Creative Visual Expression in Immersive 3D Environments

by

Rahul Arora

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Creative Visual Expression in Immersive 3D Environments

Rahul Arora
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2021

Recent developments in immersive technologies of virtual and augmented realities (VR/AR) have opened up the immersive space for 3D creation. By removing the crutch of two-dimensional proxies for input and output, the immersive medium presents a significant advance in the design of intuitive interfaces for 3D creative expression. However, immersion also presents its own unique challenges in precision, control, ergonomics, and perception. This thesis advocates for a human-centred approach to the design of creative tools for immersive environments. Conducting interviews, observational studies, and formal quantitative experiments reveals insights into the needs and aspirations of future users of immersive tools. These user studies build fundamental knowledge about the limitations of human perception and our musculoskeletal system. These insights are then utilized to build concrete design guidelines for immersive creation tools. We present four immersive tools, targeted at sketching, modelling, and animation tasks in VR/AR. These tools harness 3D mid-air interactions to improve the user experience for existing design tasks, and to enable novel design aesthetics.

Dedicated to my late झाईजी (grandmother).

Acknowledgements

Through this journey, I have learned from and been helped by so many people that it would be impossible to try to name everyone here. Still, I will try my best.

I would start by thanking my adviser Karan Singh for his support through the last five years. He has been a constant source of motivation, encouragement, and knowledge. I am also immensely grateful to my long-term collaborator and mentor Rubaiat Habib, who has always been a pillar of support. In the same vein, I have gained so much from working with David Levin, Adrien Bousseau, Tovi Grossman, Tim Langlois, Danny Kaufman, Alec Jacobson, and Wilmot Li, and I am grateful for the faith they posed in me. I am also extremely thankful to my committee members Alec Jacobson and Daniel Wigdor, external appraiser Julie Dorsey, and examination committee member Ravin Balakrishnan, for their constructive comments which helped frame this work in a broader context.

I thank my peers at DGP for being sources of inspiration as well as sounding boards for my half-baked ideas, especially Jiannan Li, Varun Perumal, Peter Hamilton, Mingming Fan, Chris De Paoli, Alex Tessier, Josh Holinaty, Abhishek Madan, Silvia Sellán, Sarah Kushner, Hsueh-Ti Derek Liu, Karthik Mahadevan, and Masha Shugrina. And doubly especially Michael Tao and Nicole Sultanum who, in addition to the above, listened to my numerous rants and complains over the years. I am also thankful to all the faculty members at DGP for their guidance, and to John Hancock, Xuan Dam, and Shaza'a Fayyaz for ensuring that all the administrative duties were a breeze and everything just worked!

I owe my gratitude to the individuals and organizations who found it prudent to give me money for this work: Adobe, Mitacs, Greg Wolfond, and Okino Computer Graphics. I am also thankful to Adobe and Autodesk for hosting me for internships and to my fellow interns and friends who helped make these internships joyous experiences, especially Ailie Fraser, David Ledo, Matt Overby, and Arjun Srinivasan. Back at home, thanks to Connie Chen for being the best roommate imaginable.

I thank my parents for putting up with a son who decided to move 11,686 km away to pursue his interests. And to my brother Prince, without him none of this would have been possible. Lastly, but the opposite of leastly, I thank Debanjana for sticking by me this whole time and constantly encouraging me to do better and to **write my thesis**.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recent advancements in Virtual and Augmented Reality (VR/AR) technologies have enabled novel ways of visualizing 3D content and interacting with it. For decades, artists, designers, and computer graphics developers have created 3D media overwhelmingly using devices that are two-dimensional. From the point of view of this thesis, the two-dimensionality of such traditional devices broadly manifests itself in two ways—the dimensionality of the interactions performed by the user (the *input*), and that of the *output* device utilized to visualize the artifact being created. While the immersive 3D visualization capabilities of VR/AR environments is extremely important and beneficial for the creation of 3D content and is taken advantage of by the tools presented in this work, the technical content of this thesis is primarily concerned with the former, that is, the novel 3D input capabilities.

