or to clip art.

In many cases it is deemed important to not just classify the shape of the sketch and target result but to also capture the semantics of the sketch, as in, “what is this a sketch of.” An excellent example of this is “Quick, Draw!” also by Google. This online application (www.quickdraw.withgoogle.com) gives you an assignment to draw something, like an apple, an octagon or a helicopter. In 20 seconds or less it will try to guess what it is that you’ve been tasked with drawing. While this is fascinating, and addictive, it demonstrates an attempt to classify sketches by grasping the semantics of the drawing from the shape. In many cases this very important. (As previously mentioned, for our purposes many of the shapes that a designer will need cannot be named easily or clearly – Figure 2).

Synthesis

Interesting results have been achieved by combining sketch input with a synthesis process to produce new original objects or compositions in either 2D or 3D. In the past, sketching was explored as a potential way to replace or supplement 3D modeling. 3D modeling systems were usually considered difficult to learn and tedious to use, inspiring a search for a simpler, more direct method to create complex 3D forms [EGGLI]. However, since many of these papers were authored, newer approaches to 3D modeling such as Sketchup, Rhino, and Modo (and even new modeling advances in the older more full-featured 3D modeling systems of Maya and 3D Studio Max) has “raised the bar.” These newer systems are easier to learn and allow for very quick modeling of basic 3D forms. An interesting recent effort by Delaney et. al. (2016) uses a deep convolutional neural network to predict occupancy of a 3d voxel grid to generate a 3d form from a 2D input sketch [DELANEY]. Users can change views and

add further information by continuing to sketch and refine or adjust the resulting 3D art asset. It's not yet clear if this approach will eventually allow for a faster, easier or more accurate 3D art asset than using traditional 3D tools.

Some success has been demonstrated by combining retrieval with synthesis to make new forms by combining things instanced from a specific domain of existing elements. One example uses sketches to retrieve 3D objects to allow a user to quickly instantiate new elements into a 3D scene. [SHIN] The ability for non-technical users to quickly assemble a complex 3D scene shows promise as a useful method for early mockups of stage sets, scenes and environments for a variety of applications. Some of the papers described above which use sketch recognition to retrieve objects from fixed domains allow users to synthesize these new forms into new compositions. Similarly, this approach can be applied to two-dimensional image compositions systems with a potential for a more open domain of elements as well. For example, Sketch2Photo [SHIN] and PhotoSketcher [EITZ] combine sketch recognition with an image search on the web to find a matching photo. These "*found*" images are then assembled and iteratively composited together into a new final image.

Machine learning

Recent advances in processing speed and parallelization have enabled artificial intelligence researchers to train machine learning systems on vast amounts of data to obtain good results that were elusive just a few years earlier. This explosion in the use of neural networks (and other approaches to AI) has likewise opened new potential for various artificial intelligent approaches to sketch recognition. In 1959 Hubel and Wiesel introduced a model of a cat's visual system that included two types of cells, a

simple cell and a complex cell. These two types correspond to the two main operators used in today's neural networks - convolution and pooling [HUBEL]. Neural networks based on these and the principles behind Rosenblatt's Perceptron [ROSENBLATT] have become standard. Backpropagation for layered neural networks was introduced originally in 1969 by Bryson and Ho and was rediscovered in the 1980's to usher in the age of "deep learning" or "deep neural networks" (DNN). As previously mentioned, these networks typically require very large data sets to train and therefore significant processing power. As this power has become more readily available after about 2010, this technology has had a significant renaissance. The vision community has demonstrated that DNNs are especially well suited to classification tasks using bitmap images or videos (sequences of bitmap images). Besides neural networks, other related algorithms proven useful for classification include; principal component analysis, k-nearest neighbor, logistic regression, Bayesian nets, decision trees and forests, support vector machines (SVM) and hidden Markov models (HMM). Our data set is designed to support various methods of classification and regression without relying on 2D bitmap information. However, generating the machine learned model from the data set is beyond the scope of this thesis and remains a topic of research.

Future of design

In 2004 at the AI lab at MIT, Adler, et al. strived to define the "Design Studio of the Future" [ADLER]. They wrote, "Sketching is a promising modality around which to build the design studio of the future." We concur and believe that now is the right time to develop the smart tools needed for early stage design exploration of the 21st century. The advent of fast computers and the development of new sophisticated

machine learning algorithms has inspired, and made possible, our own explorations on sketch-based interfaces.

It's clear to us from the literature that there remains much potential for explorations in machine assisted design. In the following chapter we describe a conceptual framework to enhance early stage design with a system we're calling, "V-Sketch." V-Sketch combines 3D sketch input with machine learning and modular libraries to offer designers a new way to explore design possibilities in three dimensions.

CHAPTER 3

DEFINITION OF TERMS

Training Data

The training data is a large data set created to allow machine learning algorithms to be trained to detect a user's intention from a three-dimensional sketch. It is comprised of a set of 3D drawings and their associated descriptions or attributes. This is the critical requirement to produce a model that can be used by the V-Sketch application to interpret a user's input sketch.

The Model

The model is a system that can interpret user input sketches into a set of attributes. The model is derived from machine learning algorithms trained on the drawings and associated attributes, the training data, to interpret the user's intention from an input sketch.

Data Generation Tool

The data generation tool is a program created in the Unreal Engine to run on the Oculus Rift in VR. It's designed to facilitate the creation of the set of training data required for machine learning. This program includes a 3D drawing interface and provides guides to manage the data creation and to save out the 3D drawing combined with the attributes that best describe the drawing. This program also manages the distribution of drawing types in the data set.

Data Visualizer

The Data Visualizer is a VR application that can read a folder of drawing files and reconstruct the 3D drawings for viewing. It also provides visual feedback indicating

the attributes that describe the current drawing which is included in the drawing's data record.

V-Sketch

V-Sketch is the name of the ultimate target application. It is a tool that allows users to sketch and then automatically converts their sketches into 3D forms to create a composition. This application has a sketching interface and employs the model to interpret the user's intention from their input sketch. V-Sketch also utilizes modular "libraries" to constrain the search for the best form to a limited set of predetermined possibilities.

Composition or Design

The composition or design is the overall 3D assemblage of forms that represents the work that the user is creating. It is comprised of one or more "elements." For example, a composition might be of a building composed of many "elements".

Elements

Elements are the 3D objects that make up any given composition or design. Each element is generated automatically from the user's input sketch. The user's intention is interpreted with the aid of the model. The intention is represented by the resulting attribute settings. These attributes are used by the **reconstruction function** to produce an element. A stone wall for example might be an element in a composition.

Sketch

A sketch is 3D line data created in VR by a user to communicate an intended type of element for the current composition. A sketch is comprised of one or more strokes. A sketch might be a collection of strokes that indicate a column for example.

Stroke

A stroke is a single continuous line or data point and serves as a component of a sketch. Strokes can be free hand lines, straight lines or single data points. Strokes are sampled at 0.02 second intervals to create a sequence of points with (x,y,z) coordinates in space. A collection of one or more strokes comprise a sketch.

Attributes (Labels, Parameters)

Attributes are the numerical and Boolean data that describes the intent of a given sketch. Often in machine learning literature, in the context of the training data set these are called “labels” or “tags.” In the context of the **reconstruction function** of V-Sketch these attributes are passed in as “parameters” and form the input to the function tasked with returning a new element of the composition. Our attributes come in two flavors, “space” and “tags.”

Attribute Space

The Attribute space is one portion of the attributes used to describe the intention or characteristic of a sketch. We have three real number attributes that indicate a point in a three-dimensional space used to describe the users intention from a given sketch. These dimensions are complexity, curvature and dimension. Complexity ranges from a simple to complex. Curvature ranges from rectilinear to curvilinear. Dimension ranges from zero dimensions to three dimensions, i.e. points (0D), lines (1D), planes (2D) to volumes (3D).

Attribute Tags

Attribute tags are Boolean flags that provide more information about the sketch. They fall into two categories. Some tags indicate that the sketch is of something, such

as a sphere or a tetrahedron, these all begin with “is a ...” Other tags describe aspects of the sketch itself and are of the form, “has ...” For example, “has rectangles” and “has ovals” etc.

Exemplars

Exemplars are 3D art assets and their associated attributes, used as guides in the Drawing Generator to present to the data set creator a 3D example of a given set of attributes. The creator then uses these forms to make drawings that are reflective of the attributes. Exemplars have been carefully calibrated to span the full range of available attributes.

Sketch Features

The goal of the machine learning algorithms is to convert a given input sketch (sequence of (x,y,z) coordinates) into an array of attributes. This can be enhanced by first converting the raw point data into more actionable aspects of the sketch by calculating some salient features of it. A set of features is extracted from the input sketch using a combination of basic data analysis and machine learning models developed for individual strokes of a sketch. Feature extraction is a common aspect of machine learning. Features might include things like, number of strokes, average length of strokes, total length of all strokes, etc. Sketch features also include a set of line features for each stroke of the sketch, these are called, stroke features.

Stroke Features

Like sketch features, stroke features are numerical values that can be extracted and used as input for a trained model. Examples include, stroke length, number of points, mean and standard deviation, mode, etc. Also, these features can be used with stroke

specific machine learning models to further extract information at the single stroke level, such as complexity and degree of curvature and dimension, the results of which are aggregated to provide additional sketch features as an input vector for the model.

Library (or “kit-of-parts”)

V-Sketch is a modular approach to design where the user (or an expert user) can create collections of 3D design elements which have been manually labelled with attributes that describe the element. For example, a collection of glass walls and forms, or a collection of general colored massing shapes. Individual libraries function as a kit-of-parts. V-Sketch allows an end user to easily switch between libraries, while creating a composition, to constrain the interpretation of a given input sketch. A selection of the element(s) from the library that best match the sketch attributes is performed by the reconstruction function.

Reconstruction Function

The reconstruction function takes as input the attributes that the model generated from the input sketch. The reconstruction function produces a new 3D element of the design based on these values and the currently chosen library. The simplest form of a reconstruction function would be to use K-Nearest neighbors to find the elements in the library which have attributes most similar to those extracted from the sketch by the model. Alternatively, a reconstruction function can procedurally generate an element instead of using a library, or some hybrid approach. The V-Sketch interface allows the user to cycle through the few elements which best match the sketch attributes. Once an element has been chosen or created by the reconstruction function it needs to be placed and oriented in the composition - this is the situating process.

Situating Process

This process places the new element chosen by the **reconstruction function** into the composition. It has access to the derived attributes and the raw data of the input sketch and attempts to match the placement, scale and orientation indicated by the input sketch.

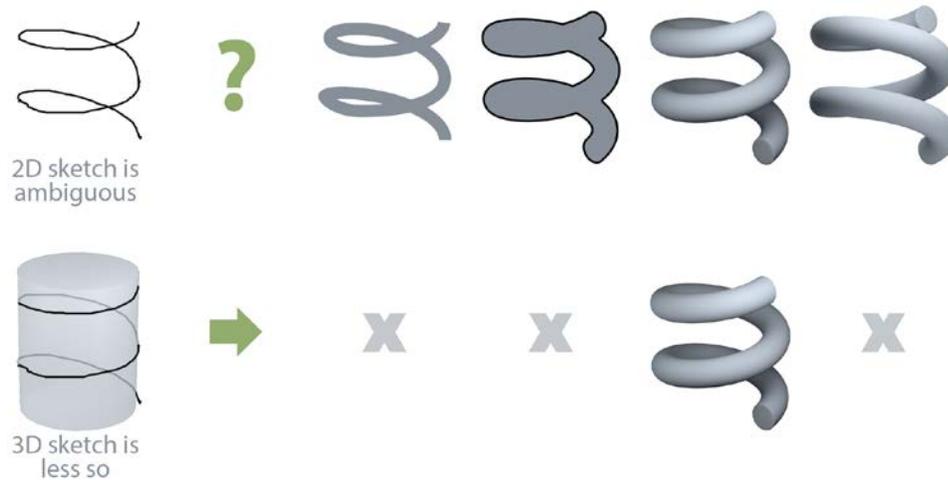
CHAPTER 4

MACHINE ASSISTED DESIGN EXPLORATION



Overall approach

We introduce the V-Sketch system. (Figure 7) This conceptual framework describes a process for combining sketch input, machine learning and flexible output to enhance early stage design exploration. Our V-Sketch system utilizes virtual reality gear such as the Oculus Rift or HTC Vive to enable a designer to quickly create design compositions in three dimensions. Instead of the more typical method of assembling elements from a table of model assets (trees, walls, windows, shapes, etc.) the user interacts directly with the machine using sketch input in three space. Sketching in three dimensions allows a designer to disambiguate a drawing and to more precisely communicate to the machine what type of form they are interested in. (Figure 8) Also, as with the situating process, many things are simplified in a full 3D VR version. In 2D, the exact placement of an object in 3D is ambiguous, making the situating process require more user input and adding significant complexity.

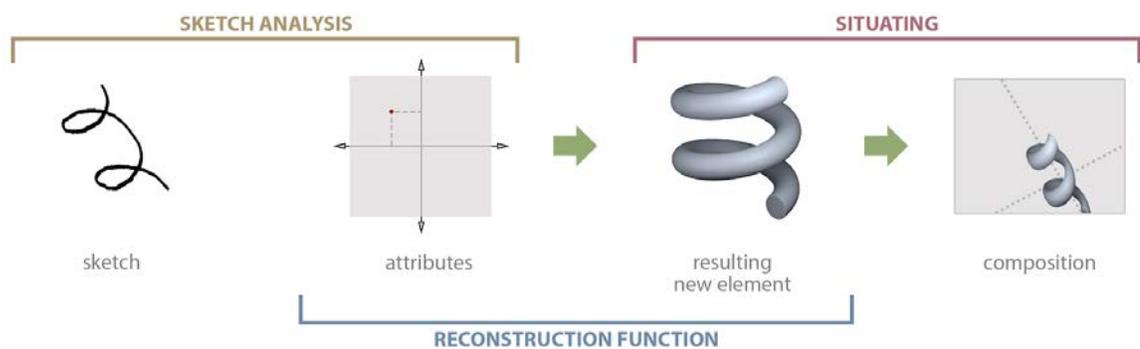


2D sketching of 3D objects is inherently ambiguous but not so in 3D Figure 8

However, it is possible, and may, in some cases, be more desirable, to create 3D sketches without the need for VR or AR gear. This can potentially be accomplished with tools such as the 3D Bezier tool in Maya or some yet to be made available tool along the lines of the research described above [PICCOLOTTO, DORSEY, TOLBA, GUTIERREZ]. These systems allow a user to sketch using a flat surface interface but produce a sketch with three-dimensional information. Whether using systems like these or sketching directly in three-space using VR gear, the additional information gained by adding the third dimension is transformative. Important things such as axis alignment, perpendicularity, and curvature in space is all made more apparent in 3D. We have implemented a VR prototype version of our system to explore this possibility further.

In our system, the user creates a composition by sketching each new element of the composition directly in the context of the existing composition. A composition is made up of elements which have been instantiated through human-machine collaboration via the input sketch. When the designer has completed a sketch, he/she

pulls a trigger on the controller to cause the machine to analyze the sketch and replace it with a new element of the design. Pulling the trigger again repeats the process and replaces the element with a different element based on the same sketch. A collection of similar elements can be cycled allowing the designer to choose from these. One of the options offered to the user is their own originating sketch. Once chosen the new element remains selected and allows the user to adjust its position and scale as desired, or to delete it and try again. The process is repeated to create an assemblage of elements that constitute the composition. This is described more thoroughly in the section titled, “V-Sketch Prototype”.