Thus, my primary thesis is

> *3D interactions in immersive environments can be harnessed to improve workflows in 3D creation applications and to engender novel design languages for 3D creative expression.*

To demonstrate the generality of this statement, I describe the tools I have built which utilize immersive 3D interactions for 3D sketching, concept modelling, animation, and projective drawing on meshes.

Does this thesis propose that 3D interactions can solve all major user interaction problems in interactive graphics? No. In fact, this thesis proposes quite the opposite. My experiments show that immersive 3D interactions have numerous inherent shortcomings. I have conducted various user studies which reveal that while 3D interactions are natural and intuitive for 3D design tasks, they are far from an all-curing panacea for the design of creative interfaces. On the contrary, the studies reveal that control and precision issues can often be exacerbated with 3D interactions. Naturally then, the tools I have designed build upon this experimental evidence and augment novel 3D interaction techniques with well-known interaction paradigms to bring out the best of both worlds. While designing these tools, I follow the principle of "low threshold, high ceiling, and wide walls" [191]. That is, the tools should be approachable, but should allow their users to build sophisticated 3D artifacts, and be generic enough to enable a variety of target applications.

1.1 Creation of 3D Content

Humans have been creating three-dimensional artifacts for millennia in the form of sculpture. From the ceramic earthenware of China, to the refined sunk reliefs of Egypt and the marble figures from Europe, ancient 3D artifacts have been discovered in all parts of the world. While physical creation remains popular, 3D creation now takes numerous digital forms. Artists model characters for games, create animations for movies, and design 3D visual effects. Product designers use their computers to design everything from a screwdriver to a car. Infrastructure and urban planners may model geometry as complex as the buildings of a megacity and the network of cables and pipelines spanning the city.

Today, almost all of these 3D creations are realized on a computer whose input and output devices are both restricted to two dimensions. Inputs are provided via a mouse or a graphic tablet, while the output device is a flat screen. Thus, artists are only able to interact with their 3D creations via a 2D window. In contrast, the immersive environments of VR/AR unlock the third dimension for visualization and interaction. Modern VR and AR devices rely on the combination of a stereoscopic head-mounted display (HMD) for immersive visualization. While systems that do not rely on HMDs such as multiprojector VR and mobile device–based AR have also been proposed, this thesis is only concerned with HMD-based immersion. The position and orientation of the HMD is actively tracked and it drives the camera used for rendering the virtual world, thus simulating a feeling of presence in the 3D space for the viewer. 3D interaction is facilitated via similarly tracked controllers, or using bare-hand tracking techniques. This combination of 3D visualization and interaction capabilities can be harnessed by creative tools for creating directly in 3D without resorting to the traditional 2D proxies.

1.2 Creative Benefits of Immersion

Immersion provides numerous benefits to creators. Being immersed in a 3D scene, a designer can simply walk around or use indirect 3D interactions to manipulate the virtual camera and examine the scene from any desirable viewpoint. By utilizing 3D interactions for 3D tasks, immersion helps design *natural user interfaces*, that is, interfaces that can be effectively invisible to the user. Immersion also provides a sense of scale absent in desktop workflows. Earlier, designers would have to employ extremely large and expensive screens or create full-scale physical prototypes, modern VR/AR devices now enable 1:1 scale visualization in a much more efficient, economic, and portable manner. Chapters 4 and 6 explore the impact of this enhanced sense of scale for 3D design tasks. Filmmakers can take advantage of immersive VR by creating virtual content that surrounds the viewer—Google Spotlight Studio's *Sonaria* [89] is a wonderful example. Storytellers can use this novel way of creating and consuming entertainment to situate viewers in the character's perspective, thus provoking a greater sense of empathy with the characters' experiences. Please see Atlas V's *Goodbye Mr. Octopus* [20] for an inspiring example.

Real-world design tasks can gain additional benefits from HMD-based Augmented Reality—modelling in context of the physical world. Imagine an interior designer populating a physical room with her virtual creations, or an architect superimposing a proposed building next to existing real-world structures. Or directly modelling augmentations for physical objects, thereby ensuring that the virtual models fit the target object and have the desired look and feel, without the need for physical prototypes. Chapter 4 describes a novel immersive tool for in situ 3D creation, along with various 3D designs created using the tool that incorporate both physical and virtual objects.

1.3 3D Inputs: Opportunities and Challenges

The goal of this thesis is to explore the novel creative possibilities of immersive environments. The process involves understanding the benefits as well as the limitations of 3D inputs, inferring concrete design guidelines for improving the user experience by mitigating the observed limitations, and designing novel tools and techniques based on the design guidelines. In Chapter 3, I describe an observational study that reveals the excitement generated by mid-air interactions among professional artists and designers. Intuitively, 3D shape creation and manipulation tasks stand to benefit from 3D inputs as the need for indirect view manipulation to explore the 3D world via 2D proxies is eliminated; creating a 3D curve no longer requires multi-view or projective sketching; and translating or rotating a 3D object becomes as simple and intuitive as in the real world. However, when interactions are moved into the 3D space, the lack of constraints can lead to added fatigue, imprecision, and control issues. With 2D input devices, the hand rests on a physical surfaces such as a mouse, a keyboard, or a drawing tablet when using pen inputs. My SymbiosisSketch system (Chapter 4) aims to solve many of these challenges for 3D design tasks by combining freeform 3D inputs with constrained 2D interactions in a hybrid interface. The use of a familiar input device, a digital pen, and connecting the 3D design process to the 2D sketching paradigm makes the SymbiosisSketch interface more *intuitive* for the user. The CASSIE concept modelling tool (Chapter 6) uses a novel optimization framework to automatically neaten freehand 3D strokes and connect strokes together to form curve networks. In Chapter 5, I propose a new algorithm for drawing curves onto triangle meshes. A key motivation for the method is reducing the degrees of freedom that a user needs to control, thereby improving drawing precision and enabling the creation of complex curves which would otherwise be rendered impossible by the constraints of the human musculoskeletal system. Chapter 7 delves into the high degree of freedom manipulations enabled by mid-air hand gestures. The central aim is understanding which degrees of freedom (DoFs) can be simultaneously controlled, and how should those DoFs map to parameters of an animation.

1.4 Contributions

The work presented in this thesis has resulted in the following peer-reviewed publications, listed chronologically.

— Chapter 3 [16] "Experimental Evaluation of Sketching on Surfaces in VR". CHI 2017. DOI: 10.1145/3025453.3025474.

— Chapter 4 [17] "SymbiosisSketch: Combining 2D and 3D Sketching for Designing Detailed 3D Objects in Situ". CHI 2018. DOI: 10.1145/3173574.3173759.

— Chapter 7 [18] "MagicalHands: Mid-Air Hand Gestures for Animation in VR". UIST 2019. DOI: 10.1145/3332165.3347942.

— Chapter 6 [271] "CASSIE: Curve and Surface Sketching in Immersive Environments". CHI 2021. In collaboration with Emilie Yu at the Denmark Technical University. DOI: 10.1145/3411764.3445158.

— Chapter 5 [19] "Mid-Air Drawing of Curves on 3D Surfaces in Virtual Reality". TOG 2021 (to appear, to be presented at SIGGRAPH 2021). DOI: 10.1145/1122445.1122456.

1.5 Overview

I begin with a brief state of the art survey of the relevant literature in digital sketching, modelling, and animation in Chapter 2, with a focus on immersive creation. Chapter 3 describes my experiments on 3D sketching to understand user expectations for mid-air sketching tools, quantify 3D sketching imprecision in comparison with 2D sketching, and explore hardware and software interventions to improve 3D sketching precision. This chapter contributes concrete design guidelines for 3D sketching tools, which are implemented by SymbiosisSketch, a hybrid 2D–3D sketching tool in AR described in Chapter 4. This is followed up by a novel technique for drawing curves on surfaces in VR environments (Chapter 5), along with a formal user study validating that the novel technique is a significant improvement over the state of the art. Chapter 6 describes a concept sketching tool which corrects and neatens freehand 3D inputs for designing well-connected curve networks. These curve networks are also surfaced automatically in real-time. Chapter 7 pivots from the creation of static geometry to animation, describing a gesture elicitation study for VR animation interfaces and a prototype gestural animation tool. I conclude in Chapter 8 with an analysis of the current state of the art in immersive creation and the future potential of immersive environments for creation and consumption of 3D content.