The primary components of the V-Sketch system

Figure 9

Core Concept

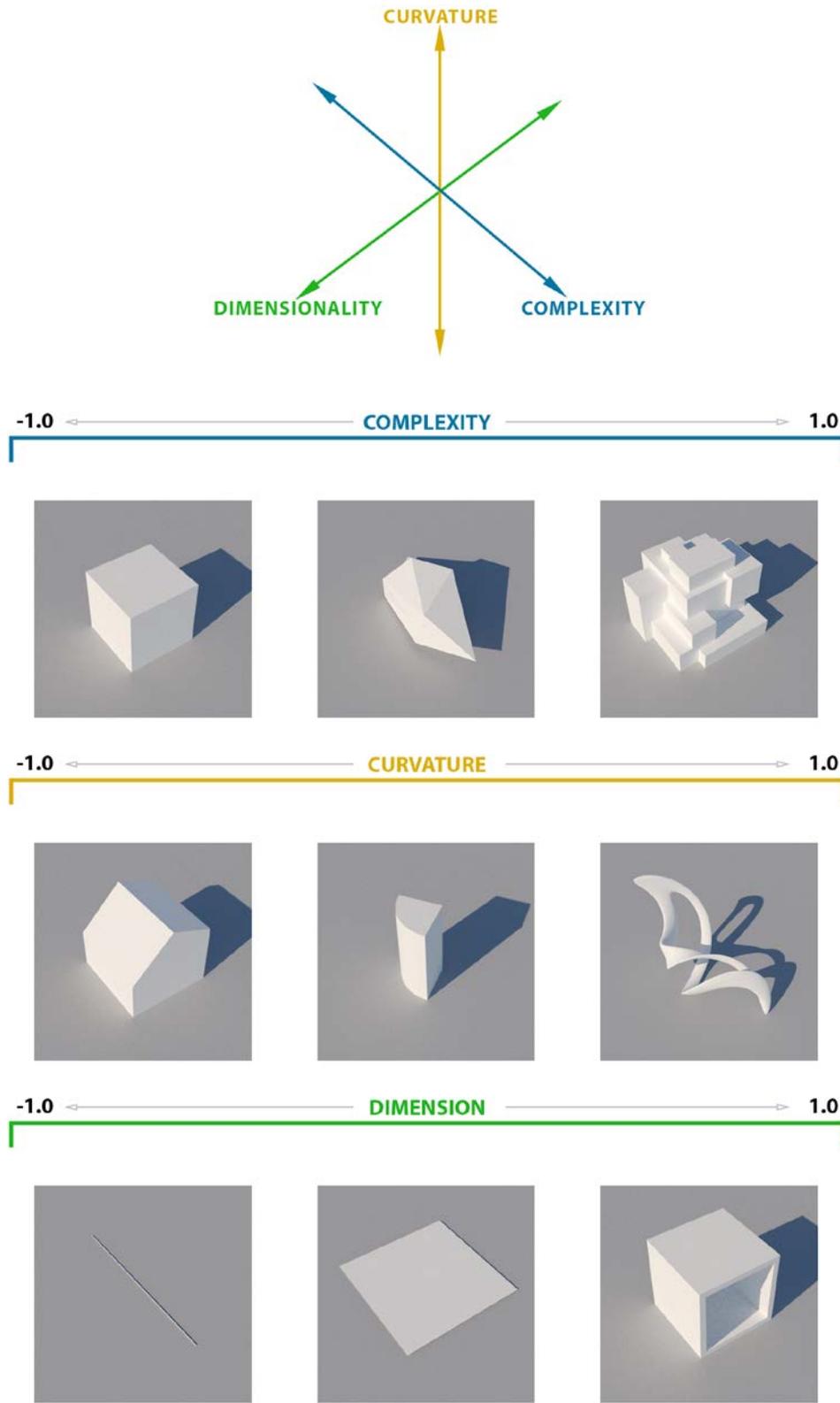
The overall approach used for the V-Sketch project is shown in Figure 9. There are three primary modules of the system. The **sketch analysis** part has the task of creating the attribute array that describes the desired results implied by the input sketch. The **reconstruction function** flexibly interprets that analysis and produces a 3D graphical element. These modules are connected by an abstract, multidimensional numerical space - the “*attributes.*”

The resulting 3D graphical element must then be positioned into the composition to best match the placement, scale and orientation of the input sketch. We call this, the **situating** process. Then the cycle repeats.

A composition is made by the user as a collection of sketches of various components or parts of the composition, an “element”. Each sketch is comprised of one or more strokes. These sketches are made by the end user to quickly and fluidly communicate to the machine what *type* of element he/she desires for the indicated position of the composition. The software captures the physical gesture of the designer’s sketches, stroke by stroke. We believe that valuable information is contained in the dynamic creation of a sketch, such as the stroke sequencing, direction and speed when making a sketch and therefore employs a process that retains this information as opposed to using a bitmap. Using a bitmap would enable the use of many proven techniques common to computer vision applications but would eliminate the availability of the sketch dynamics. In addition, capturing and working from the sketch data instead of a bitmap of the resulting sketch more easily allows for working with three dimensional sketches.

Before the primary model is employed, the sketch data is pre-processed and analyzed using a suite of statistical data analysis tools and artificial intelligence techniques to best capture the abstract qualities of the sketch. This common approach, often called, “*feature engineering*,” provides a useful “bag-of-features.” The bag-of-features is combined with a machine learned model to convert the input sketch data to an attribute vector or point in a multi-dimensional attribute space used to numerically describe the inferred intent of the sketch.

The foundation for this process is the *training data set*, which is the large data set required to allow machine learning algorithms to be trained to detect a user's intention from a three-dimensional *sketch*. It is comprised of a set of 3D drawings and their associated descriptions or *attributes*. This is the essential requirement to produce a model that can be used by the V-Sketch application to interpret a user's input sketch. This data set reflects the combination of 3D drawings and their associated attributes. The three primary attributes used to describe the sketch are **complexity**, **curvature** and **dimensionality**. (Figure 10)



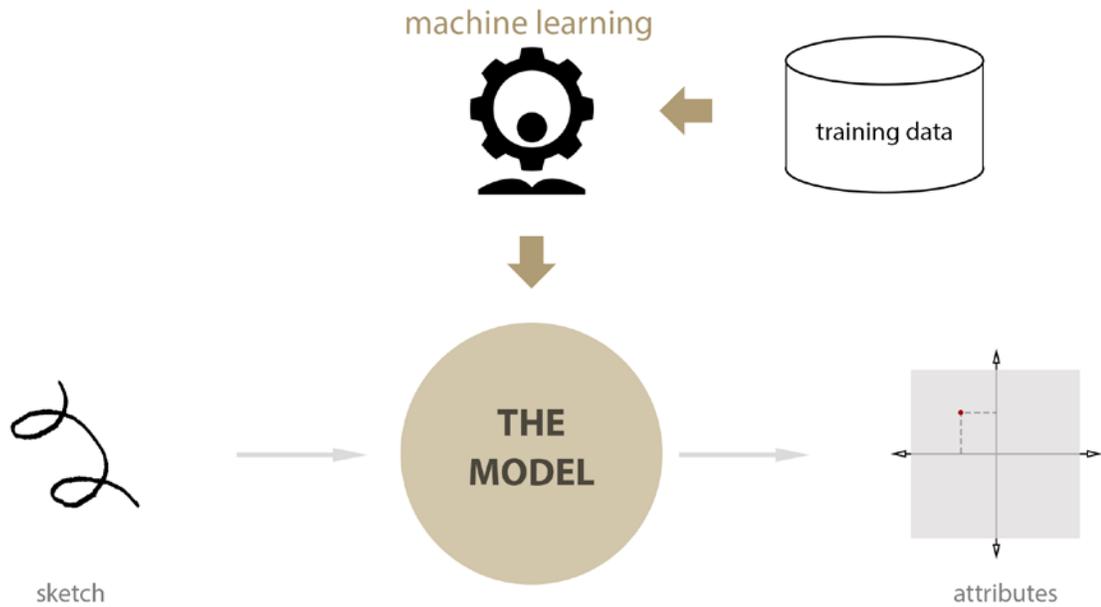
The dimensions of the attribute space

Figure 10

Complexity is a value that captures the visual or geometric complexity of the sketched form. Curvature is the degree to which the overall nature of the desired form is curvilinear versus rectilinear. Dimensionality is the degree to which the sketched form is extended through space from a one-dimensional point, to an elongated form as a line, to a planar form, and ultimately to an evenly dimensioned 3D shape. We call these three values the “*attribute space*.” In addition to the attribute space comprised of three continuous real numbers we utilize a set of Boolean flags that can be set to be true or false depending on the drawing. We call these the “*attribute tags*.” These tags denote specific things about the drawing such as what it is (a sphere, a tetrahedron, etc.) and what qualities it possesses (e.g. is upright, has right angles, etc.) These attributes are described more fully in the section titled, “Training Data Set.”

Once the machine has analyzed the sketch and converted its qualities to a numerical attribute vector as a type of description, the second primary module, the **reconstruction function**, is invoked. The **reconstruction function** is a suite of functions, which can convert the input attribute vector into a 3D object or element. Lastly this new element is “*situated*,” which means it is scaled and positioned to approximate the sketch. The process is repeated element by element as the composition unfolds.

Let’s now look more closely at the **sketch analysis** phase where the input sketch is converted into an attribute vector.



Sketch analysis using a machine learned model

Figure 11

Sketch Analysis

Clearly, it is possible to derive much useful information from an input sketch using common data analysis techniques. However, we believe that sophisticated machine learning algorithms will be required to best facilitate smooth and intuitive human-computer interaction via sketching. (Figure 11) The essential first step for a supervised learning approach to sketch analysis is to create a large data set that correlates a collection of hand drawn 3D drawings with their associated attribute values. This data is used to train algorithms to generate a model that can take any user input sketch and predict with reasonable accuracy what the attributes should be for that sketch. It is the recognition of the need for robust machine-learning trained models that motivates this thesis project to create a satisfactory training data set. Although implementation of these machine learning algorithms is beyond the scope of this thesis we include a description here of the conceptual framework for creating such a system.

Machine Learning

There are several unique machine learning challenges in this context. Most machine learning algorithms require a fixed set of features for all data samples. (samples being a features/attributes pairing) In our case there are a variable number of strokes per sketch and each stroke is comprised of a variable number of coordinates. One approach to solving this utilizes a class of algorithms called “Long Short-Term Memory” units or more commonly LSTMs. This type of Recurrent Neural Network derived algorithm has demonstrated effectiveness for streaming data such as that used in speech recognition. This approach may also be effective for the live drawing of sketch data. Alternatively, by studying the attributes that we desire to extract from the sketch, we may need to develop a hybrid approach where different algorithms are used to extract different attributes. In our case, the desired attributes fall into two main categories, a **spatial component**, which is comprised of three real numbers and therefore positions each sketch in a qualitative space of three dimensions. We call this the “*attribute space*.” There is also a set of **Boolean flags**, some of which are mutually exclusive and represent more of a standard classification problem. We refer to these as the “*attribute tags*.” Considering the nature of these attributes, described in more detail below in the section titled “*Training Data Set*,” we recommend the following approach.

1. Break each sketch into its component strokes
2. Prepare the sketch data by analyzing each stroke using both traditional statistical analysis as well as machine learned models trained on single stroke data

3. Create a collection of features from the above analysis to create a fixed length feature vector as input to regression algorithms trained with supervised learning on the same features extracted from the training set data samples.

Preparing sketch data

Many machine learning algorithms generate models that utilize a bitmap as a vector of values and detect patterns and determine categories which would otherwise be difficult to ascertain using just formal logic. In our case, we recommend avoiding, or at least not limiting to only the use of a bitmap representation of the sketch. As mentioned in the overview, we believe that working directly with line data will be more challenging up front but will ultimately lead to more flexibility and expressiveness. Bitmaps do not retain the timing involved in creating a sketch, the sequence and other physical aspects of the act of sketching, as well as the three-dimensional nature of the sketch. Our raw data consists of CSV (comma separated values) files with the first number is an index into the exemplar array used for creating the sample which is explained more in the chapter on creating the training data set. This is used for debugging and can be ignored. The next three entries are the three spatial attributes for the included sketch. A string of ones and zeros follow those values to indicate true/false for each of the Boolean attribute tags. The stroke point data, sampled at regular time intervals, follows those values with each stroke separated by a null value. Performing basic numerical analysis of our training data we were able to determine that most sketches have fewer than twelve strokes per sketch and most strokes had fewer than twenty data points per stroke.

A method of preparing data features should provide best results for machine learning algorithms. The concept is to fix the number of strokes and instead of raw x,y,z coordinates, generate a “bag-of-features” to improve feature recognition. We recommend dividing the problem into sub-parts to create a hierarchy of analyses. It’s possible to use the subset of the training dataset that includes only drawing samples comprised of just single strokes to extract stroke specific features. The following per stroke values can be extracted using standard data science techniques.

- Number points
- Stroke length
- Stroke duration (total time)
- Start point
- End point
- Distance between endpoints
- Direction trend (averaged direction vector)
- Dominant angle (rounded to nearest 5 degree)
- Speed variation
- Angle variability
- Number straight line segments detected
- Circle detected
- Dot detected (single point stroke)
- Vertical line segment detected
- Horizontal line segment detected
- Axis aligned angles detected

Then for the sketch we added the following features:

- Number of strokes
- Mean stroke length
- Mean stroke duration

- Total length
- Total time duration
- Start point first stroke
- End point last stroke
- Distance between first point and last point
- Direction trend (averaged direction vector)
- Dominant angle (rounded to nearest 5 degree)
- Speed variation
- Angle variability
- Number straight line segments detected
- Number circles detected
- Number dots detected (single point data)
- Vertical lines detected
- Horizontal lines detected
- Axis line angles detected
- Attribute values averages for each of its strokes

It is also possible to simplify the target values by turning these into binary classification problems. For instance, instead of evaluating curvature as floating-point values it may be desirable to bucket these into two binary settings of, “is curvilinear”, and, “is rectilinear.” Using a “random forests” method (an aggregation of decision trees) it is possible to reconstruct the smooth variability of the curvature variable. The attribute evaluations for each single stroke can be averaged and provided as a feature for the overall sketch analysis. This hierarchical and simplified approach using binary classification as separate steps is ongoing research.

Our experiments have indicated that better results are achieved by breaking down the problem into a hierarchical or nested set of subproblems and solving things individually instead of attempting to analyze the sketch as a whole in one step. The

approach described above, and our early experiments suggest that aggregated linear regressions with “bagging” and/or “boosting” is most promising for extracting the sketch attributes needed as input to the **reconstruction function**.

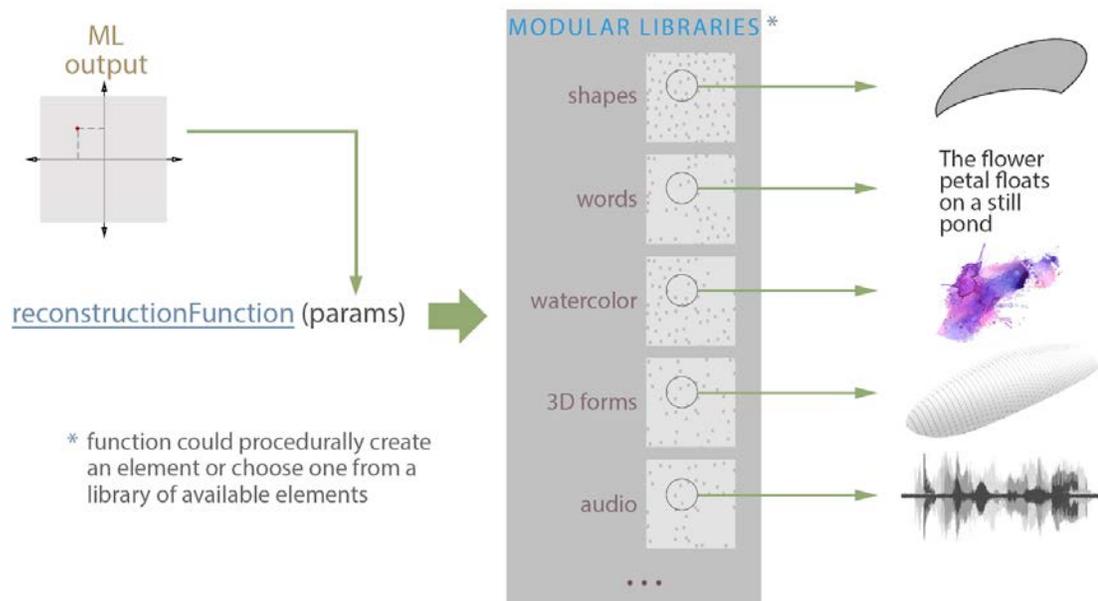
Reconstruction function

The **reconstruction function** is the second primary module of the V-Sketch system and it is sequenced directly following the sketch analysis module. As described in the “Overall Approach” section, the **reconstruction function** takes as input an array of attributes derived from the sketch by the model and returns a new element of the composition. The flexibility of this approach allows a technical artist to write code to create a new **reconstruction function** to suit his/her needs. These functions could literally be anything that takes attributes as input and returns an element of a composition. Perhaps the simplest and most powerful form would be to use a system of interchangeable libraries of predefined elements.

Libraries

At its most basic form, a **reconstruction function** can be created that searches a given library or kit-of-parts for whichever element best matches the input attributes. The architect Le Corbusier’s extolled the virtue of producing complex architecture through a use of a set of “preliminary shapes” or simple forms. Our system facilitates the use of custom sets of forms or more elaborate textured detailed shapes, whatever the designer requires at the time. These modular libraries are flexible and allow a designer to quickly switch between collections of elements while making a composition to greatly change the domain from which new elements are chosen. One example might be a library consisting of simple massing forms with transparency

while another might contain forms with specific materials like wood or concrete with photorealistic texturing. In either case an element is chosen from that constrained set which best represents the input attributes. This approach requires that the predefined library elements must be evaluated and positioned in the attribute space and labelled with attribute tags. This is described in the next subsection. It's interesting to note that these libraries which are mostly architectonic in our system could be anything, including audio clips, animating forms, 2D, etc. While a technical artist could create a custom **reconstruction function**, most artists can create and label a collection of elements to create a new library. (Figure 12)



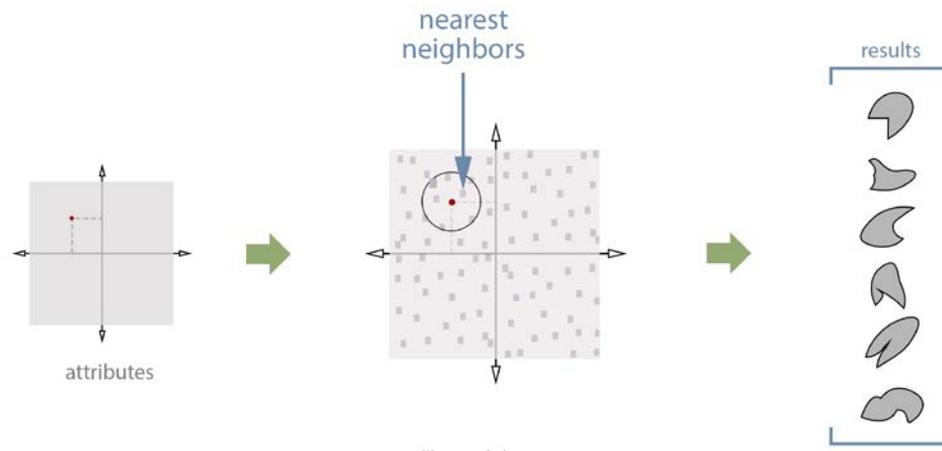
Attributes are input parameters to the reconstruction function Figure 12

Using interchangeable libraries allows a designer to create an entire vocabulary by populating libraries with assets. Imagine if you will a vocabulary for sand castles, Richard Meier buildings, Antonio Gaudi elements, or gothic cathedrals. More useful

for our purpose are low level early stage design masses of simple forms and voids, planes and lines to indicate architectural thinking. However, since our sketches are retained it is possible to switch vocabularies and see the results as the designer moves from one level of abstraction to another.