Chapter 2

Background and Related Work

In this chapter, I provide the necessary historical background to situate this work in the computer graphics and human-computer interaction literature. I will delve into four broad themes: the design and evaluation of interactions for immersive environments, the use of sketching in the design process, shape modelling via sketching, and animation authoring interfaces. This is followed by a brief introduction to experiment design and analysis process in HCI and the statistical concepts relevant to the presented work.

2.1 Immersive Interactions

For interacting with desktop interfaces, broad paradigms include the keyboard-based command line interfaces (CLIs) and the mouse-driven graphical user interfaces (GUIs). For touchscreen devices, touch-based interfaces that go beyond the traditional *windows, icons, menus, pointer (WIMP)* approach to GUI-building on the desktop were conceived [10], and touch has become the dominant interaction modality for mobile devices today. Other input modalities such as speech, pen input, sketch gestures, gaze, and computer vision–based gesture recognition have also been widely explored in the literature [240]. For immersive environments, the dominant input device today is a six–degrees of freedom (6-DoF) tracked handheld controller [70, 95, 103]. The controller acts as a proxy of the user's hand and can therefore be used to interact with virtual objects analogous to interactions with the physical world. With recent progress in sensing hardware and machine perception, directly tracking the user's hands has become feasible in real-world applications and bare handed gestures are increasingly being utilized for VR/AR interactions [145, 185, 222]. Hand gestures have the advantage of being potentially more intuitive, since the interactions do not need to be mediated via a controller; as well as expressive, since users can use their hand pose as an input signal to the computer. While other input modalities such as speech and eye tracking continue to be actively researched for immersive environments [50, 186, 227], including for creative applications [262], the focus of our work is on tracked devices and bare-handed gestures, both of which we will now discuss.

2.1.1 Interactions using a Tracked Device

Contemporary VR and AR devices ship with standard controllers which are tracked in space via a combination of optical, inertial, and/or magnetic sensors [95]. While the ubiquity of such 6-DoF controllers

is relatively recent, the tracking technology is backed by decades of sensing research. This ubiquity, combined with the tracking accuracy and the benefits of standardization, have made these controllers the input device of choice for both commercial [4, 71, 87] as well as research applications [151, 250].

However, new devices continue to be conceived in the HCI community to better support specific applications such as design sketching [68, 150, 196]. In addition, new prototype devices are designed to inform the development of future devices by evaluating the impact of factors such as grip [30] or to evaluate and improve tactile aspects such as pneumatic feedback and haptic rendering [68]. In desktop computing today, most interfaces are driven by ubiquitous devices such as mice and keyboards, but special-purpose applications utilize specialized input devices such as styli and game controllers. One can imagine an analogous future for VR and AR where standardized 6-DoF controllers dominate general-purpose computing, but custom devices abound for particular use cases.

But perhaps the ubiquitous input device of the future is no device at all. Below, we review recent work on the use of bare-handed gestures in immersive environments.

2.1.2 Bare-handed Gestural Interaction

The expressive power of hand gestures has been extensively investigated in the human-computer interaction literature [121, 158, 204, 258, 261]. Bare-handed gestures enable natural interaction metaphors for direct manipulation of virtual content. Such interactions are appealing to interaction designers as they leverage users' real-world experience manipulating physical objects. Further, the human hand can be extremely expressive, enabling the simultaneous control of many variables. Researchers have leveraged the communicative aspects of freeform hand gestures in computing systems for 3D model retrieval [101, 268], approximating 3D shapes for early stage design [136], and interacting with imaginary devices [228].

From a mechanical perspective, the human hand has 27 DoFs [6]. As such, the number of independent parameters a mid-air gesture can simultaneously specify is theoretically very high (54 for bimanual). However, many of these DoFs are heavily intertwined and have severely limited ranges of independent motion [153]. Further, the number of parameters that can be simultaneously manipulated is restricted by users' mental models relating parameters to each other and by the cognitive load of the task [100]. Brouet et al. [35] conducted a study to understand the *true* number of DoFs users specified in table-top gestures for 3D manipulation, finding that users can only manipulate 4–6 DoFs using their dominant hand. Their observations can guide the design of mid-air gestures as well.