When using a library, a **reconstruction function** can sort all elements in a library by the square of the distance to the target attribute space parameters and/or restrict the search to only elements with matching attribute tags. From this, a subset array of possible choices is created and presented to the user to make the final selection.

(Figure 13) The simplicity and potential of this cannot be overstated. Without writing any code a designer can create a unique collection of element groups and readily switch between them. Since everything is done in context, they can then both select and position the elements directly into a composition without having to access a table or list.



Top shape matches are offered to the user to select from Figure 13

Procedural generation

Another possibility for a **reconstruction function** is one where the elements are made programmatically. This is where an algorithm is crafted to produce a new 3D

object directly from the input parameters. A simple variation might be to start with a base mesh of either a sphere or a cube and then to apply various transformations based on the incoming parameters. The advantage of this is that the space can be made to be continuous and thereby constitute a near infinite variety of forms that define the domain.



Joseph Cornell, "Untitled (Hotel Eden)"

Figure 14

Creating a tagged library

Besides sketching itself, the primary channel of human agency that we have introduced with this project is the creation of custom libraries and vocabularies to match the desired degree of abstraction or style for a given project. Many art and design projects begin with a definition of the range of elements possible for a given project. The collage artist Joseph Cornell (Figure 14), like other collage artists maintained an elaborate and extensive collection of raw elements for his work. The process for a collage artist entails collecting possible elements and then selecting a

subset, often one element at a time and temporarily juxtaposing the elements in search of the proper contrast and balance of forms. Our system reflects that process by emphasizing the use of a flexible system of libraries of elements. Creating a library or kit-of-parts is neither trivial nor difficult. Digital production artists can create raw assets quickly. To this end, a library labeling system can be created that reads in each new element from a list of files and presents the user with a UI to tag each element and insert it into an internal database in the V-Sketch system. At the very least, the element is presented along with sliders to allow the user to manually set the qualities that describe it. There's one slider for each dimension of the core attribute space and a set of check boxes to allow the user to set the attribute tags as well. Better, is a system that offers the sliders but also allows the user to create a 3D drawing to match the element and automatically set the attribute values by submitting that drawing through the same analysis and machine learned model that will be used at creation time.

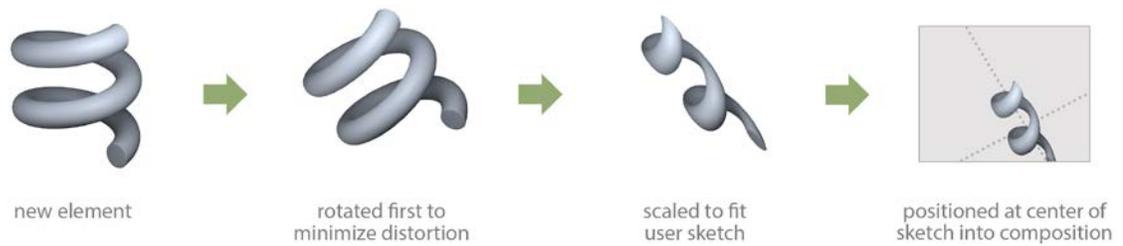
In addition to setting the attributes for each element of a library the user must be able to specify any constraints on the element. A collection of switches can be provided that allow the user to constrain the re-orientation and scaling of the element. (for example, enforce that this column remains upright, or that a sphere must be scaled evenly to remain a sphere)

It takes about 125 evenly spaced examples to define a three-attribute space (5x5x5). Sparse spaces are acceptable but will have voids that cannot accurately match some sketches. A library's attribute space can be more densely populated in some areas than others. We've used visualization tools to get a better sense for this coverage. A good healthy three-dimensional library should ideally have about 200 elements but anything

more than about eighty seems to provide reasonable performance. Small libraries require less sophisticated sketch analysis because the distance in terms of number of library elements is necessarily closer to the ideal element than it may be in large libraries. In some cases, small focused libraries might be what the designer requires to offer a more limited vocabulary. It is also evident that there will be some library elements that very rarely if ever get chosen by the **reconstruction function** and conversely there will be some elements that get chosen often. We've employed a simple process called "Library Tuning" to address this.

Library Tuning

Although creating the machine trained model to fully analyze input sketches is beyond the scope of this thesis, we did utilize the data set to tune our test libraries. Noticing that some elements were too common while others were too rare we looked for ways to better balance the system. One approach that worked quite well was to run the data set through our simplified sketch analysis process and sort and score by distance each element in the library. At each epoch, or full run through all the elements in the data set, the attributes of the library elements that had the lowest score are "nudged" in the direction of the elements that received the highest score. This is a reasonable approach since the original labelling of the library elements is subjective and the desired end goal is a good user experience.



The situating process

Figure 15

Situating

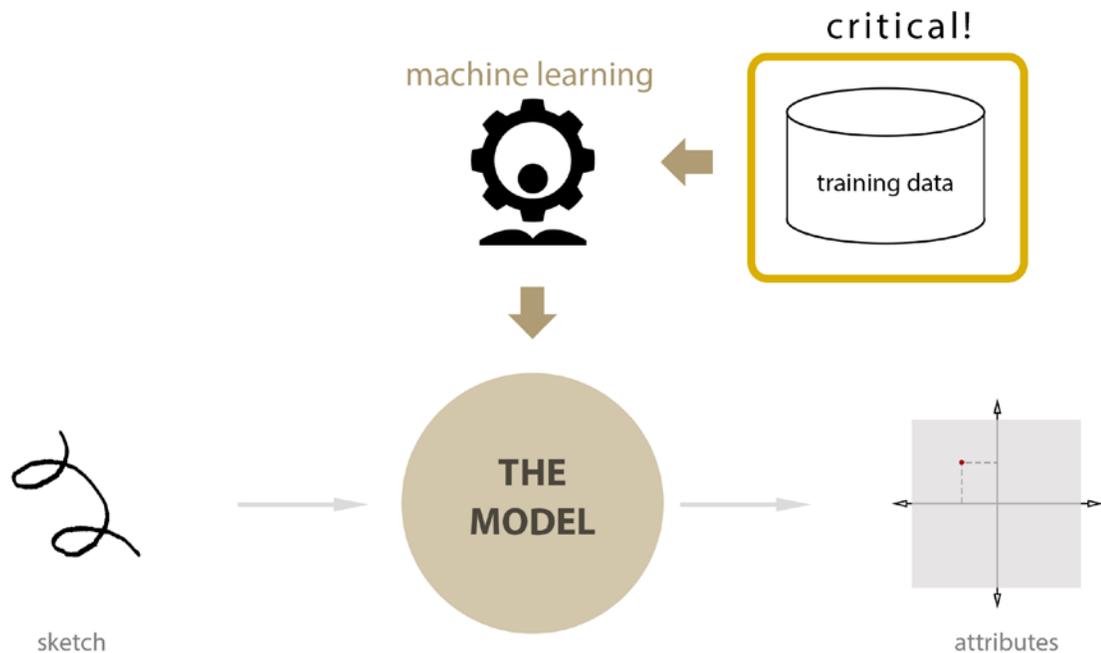
After the sketch has been processed and an element has been retrieved and/or created, the element must be placed and oriented into the three-dimensional scene. As mentioned, this is the “**situating** process”. (Figure 15) In our earlier 2D version of V-Sketch, situating was difficult due to the ambiguous 3D position indicated by a 2D sketch. In our current 3D VR version of V-Sketch the ambiguity, and hence much of the difficulty, is naturally eliminated. The situating process is to fit the resulting object with reasonable error inside the bounding box defined by the 3D sketch itself - position and scale. Although, much simpler in 3D than 2D there are some challenges that need to be accommodated. Some objects can be scaled non-uniformly to better fit the input sketch. For example, a basic box can be scaled to precisely fit the sketch. However, this means that for freely scalable elements the dimensionality attribute should perhaps not be a factor when choosing the best element from a library in the **reconstruction function**. Also, while most abstract forms can be scaled non-uniformly, it is optimal if the form is rotated to best align with the sketch dimensions before scaling. This ensures that the nature of the shape is preserved. For example, a tall thin cone should be aligned to a conical sketch such that the longest dimensions are the same before scaling to fit. Otherwise a sketch of a tall cone would result in a distorted cone element. It’s one thing to recognize that a sketch is of a cone and

another to recognize which end is the pointed end and to re-orient the element to best fit the sketch. Another good example of this is with planar forms. The library element of a planar form might lie along the x, y plane but the user may sketch a matching plane in the x, z plane. In this case the scaling to fit must rotate the plane along the x axis and then scale to fit the sketch. This re-scaling might suggest that the dimensionality attribute is less important or, as previously mentioned, non-orthogonal to the other dimensions of complexity and curvature. This is true to some degree; however, lines can only be scaled in one dimension and planes in two. This fact keeps the dimensionality attribute valuable as some forms are specific in their dimension. For example, a low-relief sculpture like those of the baptistry door in Florence derive in part their nature from the dimensionality that lies somewhere between a plane and a solid. Similarly, long thin forms start to appear more like lines.

As shown, the situating process may involve rotations and non-uniform scaling needed to best match a new element to the input sketch. However, the element that's been instantiated may have constraints that have been placed on it when the element was added to the library or created procedurally. These constraints are sometimes necessary for certain elements, for instance, staircases, trees and colonnades should by default remain upright. The setting of these constraints at authoring time is described in more detail in the section on creating a tagged library.

CHAPTER 5

TRAINING DATA SET



Training data set is critical for machine learning

Figure 16

Goals

The desired goal of using sketch input to collaborate with intelligent software to enhance the design process will require the ability for the machine to analyze the users input sketch with robustness and accuracy. An algorithmic **model** must be developed that can interpret the input sketch and produce an array of attributes describing the user's intention. The critical component is the training data set that pairs sample 3D drawings with accurate attribute descriptors to be used to train machine learning algorithms producing such a model. (figure 16) A good data set will contain a wide range of exemplary drawings that have representative samples of every reasonable combination of drawing types defined by the attribute descriptors. We have generated this data set. It is important to note that all the sample drawings in our data set were

done by a single individual and as such an inherent bias is unavoidable. However, despite this limitation, much effort was made to generalize the drawings as much as possible. This is described more fully in the subsection on the data set creation process. Lastly, the most valuable part of the data set is the 3D drawings themselves, the creation of which is arduous and requires direct human creation at significant time cost. It is not so difficult to manually or algorithmically re-label the drawings to suit any future unforeseen needs.

Attributes

As briefly described previously, the set of numerical attributes that are used as an abstract representation of abstract forms is the linchpin of the system. There is no established clear method for describing objectively a large array of abstract forms. Gestalt psychology principles are marginally useful as are other personal systems put forth by various artists such as Jean Arp and others. With an understanding of the necessity for subjectivity in choosing the attributes themselves, it is understood that this attempt to do so is not likely to be definitive but rather serves as a solid foundation for others to build upon or modify to suit their own needs.

As mentioned already our attributes are of two types, the attribute space comprised of three real numbers and the attribute tags which is a collection of Boolean true/false variables. The attribute space is most easily grasped by reviewing Figure 10. The three independent axis variables are complexity, curvature and dimensionality. These three continuous variables have been shown in our work to facilitate an effective high-level categorization of any abstract 3D form. The degree of complexity and curvature (as opposed to rectilinear) are readily understood. Dimensionality on the other hand has

proven essential but problematic. Dimensionality is our attempt to divide forms by their extension in space. A very low value would indicate a point in space, while a line and then a plane and low-relief and on to a fully evenly dimensioned solid describe the dimensionality vector. Having this third dimension is critical primarily for separating planar elements from fully 3D elements and the degrees between. While useful there are problems with this variable. For one, it is not completely orthogonal to the other two axes. Imagine a curved line and its bounding box. Whereas a straight line can have only one dimension of length, a curve needs at least two dimensions to bound its area. It lies on a single plane or can potentially even be fully three dimensional. Therefore, in our system curvilinear sketches have a minimum bound on the dimensionality axis. Also, a planar element may be complex, but it is inherently not as complex as a complex 3D form. The dimensionality variable is useful but is not entirely independent of the other two axes. Another practical problem with this variable is that many forms can be scaled non-uniformly to fit a wider range of values in our attribute dimensionality axis. This means that the **reconstruction function** may select an element based on its dimensionality instead of another more appropriate element which can be scaled non-uniformly to match exactly the dimensionality attribute of the sketch. This can be illustrated easily with a conic form. If the point of the cone is near the circular base it approaches a plane (and becomes more curvilinear). As the height increases and the base decreases in diameter it approaches a line (and becomes less curvilinear, relatively). While if the diameter matches the height, it becomes an evenly dimensioned 3D form. In our system these would be differentiated forms, suggesting that while our language might refer to these all as

“cones” they are different abstract forms. This is perhaps a limitation of our language. We can avoid this issue by providing multiple “cones” with different dimensionality. We can link them linguistically by setting the appropriate Boolean tags indicating that all of these are still, “cones.”

A set of Boolean tags to further describe the intended shape drawn were chosen and refined over several iterations. This set of attribute tags reflects an architectonic focus but should also be sufficient for general abstract purposes.

Beyond the three real number continuous attributes there are also Boolean tags:

- Is a box (all right angles)
- Is a sphere
- Is a cylinder
- Is a cone
- Is a tetrahedron
- Is a pyramid
- Is an octahedron
- Is a line
- Is a plane
- Is a wall
- Is an arch
- Is a column
- Is [are] stairs
- Is a shell (thin curved shape)
- Is a tube
- Is organic
- Is irregular (neither curved nor straight lines - like a coastline)
- Is upright (usually meaning it has vertical edges)
- Is horizontal (lying flat parallel to the ground plane)
- Is a scribble

- Is a spiral
- Is a shard (fragmentary shape)
- Is an arrow
- Is foliage
- Is a lattice (3D grid or trellis)

Feature labels can be used if the feature is a significant characteristic of the form:

- Has rectangles
- Has ovals (including circles)
- Has triangles (as an important feature)
- Has a skew
- Has a wedge (like a roof or a ramp)
- Has hexagonal angles (120 or 60 degrees)
- Has right angles
- Has six sides
- Has a grid
- Has symmetry (as an important feature)
- Has stipple (dots)
- Has hatching (several similar strokes)
- Has some horizontals (as an important feature)
- Has some verticals (as an important feature)

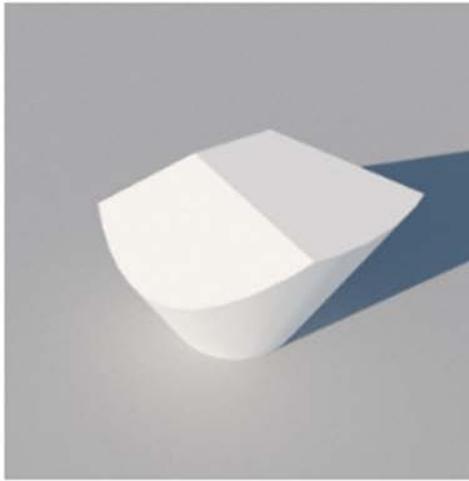
“cones,” are represented in various forms, such as short, flat disk type of cones and tall thin cones. We also employ several open-to-interpretation types of exemplars that present to the creator as just text. The following is a list of some of these text-based exemplars.

- Column
- Tree
- Scribble (several different types with different tags to reflect their nature)
- Hatching
- Stippling
- Potato (free form shape not quite a sphere)
- Cucumber (more elongated free form shape)
- Stairs
- Single line (many types with various tags)

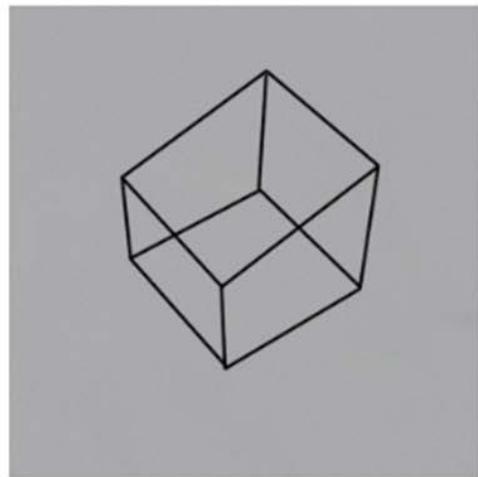
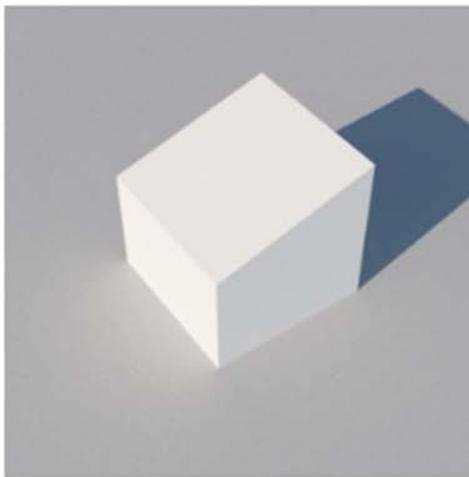
Exemplars may also include a set of constraints that may be applied to that exemplar to restrict the random variations. The data generation tool applies these constrained random variations before presenting the exemplar to the user. The constraints used include the following.

- Upright (only rotate around upward axis to preserve upright quality)
- Usually upright (decreases likelihood of transformations other than around up axis)
- Common 90-degree rotations on any axis
- Uniform scaling only
- No random scaling

Both the exemplar and a sketch made of that form using it as a guide indicates that the sketch is intending to convey similar attributes. (Figure 18)

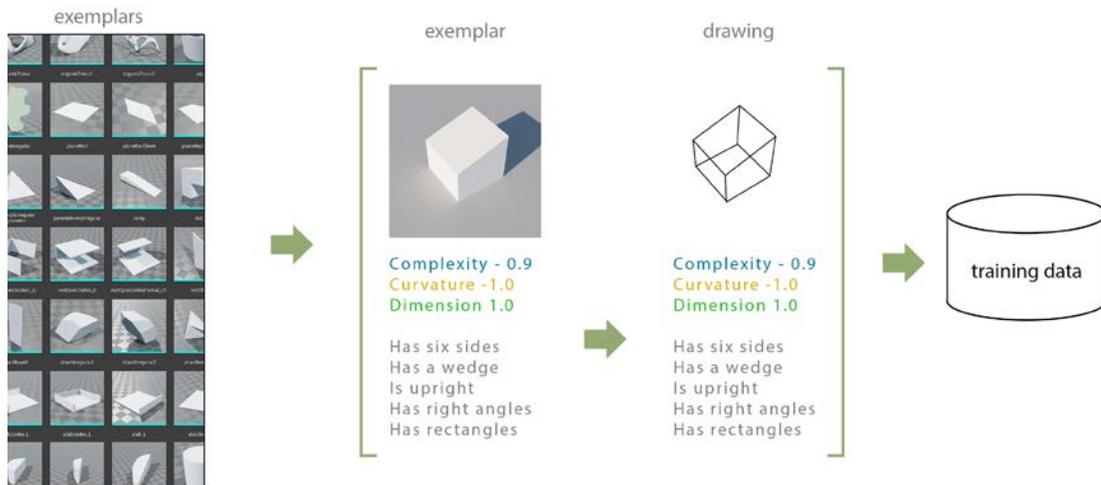


Complexity 0.1
Curvature -0.2
Dimension 1.0
Is a shard



Complexity -0.9
Curvature -1.0
Dimension 1.0
Has six sides
Has a wedge
Is upright
Has right angles
Has rectangles

Exemplars, associated drawing and their related attribute settings Figure 18



Data set generation process overview

Figure 19

The Process

One approach to creating the data set is to create random drawings and then label each of them appropriately. However, this approach does not guarantee that the distribution of types of drawings is desirable. (too many drawings of bunnies for example) Further, as the process of creating drawings continues the author would likely tend to drift and lose the consistency required to produce a data set suitable for linear or other classifications.

We've developed a process that allows for control of the types of drawings produced, their relative frequency in the data set and to promote diversity (rotation, scale and type). To this end, we have created an interactive application in VR to facilitate the creation of the data set. (Figure 19) This application presents a sequence of 3D shapes for the creator to use as a template or guide for creating a new drawing. As described in the previous section, these 3D shapes have all been tagged with a set of attribute descriptors. We call these forms "exemplars" as they function as examples of a form with the given attributes. For each exemplar presented, the author makes a

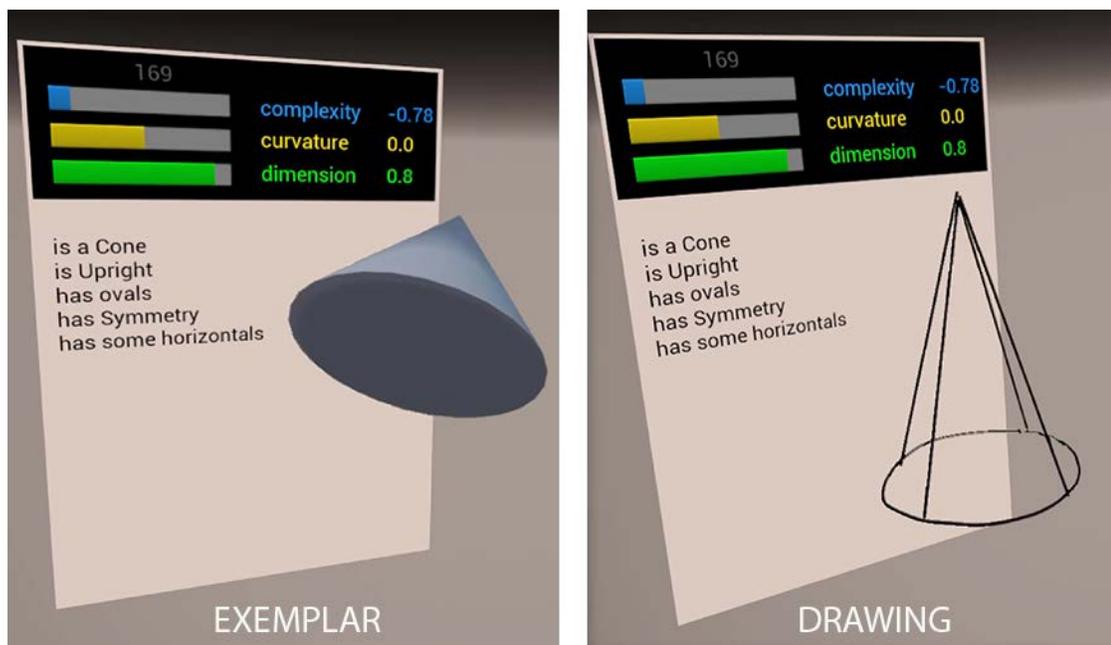
drawing to represent the character of the exemplar and then the drawing and the attributes of the exemplar are saved as a record in the data set, thus linking the drawing to the attributes. Each of these records are recorded individually in separate files and then are collated into a single data file with thousands of such records.

Data Generation Tool

The data generation tool is a sophisticated interactive system developed for efficiently generating a large collection of drawing and attribute pairs. The system was built using the *Unreal Engine 4* and the *Oculus Rift* VR gear. The system employs a three-dimensional drawing tool with only the essential features required for the task. For example, users cannot change the size or color of the drawing tool. Pulling the left trigger executes a drawing/exemplar cycle by saving the current drawing and attribute pair to an external file while loading in the next exemplar. It appears to the user that the drawing and exemplar are directly in front of their face floating in space about 15 inches away and the exemplars are usually approximately 12 inches in diameter. This scale has proven to be the most accessible for drawing easily within arm's reach, while sitting down. The user can, however, freely rotate, scale and position the exemplar and drawing, as well as move their view position to view it from another angle. The exemplar is selected from the array of possible exemplars (nearly 300) with a weighted random distribution. Boxes, planes, single lines and walls have a higher percentage likelihood of appearing to increase their representation in the final data set. At each iteration the exemplar is chosen and transformed. The transformations involve scaling, rotating and placement, within the confines of the exemplar constraints described in the preceding section. Care is taken to adjust any tags required as needed

after transformation. For example, the tags, “upright,” “has horizontals,” and “has verticals” might need to be set to “false” if the object has been rotated around any axis other than the up axis. Similarly, non-uniform scaling can require adjustments to tags such as, “is a Sphere,” or “has right angles.” The goal here is to avoid any unintended bias that might create false correlations in the data such as, “all spheres are centered at the origin,” etc. Ideally the drawing data will resemble natural sketches made by users of the V-Sketch application described in this thesis or similar applications.

In addition to displaying a transformed variation of the chosen exemplar, the tool also displays the relevant attribute values for this exemplar/drawing. For easy reference the data is displayed on a static plane just out of arm’s reach. (Figure 20)



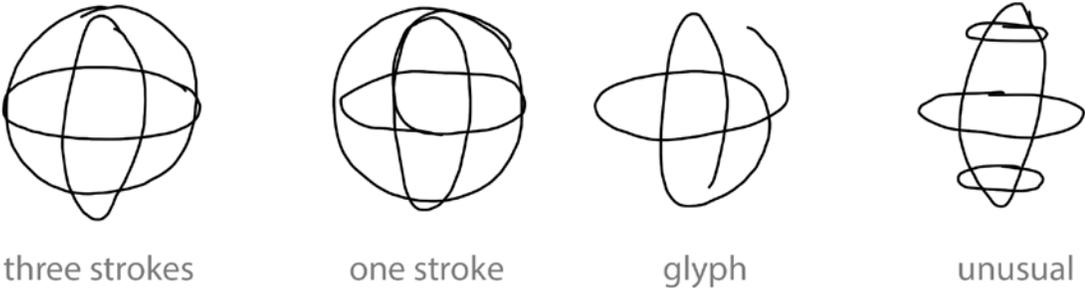
An exemplar and its drawing as seen in the data generator tool Figure 20

The attribute space variables are displayed in color coded bar graphs along the top with any attribute tags displayed below. This information is essential for creating the drawing as it informs the creator what features must be maintained. For example, one

exemplar is comprised of only text that says “1 line,” instructing the user to draw a single line. The type of line is indicated by the displayed attributes, such as, is the line three dimensional, or does it lie along a plane? How much curvature should the line display, etc.

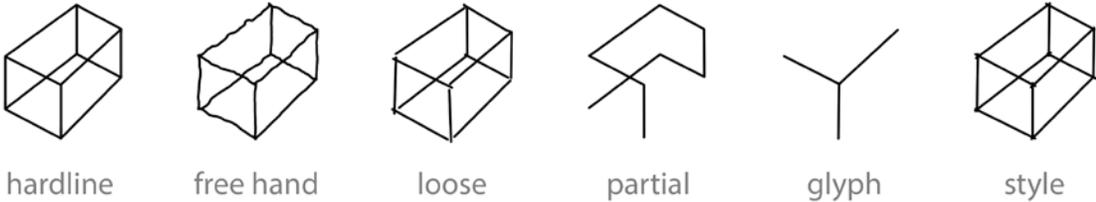
Bias Reduction

Effort was made to limit the amount of bias in the data. For example, there are several ways to draw a sphere in three dimensions and there are no clearly established methods for doing so. The drawings in the data set were intentionally varied to produce a more robust and generalizable model from the data. (Figure 21)



There are many ways to draw a sphere in three-dimensions Figure 21

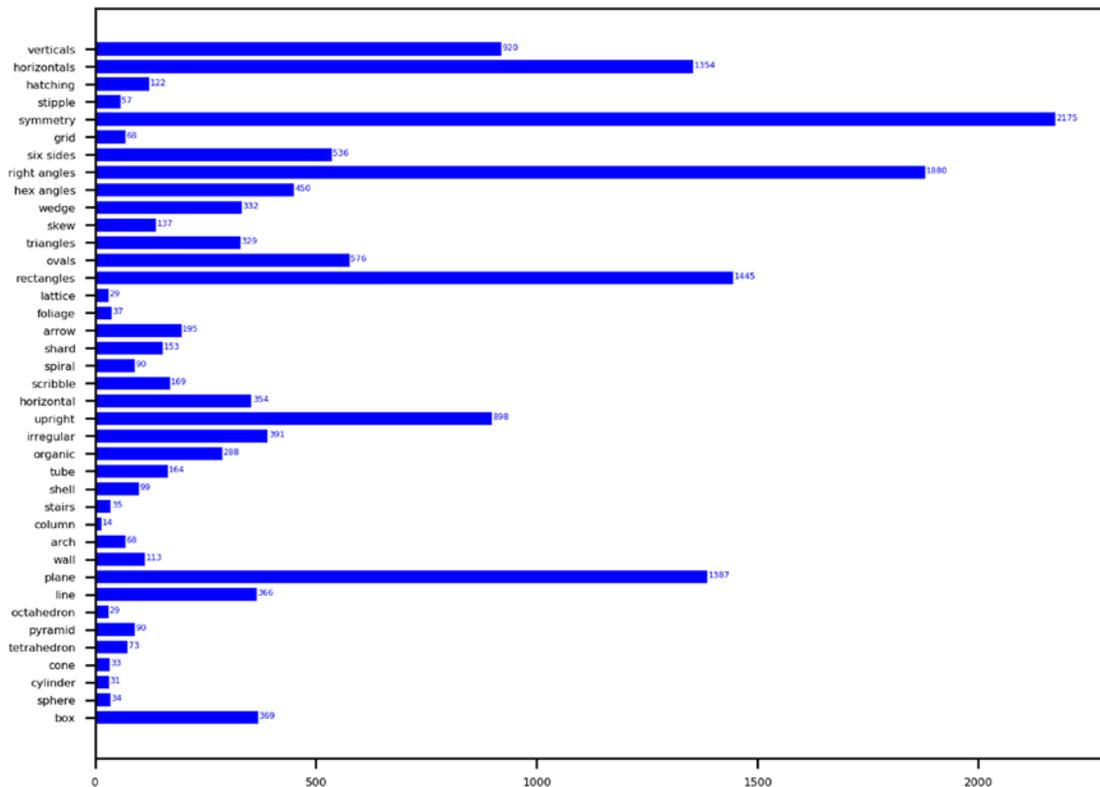
Besides a variety of approaches to representing forms in 3D, there is also the question of style. Similarly, efforts were made to vary this as well by using both free hand drawn straight lines as well as hard lines and varying degrees of completeness and looseness. Sometimes corners were accurate, sometimes open, sometimes overshot to create an overlapped and emphasized corner. (Figure 22)



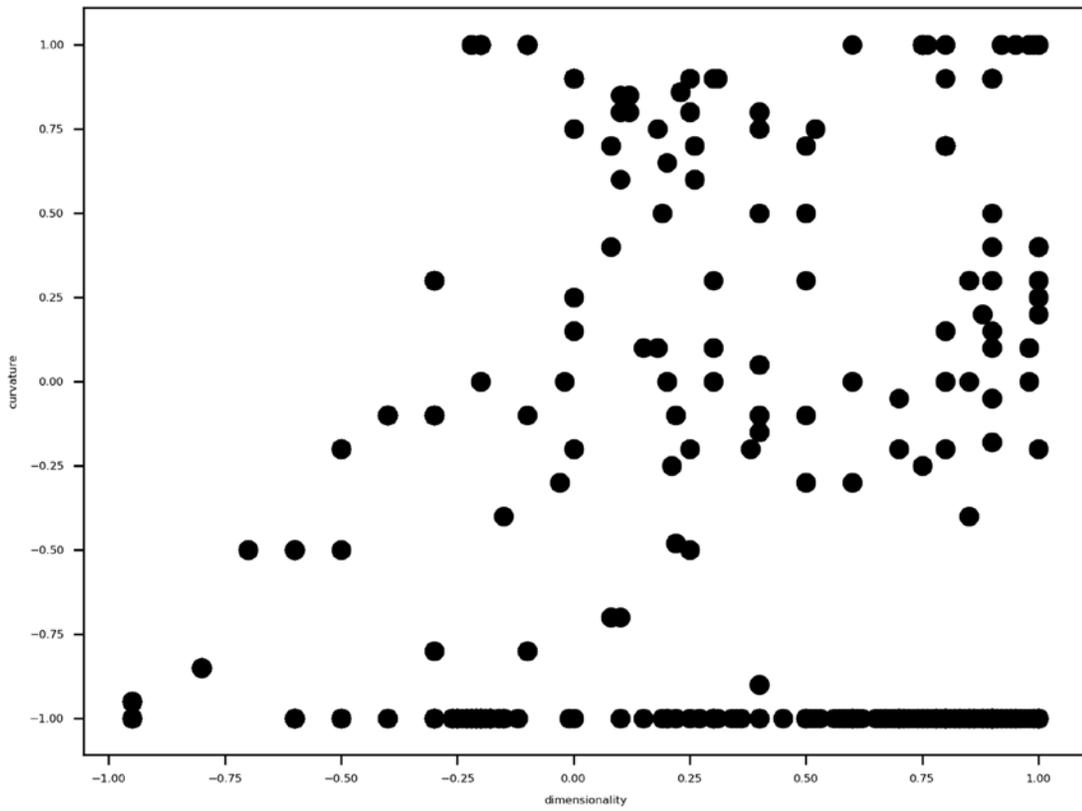
Efforts were made to vary drawings Figure 22

Results

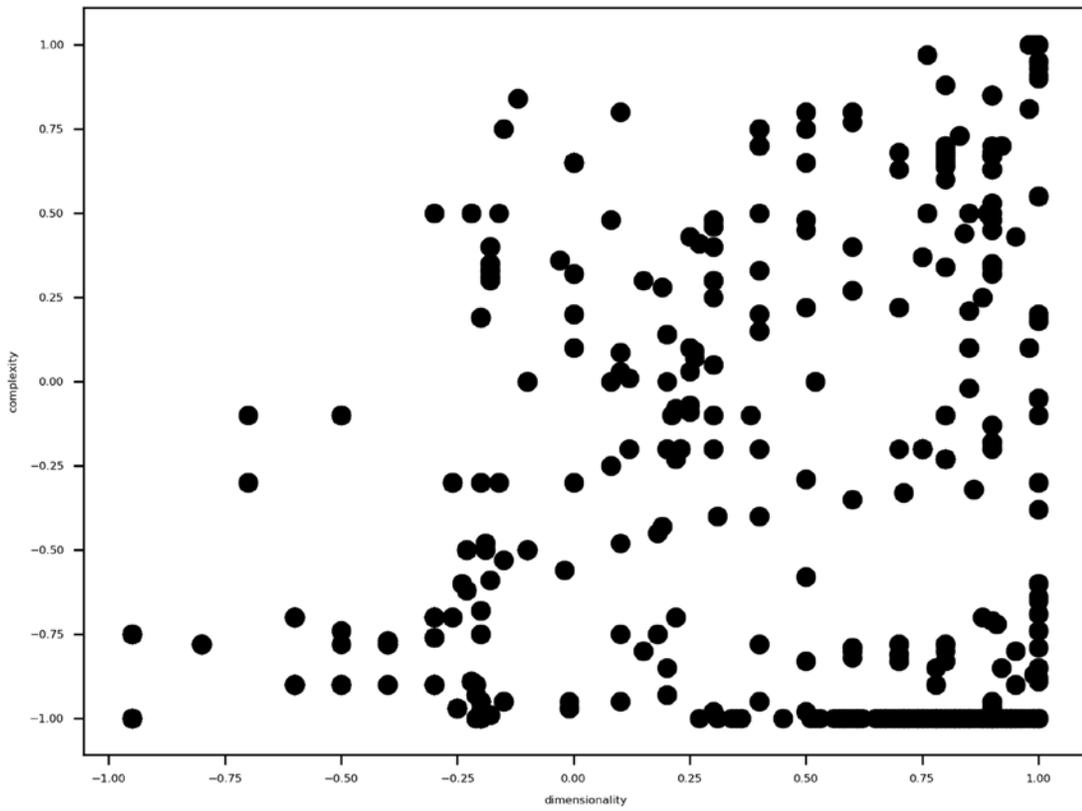
Over 5,000 organic data records or samples were created by hand. A further 5,000 have been generated by duplicating this original set with random perturbations to adjust the data. Care must be taken in that process to preserve the integrity of the attribute tags. Perturbations may include randomly rotating around the up axis, uniform scaling and slight changes in the position of each sketch. More dramatic changes such as randomly removing strokes from multi-stroke drawings or changing stroke order slightly is a further possibility but runs the risk of possibly losing essential information. The data distribution is seen in Figures 23 – 26.



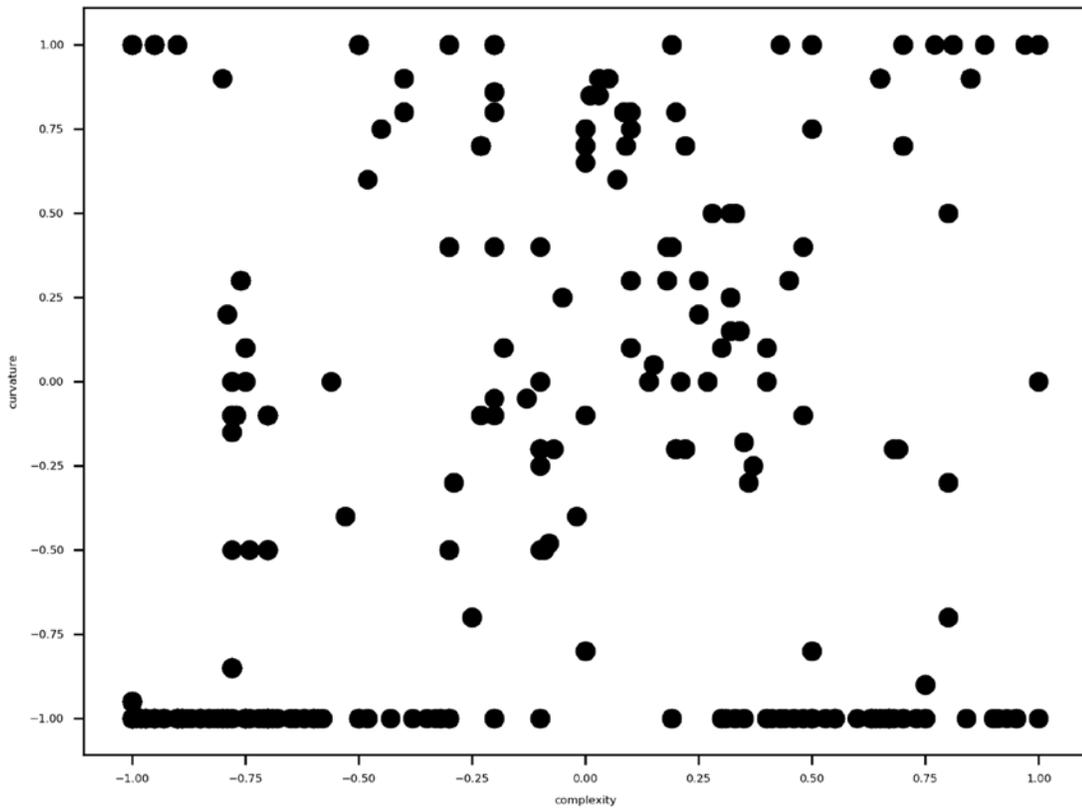
Histogram of Boolean attribute settings for over 4,000 samples Figure 23



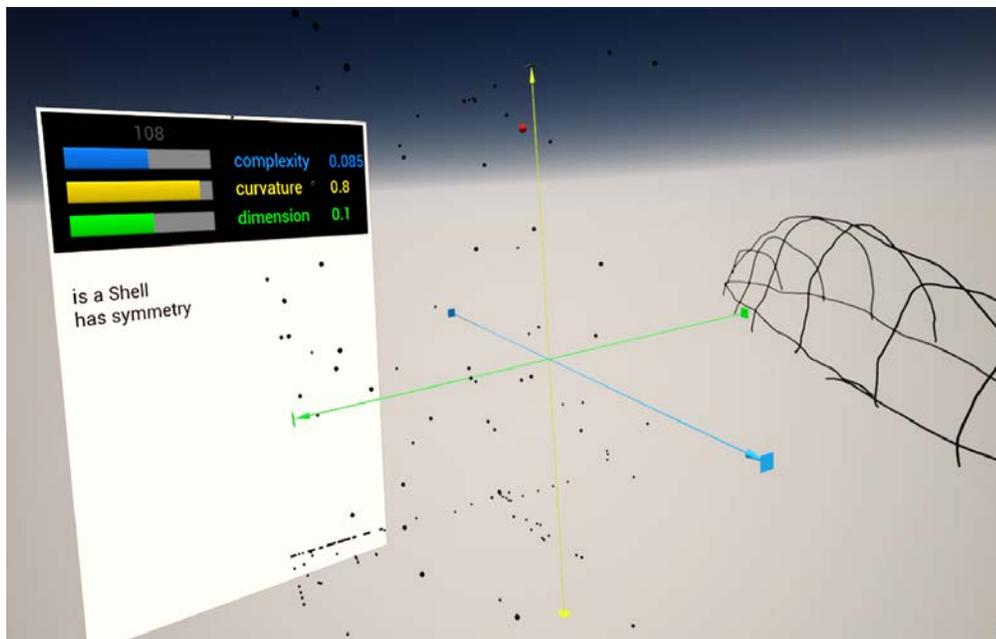
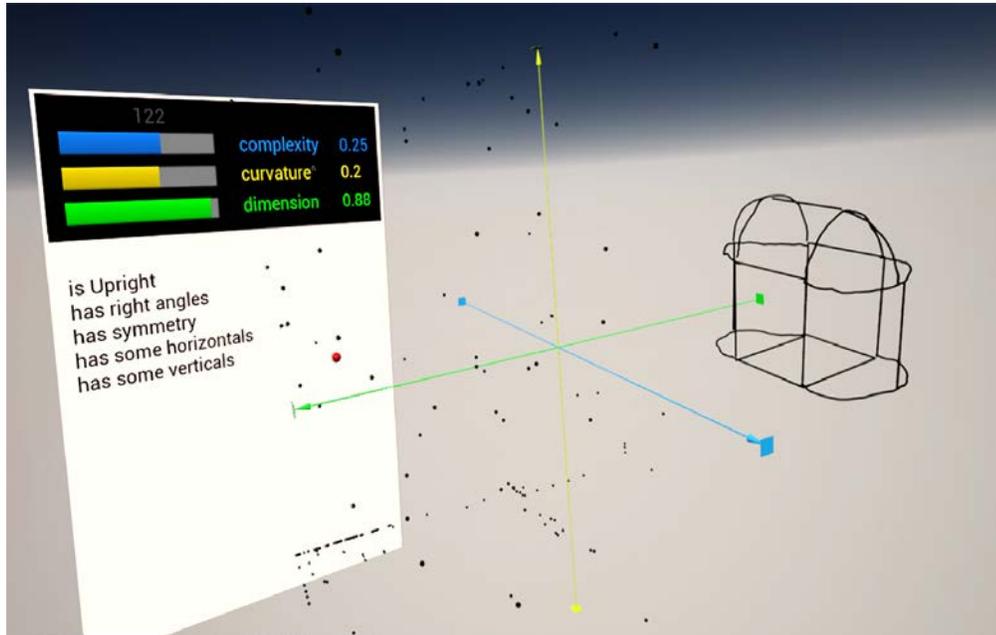
Scatter plot of exemplar attributes dimensionality and curvature Figure 24



Scatter plot of exemplar attributes dimensionality and complexity Figure 25



Scatter plot of exemplar attributes complexity and curvature Figure 26



Screen grab from VR data visualization tool

Figure 27

Data visualization

An interactive 3D data visualization tool was also made in addition to the data generation tool. This application, also in VR, loads any subset of the entire data set and visualizes the data for the user. (Figure 27) There is the same attribute

visualization as there is in the data generation tool, just beyond arm's reach. There is also the 3D drawing which this time has been reconstructed directly from the data sample. The entire data set can be displayed as spheres in a three-dimensional space with an axis for each of the attribute space dimensions. The axes are indicated by color matching the numerical display of the attributes. The currently selected data record is displayed as the only red sphere in the data set space. The interface allows the user to cycle sequentially through the data set samples or to directly select a point in the attribute space designating values of an individual data sample (drawing / attribute pair). The data set and drawing can be manipulated, allowing the user to position, rotate and scale each of them independently. Overlapping the selection icon (a sphere attached to the right-hand controller) with any data point sphere in the data set representation and pulling the right-hand trigger button will replace the current drawing and attributes with the drawing/attribute sample represented by the selected data sphere.

This immersive three-dimensional data visualization helps to reveal various clustering as well as the organization of the data that would otherwise be difficult to visualize. For example, there is a natural cluster of forms that are rectilinear and complex which is not surprising for a data set designed especially for architectural forms. Also, there's a distinct plane of points where the dimensionality has been set for planar forms. There is also a large cluster of data points with no curvature, no complexity and slightly varying but mostly high dimensionality. These of course are the forms that represent boxes of various dimensions.

CHAPTER 6

V-SKETCH PROTOTYPE

In addition to the training data set described above, we present a prototype version of the system that would integrate all the elements of the system we envision. This prototype we're calling V-Sketch uses a somewhat simple heuristic for the sketch analysis portion of the process. This is used in place of the anticipated more sophisticated machine learned solution built using the training data set. The prototype demonstrates the potential for this process while at the same time illustrating the need for more robust sketch analysis. (Figure 28)



Prototype uses a simpler heuristic in place of the model

Figure 28

Using this application, it is evident that sketching to both select (or generate) an element while at the same time scaling and placing all within the context of the composition is a notable advancement for early stage design. Ideas can flow with significantly less friction, similar to sketching with traditional media. The prototype reconfirms our belief that this easy flowing method of creating three-dimensional media will allow artists and designers to more easily and quickly explore a wider range of ideas. Importantly, working with this method simultaneously facilitates a designer's ability to more easily tap their unconscious impulses and discover their deepest inspirations.

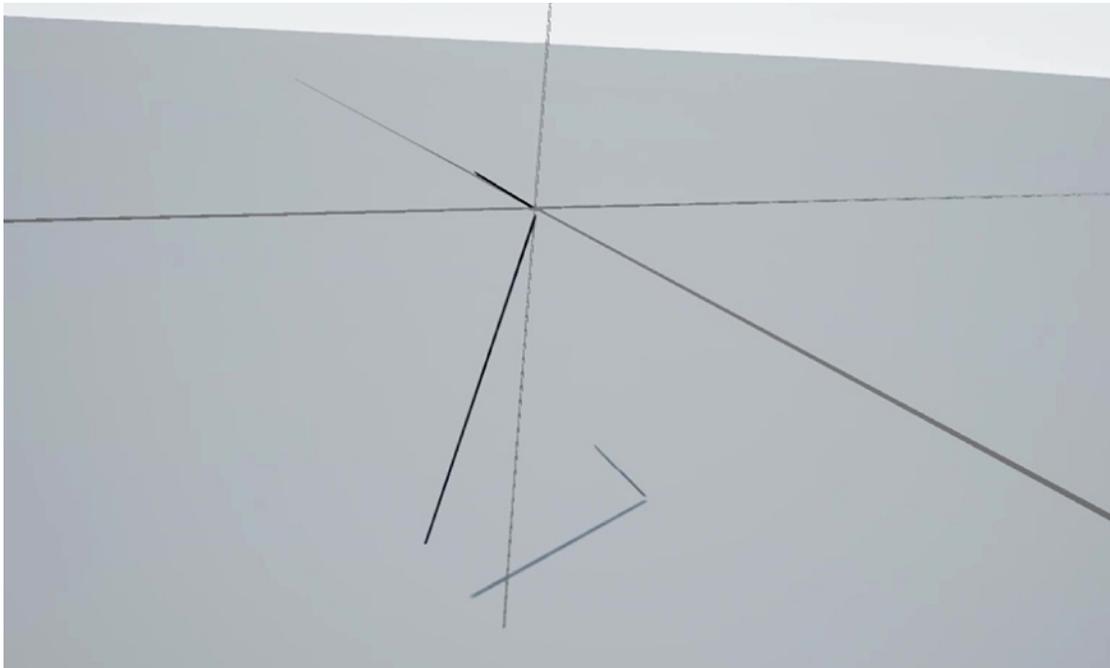
Sketching and interpreting those sketches into 3D elements alone is not enough to enable artists and designers to create worthwhile 3D compositions. The user interface for the V-Sketch prototype is minimal and yet still is quite powerful.

Interface and features

There's nothing simpler than picking up a pencil and a piece of paper and sketching. V-Sketch is a VR application and necessarily it will not be quite so simple, however, effort has been made to make it as easy to "pick-up-and-make-something" as it is with pencil and paper.

Environment

The application loads with a default background which would be one of several choices in a more developed version. The background is daylight over an open neutral plane. The scale is set to be about 1:10. Scale in this setting is limited primarily by the arm reach of a human user. A full-scale version would require a lot of navigation to overcome the limits of reach. The working area in a seated VR experience is roughly one to three feet in front of a user and about four feet side to side and nearly three feet high. A ground plane is provided to overcome the inherently uncomfortable feeling of being "untethered" when working in VR and provides an initial sense of space, albeit wide-open. Sketches and objects cast their shadows onto this plane to further the sense of grounding. The ground plane is positioned about as high as a low table, at arm's length but not likely to interfere. There is a small black sphere attached to the right-hand controller as an icon representing the tip of the drawing tool. The environment is designed to work well with the black line of the sketching tool.



Temporary axis to ground drawing lines in 3D

Figure 29

Sketching Tool

The sketching tool is kept very minimal. Pulling the trigger of the right-hand controller and moving the right hand, produces a flowing, free-hand drawn black line of uniform thickness in three dimensions. Users cannot change the size of the sketched line nor the color or style. An “axial guide” of thin grey lines is displayed with the origin at the starting point of each new stroke. This unique innovation has proven to be invaluable as an aid to sketching in VR. (see Figure 29) The axial guide provides a sense of context and space for the stroke and does much to combat the challenging “unmoored” sensation reported by users when sketching in VR. Squeezing the grip button of the left controller temporarily converts the drawing tool to a “straight line” tool. This tool uses the common “rubber-band” user feedback. This shows the user the line from their start position to their current position while they move their hand with the trigger held, but not actually making the line until they release the right trigger.

Pressing the 'A' button will delete the last stroke, continuing until there are no strokes left. Pressing the 'B' button will delete the entire current sketch. Moving the right joystick up or down will scale the current sketch. This sketching tool is identical to the one used to create the drawings featured in the training data set.

Instantiating a New Element

At any point the user can convert the sketch to a new element of the design by pressing the trigger button of the left-hand controller. This causes the sketch to be analyzed to produce an attribute vector that best matches the sketch. The prototype code uses placeholder algorithms to make an estimate of what an actual machine trained algorithm might produce. Complexity is determined by a function of the number of strokes and total stroke length. Curvature is based on the ratio of curved strokes to straight lines and Dimensionality is based on the ratio of the area of bounding box of the sketch, to the area of the longest dimension cubed. This approximation fails often and illustrates the need for more sophisticated solutions. However, it is sufficient to display the effectiveness of the overall approach that this thesis is based upon and serves as motivation for future work.

Once the attribute variables have been determined from the sketch analysis a reconstruction function simply sorts all the elements in the currently active library based on the distance of each element's attribute values to the target values produce by the analysis. With the library of elements thus sorted, the top six elements are loaded into an array along with the originating sketch as the default seventh element. The sketch is replaced by the first element of this array, the closest match to the attribute values. Pulling the trigger of the left hand again replaces that with the next element in

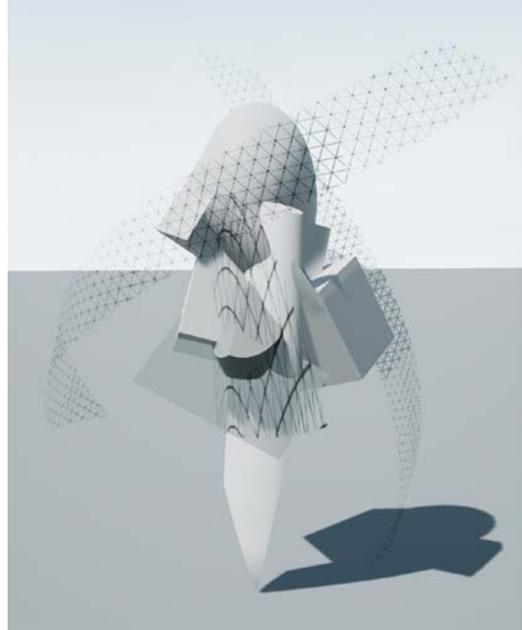
the array until the original sketch is displayed and cycles continuously from there. The new element has been situated by scaling the element to fit the axis aligned bounding box of the sketch and is positioned at the center of the sketch. At any point the user can hold the grip of the right-hand controller to adjust the position and orientation of the new element. The up and down axis of the right thumbstick will scale the new element.

Pressing the 'A' button at this time will delete the element. Pressing the 'B' button will delete the entire composition. Holding both left and right grip buttons down simultaneously allows the user to position, orient and scale the entire composition just as with the right grip only allows for the current element. Pressing the 'X' button of the left controller creates a duplicate of the current element. Pressing the 'Y' button takes a screen shot of the current composition and saves both a single and stereo image. (Figure 30)



Example saved screen shot from prototype

Figure 30



Three demonstration libraries for prototype

Figure 31

Libraries

Moving the left joystick right or left will cycle the selection of the current library. The name of the new library is displayed briefly to the user with a number indicating the total number of elements in the library. A few simple primitive libraries have been created to demonstrate the potential. One library, titled, “Library One” (upper left in Figure 31) is a basic library of simple shapes displaying a wide array of object types, simple, complex, curvilinear, rectilinear, planar and volumetric. Most of the objects have a custom material applied named, “Fade.” This material is such that whenever a new composition is started two nearly random colors are selected and each element is assigned a color between these two, incrementing 1/10 at each iteration. Then the color returns to the initial color in a similar fashion. This allows the forms to read individually relative to one another as opposed to merging to become one new shape. The color steps are subtle enough that the new composition is typically harmoniously linked by color. Another demonstration library has an even smaller set of 3D art assets that have been chosen to represent typical forms found in a California modern style home design, titled, “Library Modern.” (bottom in Figure 31) These elements include several with materials such as, poured concrete, white stucco, dark walnut paneled forms, light oak beaded panels, dark brushed aluminum, travertine marble, glass with mullions, grass and abstract transparent planes. As a demonstration, one custom material has been used to demonstrate the possibility of, “scale consistent” materials. The concrete material has been created such that as the elements are created at various sizes and possibly further scaled by the user, the scale of the texture remains consistent

and fixed to the world space, as one would normally desire. The third library is a small experimental library showcasing non-realistic materials designed to reflect the quick sketch-like nature of these compositions. It is titled, “Library Sketchy.” (upper right in Figure 31)

Library creation

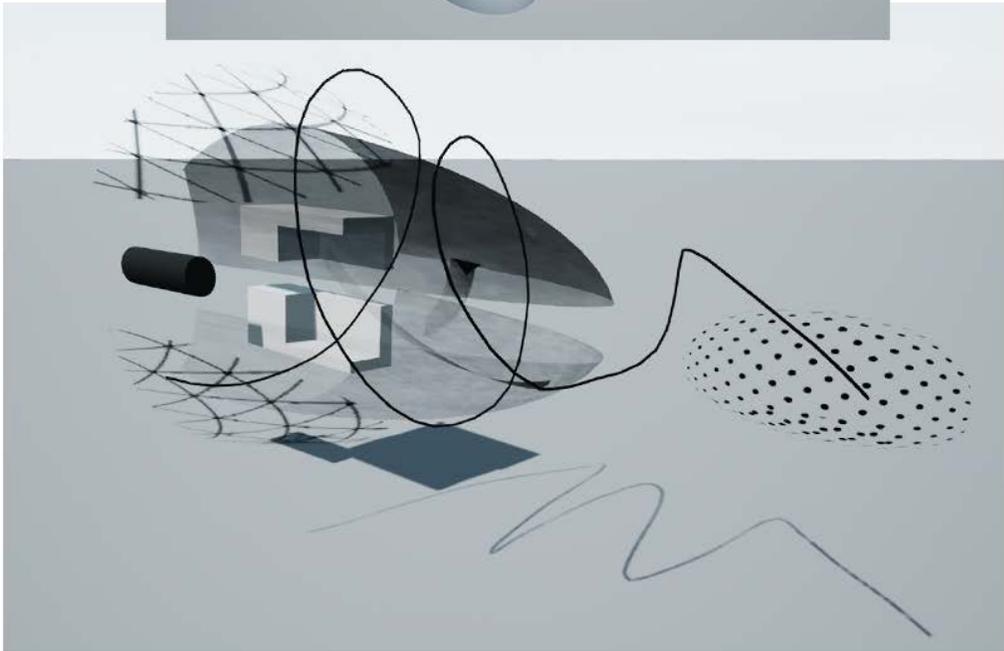
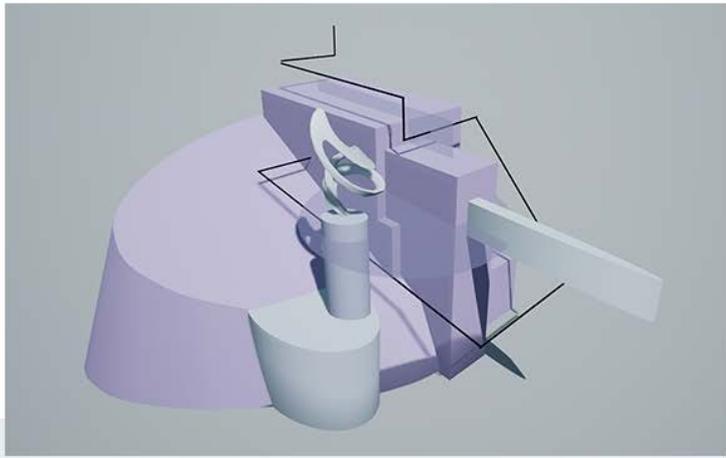
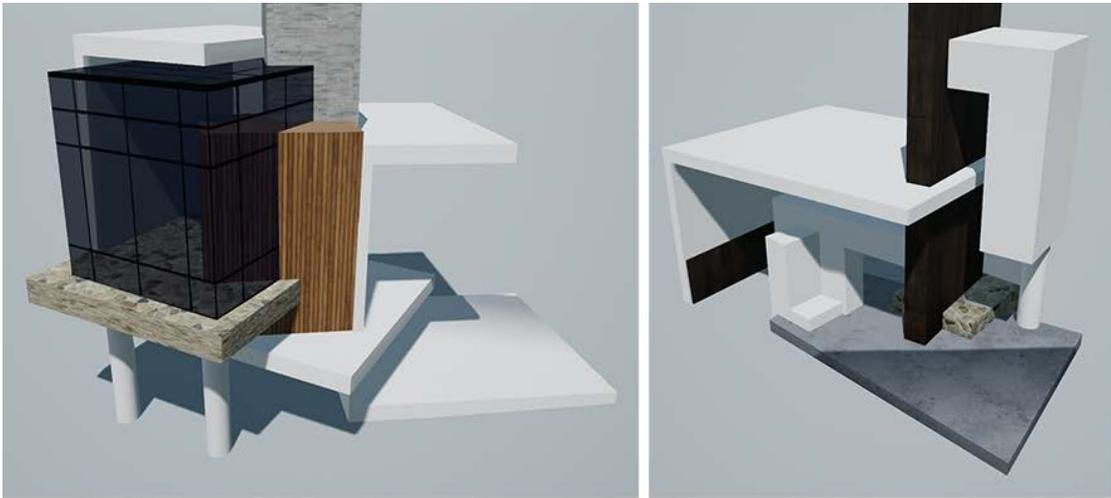
These libraries were created using standard 3D modeling methods known by most students with experience in digital media creation. These were created using Autodesk *Maya*. The materials were custom made in the *Unreal Engine* but use standard methodologies well understood by digital artists. These assets are added to the V-Sketch system, one by one and their attribute settings are manually set in the *Unreal Editor*. These settings can be tuned. In the V-Sketch application the author made sketches of each of the library elements and observed what attribute values the heuristic determined for each sketch. After several samples, these values were used to influence the attribute settings for the given element in the library. Working this way, even the placeholder sketch analysis system can interpret sketches reasonably.

Example output

The V-Sketch application has not been functional for a very long time and as such has not been exhaustively tested. However, early experiments can confirm that many of our assumptions are correct. This process does indeed allow for quick design exploration in a process that has many of the same benefits of sketching. It feels like, “sketching with concrete,” or marble and glass. The designer can find a relaxed, intuitive flow that is much more difficult to achieve with other methods, such as traditional 3D modeling or even easy to use programs such as, “Sketch-Up.” In

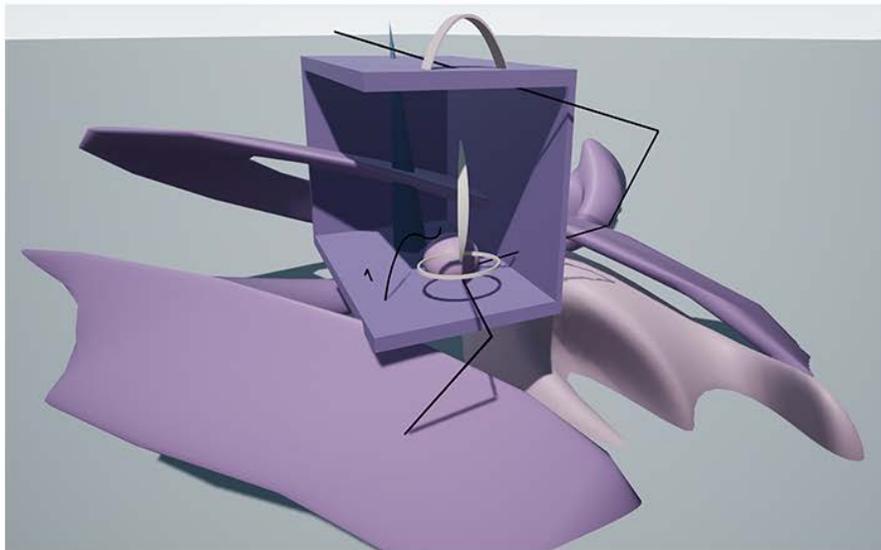
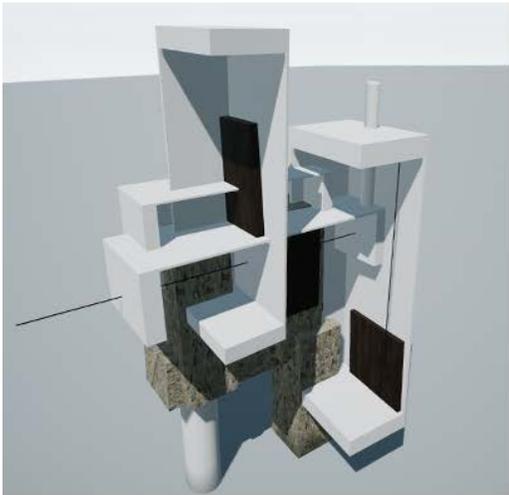
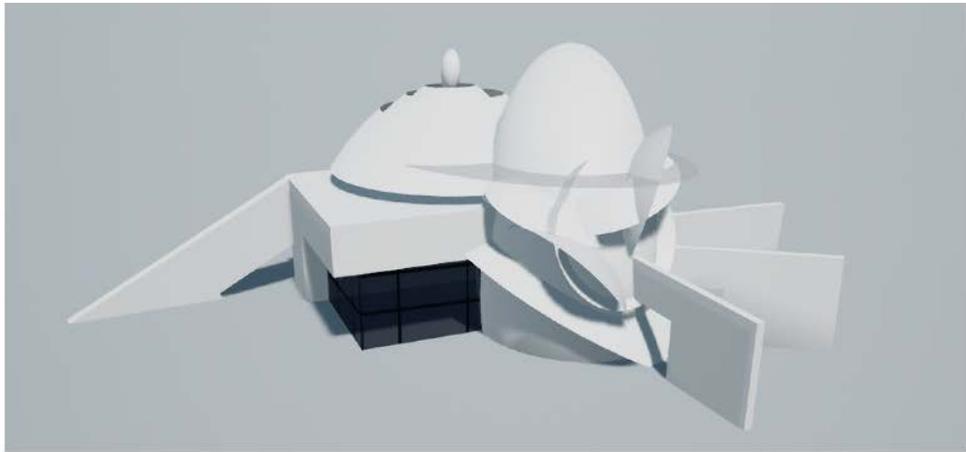
addition to that, the added benefit of creating these sketches in VR promotes the sculptural and spatial aspects of these designs in ways that 2D sketching cannot. It is also clear that the interaction with the machine as it serves up elements that the user might not have thought of enhances the process greatly. Even in this basic prototype, it feels as if the machine is in collaboration, like a jazz duo improvising to find a pattern unthought of previously. Lastly, it's now clear that the flexible, modular approach, utilizing interchangeable libraries or kit-of-parts is tremendously beneficial. Moreover, using the application is extremely fun and enjoyable.

Please see the example output from the V-Sketch Prototype, Figures 32 - 35



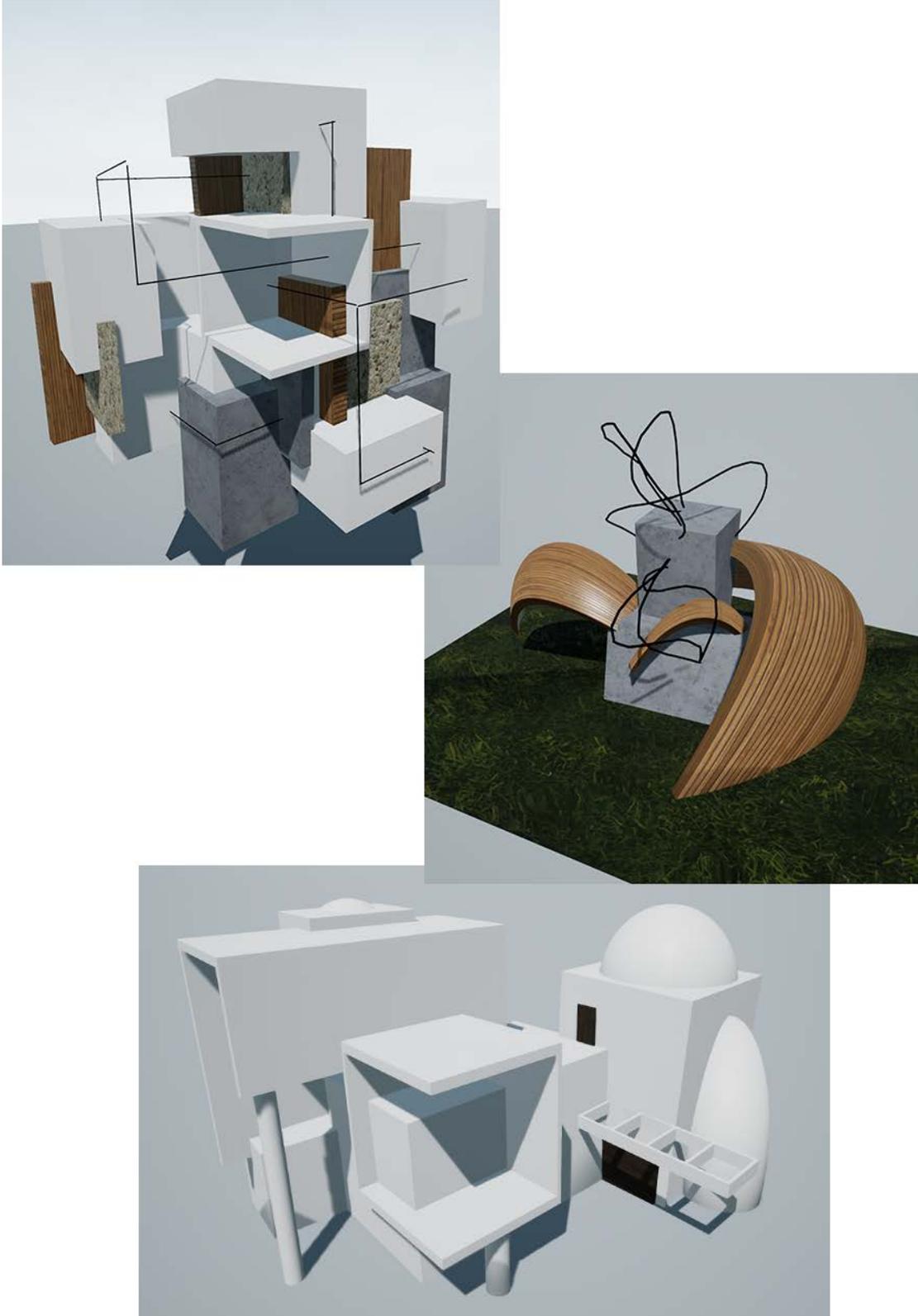
Example output from the V-Sketch Prototype

Figure 32



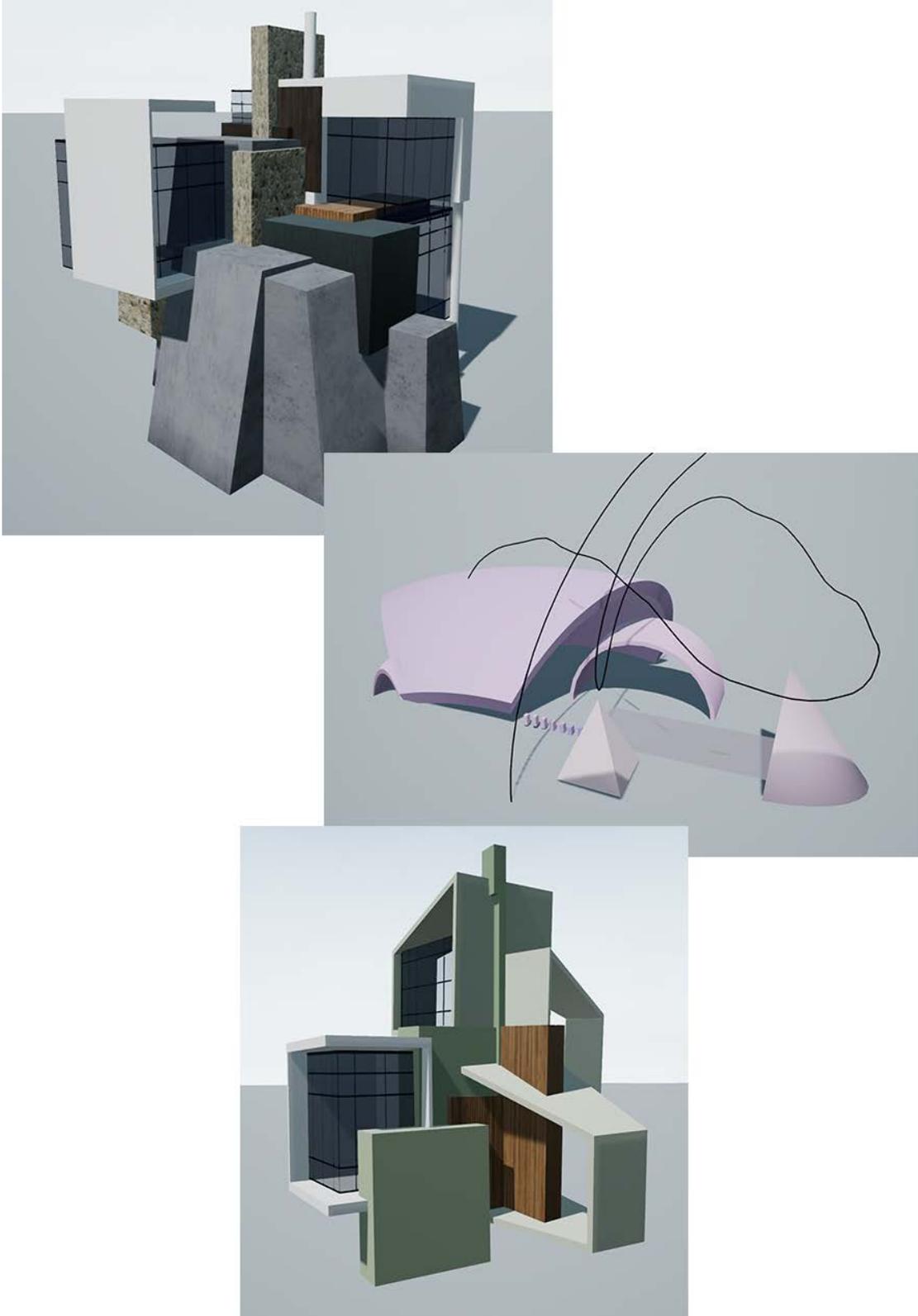
Example output from the V-Sketch Prototype

Figure 33



Example output from the V-Sketch Prototype

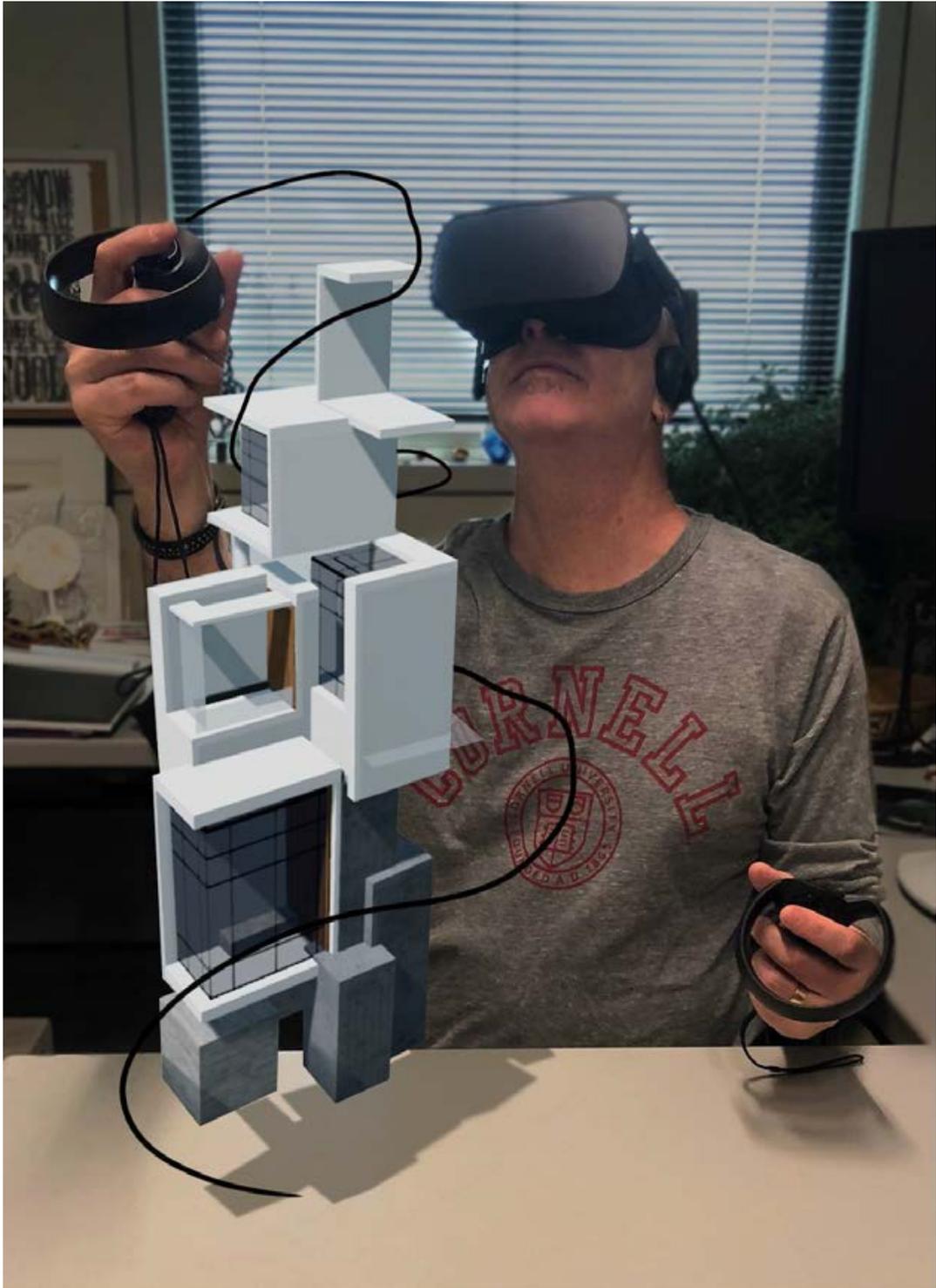
Figure 34



Example output from the V-Sketch Prototype

Figure 35

CHAPTER 7



A mixed reality mockup of the V-Sketch experience

Figure 36

CONCLUSION

Evidence suggests early stage design is primarily a process of exploration and discovery. It is also well understood that many critical decisions of any design problem are made at these early design stages. We have asked the question whether new emerging technologies can enhance this process, and if so, how might we best amplify human capabilities?

A primary method that is optimized for this process is the simple “sketch”. A sketch is easy to generate, is spontaneous and quick, and enables a creator to most easily tap into his/her intuitive design instincts. Sketching has been a beloved design tool for centuries. Unfortunately, although understood by the creator, a sketch is drawn on a two-dimensional surface like a plain sheet of tracing paper. It is ambiguous, can have many interpretations, and lacks sufficient information for accurate three-dimensional model representation. In fact, a sketch of an object or scene can be described as a “noisy projection of a three-dimensional object seen by a monocular observer from a single point of view”.

Can we create an input method which provides more unambiguous three-dimensional information and yet maintain all the ease of use and beneficial characteristics of the simple tracing paper sketch metaphor? Our goal has been to extend the potential of the “sketch” into three-dimensional media and augment the process with the aid of a computer. We present an interactive three-dimensional sketching system within a virtual reality environment as an excellent method to promote better human-computer interaction.

Emerging technologies of machine learning and virtual reality are enabling new forms of interaction and digital creation. This new human-machine collaboration creates an “augmented

intelligence” or an “augmented creativity” that was not possible just a few years ago. We stand at the dawn of the era of thinking machines. This symbiotic relationship offers a more optimistic potential future in place of one where humans relinquish design responsibilities to intelligent algorithms.

While recognizing that sketching is an excellent method for quickly exploring design ideas, there is an inherent difficulty in translating two dimensional sketches into a three-dimensional form. Noble attempts have been made to bridge this challenging divide but in all known cases the

interface required to determine the three-dimensional nature of the sketch interferes with the free-flowing spontaneity of the sketching experience.

However, one pillar in this new age of digital design is the rapidly improving capabilities of virtual reality. The ability to use a computer to generate an interactive, realistic, three-dimensional world is deeply transformative. In our case it allows the user to experience virtual designs as if viewing a flexible architectural model perceived to be a tangible form just inches from the user. Perhaps more importantly, it allows the user to easily make sketches in three dimensions. These sketches disambiguate the designers’ intent by avoiding the inherent spatial uncertainty of a two-dimensional drawing of a three dimensional form.

Also transformative are the recent developments in artificial intelligence in general, and more specifically, machine learning, which has been enabled by advances in

computer hardware and processing speed. This project explores and presents novel methods to use supervised machine learning for classification and regression algorithms to build a model capable of describing three dimensional sketches in a way that is valuable for the design of abstract 3D forms. These algorithms, however, require a large data set of 3D drawings and their associated descriptions, which we call “attributes.” A major component of this project was to create this data set consisting of over five thousand data samples. This effort required the creation of a data set generation tool as well as a data set visualization tool. The unique challenges required to train models from 3D drawings was discussed and presented along with a method of using this data to train a model. However, the actual machine learning part of this project was beyond the scope of this thesis.

A V-Sketch prototype was created to demonstrate the potential of this approach. It illustrates the effectiveness of our three staged approach, consisting of, the sketch analysis, situating and reconstruction function, while using the attribute description vector as the binding concept that ties it all together. The prototype uses a simpler heuristic to approximate the machine learned model, and also a somewhat simplified approach to situating. The prototype demonstrates both the need for better sketch analysis made possible with machine learning and yet at the same time provides inspiration and motivation for doing the work. Three modular, 3D element libraries were also created as part of the prototype to showcase the flexibility of our modular approach. The prototype in its current form clearly demonstrates that this approach has great potential to enhance early stage design exploration and warrants further development.

By combining custom libraries with a V-Sketch type of system, this approach can be applied most easily to architectural design, but also equally as well for sculpture, virtual worlds, education and other disciplines. The prototype will be published open source on Github along with the data set, the data set generation tool and the data set visualization tool.

We hope that others will see the value in pursuing this line of inquiry and join us in developing the concept further.

REFERENCES

- AARON, S., FARAMARZ, S., MARIO, C.S. 2006. Transformation strokes. Proceedings of the 3rd Eurographics Workshop on Sketch-based Interface and Modeling, Vienna, Austria
- ACHTEN, H., DE VRIES, B. AND JESSURUN, J. 2000. DDDOOLZ. A Virtual Reality Sketch Tool for Early Design. CAADRIA 2000 [Proceedings of the Fifth Conference on Computer Aided Architectural Design Research in Asia
- ACHTEN, H., ROELEN, W., BOEKHOLT, J.-TH., TURKSMA, A. AND JESSURUN, J. 1999. Virtual Reality in the Design Studio: The Eindhoven Perspective. Architectural Computing from Turing to 2000 [eCAADe Conference Proceedings / ISBN 0-9523687-5-7] Liverpool (UK)
- ADLER, A., EISENSTEIN, J., OLTMANS, M., GUTTENTAG, L., DAVIS R. 2004. Building the design studio of the future. Making Pen-Based Interaction Intelligent and Natural
- ALVARADO C., DAVIS R. 2001. Preserving the freedom of paper in a computer-based sketch tool. Proceedings of HCI international
- ALVARADO C., DAVIS R. 2004. SketchREAD: a multi-domain sketch recognition engine. Proceedings of the seventeenth annual ACM symposium on user interface software and technology. ACM; p. 23–32
- ALVARADO C., LAZZARESCHI M. 2007. Properties of real-world digital logic diagrams. Proceedings of the first international workshop on pen-based learning technologies. IEEE; . p. 1–6
- ANTHONY L., WOBROCK JO. 2010. A lightweight multistroke recognizer for user interface prototypes. Proceedings of graphics interface. Canadian Information Processing Society;. p. 245–52
- ANTHONY L., WOBROCK JO. 2012. \$N-protractor: a fast and accurate multistroke recognizer. Proceedings of graphics interface. Canadian Information Processing Society;. p. 117–20
- ARANDJELOVIC, R , SEZGIN, TM. 2011. Sketch recognition by fusion of temporal and image-based features. Pattern Recognit ;44(6):1225–34
- ARNOWITZ, E. MORSE, C. GREENBERG, DONALD P. 2017. vSpline: Physical Design and the Perception of Scale in Virtual Reality. ACADIA 2017: Disciplines & Disruption [Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture] Cambridge, MA
- BAILEY, ROHAN 2000. The Intelligent Sketch: Developing a Conceptual Model for a Digital Design Assistant. Eternity, Infinity and Virtuality in Architecture [Proceedings of the 22nd Annual Conference of the Association for Computer-Aided Design in Architecture] Washington D.C.
- BOURGUIGNON, D., CANI, M.P., DRETTAKIS, G. 2001. Drawing for Illustration and Annotation in 3D. Computer Graphics Forum, 20, 3 , 114-122. (Proc. Eurographics '01)
- CAKMAK, S. 2016. Clustering free-form sketch scenes through perceptual similarity. Turkey: Koc University; Master's thesis

- "CALDERON, C., VAN SCHAİK, P., AND HOBBS, B 2000. Is VR an Effective Communication Medium for Building Design?. Proceedings. Virtual Reality International Conferences 2000, Laval, France"
- CHEN, D.-Y., TIAN, X.-P., SHEN, Y.-T., OUHYOUNG, M. 2003. On visual similarity based 3d model retrieval. *Comput. Graph. Forum (Proc. Eurographics)* 22, 3, 223–232
- CHEN, T., MING CHENG, M., TAN, P., SHAMIR, A., MIN HU, S. 2009. Sketch2photo: internet image montage. *ACM SIGGRAPH Asia*
- COHEN J.M., HUGHES, J.F., ZELEZNIK R.C. 2000. Harold: A World Made of Drawings. *Proceedings of NPAR*, pp. 83-90
- COLE, F., GOLOVINSKIY, A., LIMPAECHER, A., BARROS, H. S., FINKELSTEIN, A., FUNKHOUSER, T., RUSINKIEWICZ. 2008. Where do people draw lines?. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 27, 3
- CORTES, C., VAPNIK V. 1995. Support-vector networks. *Mach Learn*;20(3):273–97
- COSTAGLIOLA, G., DE ROSA, M., FUCCELLA V. 2014. Local context-based recognition of sketched diagrams. *J Vis Lang Comput*;25(6):955–62
- DELANOY, J., BOUSSEAU, A., AUBRY, M., ISOLA, P., EFROS, A.A. 2016. What You Sketch Is What You Get: 3D Sketching using Multi-View Deep Volumetric Prediction. *Inria Universite C'ote d'Azur, ^Ecole des Ponts ParisTech, UC Berkeley*
- DO, ELLEN YI-LUEN 2000. Sketch that Scene for Me: Creating Virtual Worlds by Freehand Drawing. *Promise and Reality: State of the Art versus State of Practice in Computing for the Design and Planning Process [18th eCAADe Conference Proceedings*
- DORSEY, J., Xu, S., SMEDRESMAN, G., RUSHMEIER, H., MCMILLAN, L., 2007 The Mental Canvas: A Tool for Conceptual Architectural Design and Analysis, *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, October.
- DORSEY, J., LEONARD M. 1998. Computer Graphics and Architecture: State of the Art and Outlook for the Future. *Computer Graphics*, 32(1): 45-48
- DOUGLAS, D.H., PEUCKER, T.K . 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartogr Int J Geogr Inf Geovis*;10(2):112–22
- EGGLI, L., CHING-YAO, H., AND BRUDERLIN, B.D. 1997. Inferring 3D models from freehand sketches and constraints. *Computer-Aided Design* 29 (2), 101–112
- EITZ, M., HAYS, J., ALEXA, M . 2012. How do humans sketch objects?. *ACM Trans Graph*;31(4):44–51
- EITZ, M., RICHTER, R., BOUBEKEUR, T., HILDEBRAND, K., ALEXA, M. 2012. Sketch-based shape retrieval. *ACM Trans. Graph. (Proc. SIGGRAPH)* 31, 4
- EITZ, M., RICHTER, R., HILDEBRAND, K., BOUBEKEUR, T., ALEXA, M. 2011. Photosketcher: interactive sketch-based image synthesis. *IEEE Computer Graphics and Applications*
- EMILIEN, A., VIMONT, U., CANI, M., POULIN, P., BENES, B. 2015. WorldBrush: Interactive Example-based Synthesis of Procedural Virtual Worlds. *SIGGRAPH*

- '15 Technical Paper, August 09 – 13, 2015, Los Angeles, CA. ACM Trans. Graph. 34, 4, Article 106, 11 pages
- FAUGERAS, O. 1993. Three-dimensional computer vision: A geometric viewpoint. MIT Press, Cambridge, MA, USA
- FERNANDEZ-PACHECO, ALBERT, F., ALEIXOS, N., CONESA, J. 2012. A new paradigm based on agents applied to free-hand sketch recognition. Expert Systems with Applications 39 7181–7195
- FONSECA, M., BARROSO, B., RIBEIRO, P., JORGE, J. 2004. Sketch-based retrieval of clipart drawings. AVI'04: Proceedings of the ACM Press, New York, NY, USA
- FUNKHOUSER, T., MIN, P., KAZHDAN, M., CHEN, J., HALDERMAN, A., DOBKIN, D., ACOBS, D. 2002. A Search Engine for 3D Models. ACM Transactions on Graphics, Vol. V, No. N, 10 202002
- FUNKHOUSER, T., MIN, P., KAZHDAN, M., CHEN, J., HALDERMAN, A., DOBKIN, D., JACOBS, D. 2003. A search engine for 3D models. ACM Trans. Graph. 22(1), 83–105
- GUTIERREZ, A., CASSAB, N., GREENBERG, DONALD P. 2017. Cuttlefish 3D, Explorations in 3D Drawing on 2D Surfaces. Cornell Program of Computer Graphics, Unpublished
- HAMMOND, T.A . LADDER. 2007. a perceptually-based language to simplify sketch recognition user interface development. Ph.D. thesis Massachusetts Institute of Technology
- HEROT, C. F. 1976. Graphical input through machine recognition of sketches. Computer Graphics (Proc. SIGGRAPH) 10, 2, 97
- HESSAM, J.F., MANOLYA, K., BOYALI, A. 2017. MATRACK: block sparse Bayesian learning for a sketch recognition approach. Springer Science+Business Media New York
- HSE, H., NEWTON, A.R. 2004. Sketched symbol recognition using Zernike moments. ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1 pp. 367–370
- HSE, H., NEWTON, A.R. 2004. Sketched symbol recognition using Zernike moments. Proceedings of the seventeenth international conference on pattern recognition, ICPR, vol. 1. IEEE; p. 367–70
- HUBEL, D.H., WISEL, T.N., 1959. Receptive fields of single neurones in the cat's striate cortex
- IGARASHI, T., HUGHES J.F, 2001. A Suggestive Interface for 3D Drawing. 14th Annual Symposium on User Interface Software and Technology, ACM UIST'01, Orlando, FL, November 11-14
- IGARASHI, T., MATSUOKA, S., TANAKA, H. 1999. Teddy: A sketching interface for 3D freeform design. Proceedings of SIGGRAPH '99, pp. 409–416. ACM Press, New York, NY, USA
- IP, H.H.S., CHENG, A.K.Y., WONG, W.Y.F., FENG, J. 2001. Affineinvariant sketch-based retrieval of images. CGI '01: Proceedings of the international Conference on Computer Graphics, p. 55, IEEE Computer Society, Washington, DC, USA

- KARTHIK, R., AND SUYU, H. 2006. Sketch-based 3D engineering part class browsing and retrieval. *EuroGraphics Symposium Proceedings on Sketch-based Interfaces and Modeling*, 131–138
- KATO, T., KURITA, T., OTSU, N., HIRATA, K. 1992. A sketch retrieval method for full color image database-query by visual example. *Pattern Recognition, Vol.I. Conference A: Computer Vision and Applications, Proceedings., 11th IAPR International Conference on*, 530–533
- LEE, Y., ZITNICK, C., COHEN, M. 2011. ShadowDraw: real-time user guidance for freehand drawing. *ACM Trans. Graph. (Proc. SIGGRAPH) 30, 4*, 27:1–27:10
- LICKLIDER, J. C. R. 1960. Man-Computer Symbiosis. *IRE Transactions on Human Factors in Electronics*, volume HFE-1, pages 4-11, March 1960. MIT Computer Science and Artificial Intelligence Laboratory.
- MAHONEY, J.V., FROMHERZ, M.P. 2002. Three main concerns in sketch recognition and an approach to addressing them. *Proceedings of the AAAI spring symposium on sketch understanding*; p. 105–12
- MILLER, E.G., MATSAKIS, N.E., VIOLA, P.A. . Learning from one example through shared densities on transforms. *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1. IEEE; 2000. p. 464–71
- NIELS, R., WILLEMS, D., VUURPIJL, L. 2008. The NICICON database of handwritten icons for crisis management. 2. Nijmegen, The Netherlands: Nijmegen Institute for Cognition and Information, Radboud University Nijmegen
- NISHIDA, G., GARCIA-DORADO, I., ALIAGA, D., BENES, B., BOUSSEAU. 2016. Interactive Sketching of Urban Procedural Models. *ACM Trans. Graph.* 35, 4, Article 130 11 pages
- OLTMANS, M. 2007. Envisioning sketch recognition: a local feature based approach to recognizing informal sketches. thesis Massachusetts Institute of Technology
- OUYANG, T., DAVIS, R. 2007. Recognition of hand drawn chemical diagrams. *Proceedings of the AAAI*, vol. 7; p. 846–51
- OUYANG, T., DAVIS, R. 2009. A visual approach to sketched symbol recognition. *IJ- CAI*, 9; p. 1463–8
- OUYANG, T., DAVIS, R. 2009. Learning from neighboring strokes: combining appearance and context for multi-domain sketch recognition. *Proceedings of the advances in neural information processing systems*; p. 1401–9
- OWADA, S., NIELSEN, F., OKABE, M., AND IGARASHI, T. 2004. Volumetric illustration: Designing 3D models with internal textures. *ACM Trans. Graph.* 23(3), 322–328
- PENTLAND, A., AND KUO, J. 1989. The artist at the interface. *Vision Science Technical Report 114*, 18–26
- PICCOLOTTO, M. A., 1998 Sketchpad+ architectural modeling through perspective sketching on a pen-based display. Master's thesis, Cornell University
- PLATT, J. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Adv Large Margin Classif* ;10(3):61–74
- PLIMMER, B., BLAGOJEVIC, R., CHANG, SH-H., SCHMIEDER, P., ZHEN, J.S. 2012. RATA: codeless generation of gesture recognizers. *Proceedings of the*

- twenty-sixth annual BCS interaction specialist group conference on people and computers. British Computer Society; p. 137–46
- PLIMMER, B., FREEMAN, I. 2007. A toolkit approach to sketched diagram recognition. Proceedings of the twenty-first British HCI group annual conference on people and computers: HCI... but not as we know it, vol. 1. British Computer Society; p. 205–13
- ROMERA-PAREDES, B., TORR, P. 2015. An embarrassingly simple approach to zero-shot learning. Proceedings of the thirty-second international conference on machine learning; p. 2152–61
- ROSENBLATT, F., 1958. The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain, Cornell University
- RUBINE, D. 1991. Specifying gestures by example. SIGGRAPH Comput Gr;25(4):329–37
- SAFAR, M., SHAHABI, C., SUN, X. 2000. Image retrieval by shape: A comparative study. IEEE International Conference on Multimedia and Expo, pp. 141–144
- SANGKLOY, P., BURNELL, N., HAM, C., HAYS. 2016. The Sketchy Database: Learning to Retrieve Badly Drawn Bunnies. ACM Trans. Graph. 35, 4, Article 119 12 pages
- SEZGIN, T.M., DAVIS, R. . Sketch recognition in interspersed drawings using time-based graphical models. Comput Gr 20 08;32(5):50 0–10
- SEZGIN, T.M., DAVIS, R. 2002. Generating domain specific sketch recognizers from object descriptions. Proceedings of the student oxygen workshop, vol. 17
- SHEPP, B.E., BALLESTEROS, S. - EDITED BY 1989. Object Perception: Structure and Process. Lawrence Erlbaum Associates, Inc.
- SHILMAN, M., PASULA, H., RUSSELL, S., NEWTON, R. 2002. Statistical visual language models for ink parsing. Proceedings of the AAAI spring symposium on sketch understanding; p. 126–32
- SHIN, H.J., IGARASHI, T. 2007. Magic Canvas: Interactive Design of a 3-D Scene Prototype from Freehand Sketches. Graphics Interface Conference
- SHPITALNI, M., LIPSON, H. 1996. Identification of faces in a 2D line drawing projection of a wireframe object. IEEE Trans. Pattern Anal. Mach. Intell., 18(19), 1000–1012
- SHPITALNI, M., LIPSON, H. 1996. Optimization-based reconstruction of a 3D object from a single freehand line drawing. Computer-Aided Design, 28(8), 651–663
- SKURICHINA, M., DUIN, R.P.W. 1996. Stabilizing classifiers for very small sample sizes. Proceedings of the third international conference on pattern recognition. ICPR- vol. 2. IEEE; 1996. p. 891–6
- SONG, Y., DAVIS, R., MA, K., PENNY, D.L. 2016. Balancing appearance and context in sketch interpretation. Proceedings of the twenty-fifth international joint conference on artificial intelligence
- SUTHERLAND, I. 1964. SketchPad: a man-machine graphical communication system. Proc. AFIPS, 323–328

- SZUMMER, M. 2005. Learning diagram parts with hidden random fields. Proceedings of the eighth international conference on document analysis and recognition (ICDAR'05). IEEE; p. 1188–93
- TANAKA, T., NAITO, S., TAKAHASHI, T. 1989. Generalized symmetry and its application to 3D shape generation. *Visual Computer*, 5(1–2), 83–94
- TARANTA, II E.M., LAVIOLA, JR. J.J. 2015. Penny pincher: a blazing fast, highly accurate \mathcal{S} -family recognizer. Proceedings of the forty-first graphics interface conference. Canadian Information Processing Society; p. 195–202
- TAX, D.M.J. 2001. One-class classification. Ph.D. thesis TU Delft, Delft University of Technology
- TAX, D.M.J., DUIN, R.P.W. . Data description in subspaces. Proceedings of the fifteenth international conference on pattern recognition, vol. 2. IEEE; 2000. p. 672–5
- TAX, D.M.J., DUIN, R.P.W. 2004. Support vector data description. *Mach Learn*;54(1):45–66
- TIRKAZ, C., EISENSTEIN, J., SEZGIN, T.M., YANIKOGLU, B. 2015. Identifying visual attributes for object recognition from text and taxonomy. *Comput Vis Image Underst*;137:12–23
- TOLBA, O., DORSEY, J., MCMILLAN, L. 1999. Sketching with Projective 2D Strokes. Proceedings of the ACM Symposium on User Interface Software and Technology, Asheville, NC
- TOLBA, O., DORSEY, J., MCMILLAN, L., 2001 A Projective Drawing System, Proceedings of the 2001 ACM Symposium on Interactive 3D Graphics (Research Triangle Park, NC), pages 25-34, March
- TUMEN, R.S., ACER, M.E., SEZGIN, T.M. . Feature extraction and classifier combination for image-based sketch recognition. Proceedings of the seventh sketch- based interfaces and modeling symposium
- VATAVU, R-D., ANTHONY, L., WOBROCK, J.O. 2012. Gestures as point clouds: a \mathcal{S} p recog- nizer for user interface prototypes. Proceedings of the fourteenth ACM in- ternational conference on multimodal interaction. ACM; p. 273–80
- VESELOVA, O., DAVIS, R. 2006. Perceptually based learning of shape descriptions for sketch recognition. Proceedings of the thirty-third international conference on computer graphics and interactive techniques, SIGGRAPH Boston, MA, USA; 2006. p. 28
- VIOLA, P., JONES, M.J., SNOW, D. 2005. Detecting pedestrians using patterns of motion and appearance. *Int J Comput Vis* ;63(2):153–61
- VRIES, B. DE, JESSURUN, J. AND ENGELI, M 2000. Development of an intuitive 3D sketching tool. Timmermans, Harry (Ed.), *Fifth Design and Decision Support Systems in Architecture and Urban Planning - Part one: Architecture Proceedings* (Nijkerk, the Netherlands)
- WERTHEIMER, M. 1938. *A Source Book of Gestalt Psychology*. 1923. Translation in W. D. Ellis (ed.) New York: H. B. J.
- WITKIN, A. P. 1980. *Shape from Contour*. MIT AI-TR-589 November
- WOBROCK, J.O., WILSON, A.D., LI, Y. 2007. Gestures without libraries, toolkits or training: a \mathcal{S} 1 recognizer for user interface prototypes. Proceedings of the

- twentieth annual ACM symposium on user interface software and technology. ACM; p. 159–68
- YANIK, E., SEZGIN, T.M. 2015. Active learning for sketch recognition. *Comput Gr*;52:93–105
- YESILBEK, K.T. SEZGIN, M.T. 2017. Sketch recognition with few examples. *Computers & Graphics - Volume 69*, December, Pages 80-91
- YU, Q., YANG, Y., SONG, Y-Z., XIANG, T., HOSPEDALES, T. 2015. Sketch-a-net that beats humans. *Proceedings of the British machine vision conference (BMVC)*
- ZELEZNIK, R.C., HERNDON, K.P., HUGHES, J.F. 1996. SKETCH: An interface for sketching 3D scenes. *Proceedings of SIGGRAPH '96*, pp. 163–170. ACM Press, New York, NY, USA
- ZHANG, D., LU, G. 2002. Shape Based Image Retrieval Using Generic Fourier Descriptors. *Proc. of 5th Asian Conference on Computer Vision (ACCV)*
- ZHEN, J.S., BLAGOJEVIC, R., PLIMMER, B. 2012. Automated labeling of ink stroke data. *Proceedings of the international symposium on sketch-based interfaces and modeling. Eurographics Association*; p. 67–75
- ZHU, F., XIE, J., FANG, Y. 2016. Learning Cross-Domain Neural Networks for Sketch-Based 3D Shape Retrieval. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*
- ZHU, X. 2005. Semi-supervised learning literature survey. Technical Report. Computer Sciences, University of Wisconsin-Madison