Traditionally, gestural interfaces have been designed by experts and tested a posteriori [190, 263]. The contrasting interface design methodology is *participatory design*, where users participate in the formative design process. Wobbrock et al. [261] first advocated this process for gesture design, with an elicitation study for performing canonical UI manipulation and navigation tasks on the table-top. Following Wobbrock et al., *user-designed gestures* have been utilized to build interfaces for a variety of domains [195, 199, 243, 249]. Particularly relevant to immersive creation are the recent works on grasping [268], remote object manipulation [272] and music composition [145] in VR environments, and the elicitation study of Piumsomboon et al. [185] for AR-based interfaces.

In Chapter 7, I follow the same guiding philosophy for building a VR animation interface. However, an important point of departure is that all these works assumed a well-defined set of atomic operations for eliciting related gestures, which is not desirable in our case. In this aspect, the closest exploration is that of Aigner et al. [7], which attempts to elicit gestures for complex multi-part tasks.

Another key contribution of this thesis to gestural interfaces is a taxonomy of gestures useful for

VR-based animation tasks. While I am not aware of any existing work exploring gesture taxonomies for animation authoring, I build on prior gesture taxonomies proposed for other application scenarios. McNeill's well-known classification [158] partitions gestures into *iconics*, *metaphorics*, *beats*, *cohesives*, and *deictics*. However, such taxonomies proposed in linguistics and psychology focus on gestures executed to augment speech in human communication. More relevant is recent HCI work on gesture classification by Karam and schraefel [121], Wobbrock et al. [261] and Aigner et al. [7] which classify gestures based on the level of implied abstraction. But, unlike prior works, the classification scheme used in Chapter 7 is customized for animation-specific spatiotemporal tasks.

2.2 Design Sketching

Sketching is a fundamental aid for design and communication. In the early stage of design, designers use rough ideation sketches to externalize their mental ideas and explore the design space [66]. More detailed sketches help communicate the design concept to clients and peers. Numerous psychology studies have established the importance of sketching for creativity, peer communication, and improved design outcomes [8, 134, 251]. Sketching via physical media dates back millennia, and digital sketching itself is over half a century old, going back to Sutherland's revolutionary SketchPad system [230]. Traditionally, sketches—whether created using physical media or digital tools—are two-dimensional. However, design sketches typically depict real-world, three-dimensional objects. By enabling the direct creation of 3D curves mid-air, immersive environments and mid-air interactions have therefore opened up a profound new direction for design sketching.

2.2.1 3D Sketching and Curve Creation

The traditional method for creating curves in 3D is by using a set of interface elements to specify multiple geometric constraints leading to the final 3D curve. Desktop modelling software allows users to drag control points in 3D space to create the desired curvature. Alternative interfaces in the research literature include specialized hardware such as Grossman et al.'s tape-based system [93], or using software-based interfaces such as the one proposed by Bae et al. [22] to specify projection planes. Closer to my topics of study is a class of 3D curve creation techniques that involve freehand 3D motion to specify curves. Due to the lack of 3D display technologies, early systems utilizing freehand 3D input, such as the 3-Draw tool [200] were forced to use 2D displays. More recently, 3D display and head tracking technologies have enabled the creation of complex and high quality curve networks [130, 142, 255] using mid-air 3D motion as direct input. Notably, such systems have mostly been explored for generating freeform "organic" curves and not for structured design curves. This could be due to the innate difference in human drawing ability in 2D compared to 3D, and this hypothesis forms an important motivation of the initial observation sessions presented in Chapter 3.

One of the earliest examples of immersive 3D modelling is HoloSketch [58], that supported 3D creation of primitives and freeform tube and wire geometries. Despite its novel interface, the system evaluation reported issues associated with a lack of motor control and the resulting noisy strokes plaguing mid-air 3D drawing. Motor control issues have been further corroborated by recent studies [130] conducted with contemporary hardware. Inspired by HoloSketch, CavePainting [131] utilizes a cave environment, and Tilt Brush [87] an HMD-based VR environment for 3D sketching. Despite the focus on creating 3D strokes, users naturally use multiple nearby strokes to depict surfaces. While a "guides" tool in Tilt

Brush allows the projection of 3D strokes onto a predefined set of surfaces, executing accurate 3D strokes remains challenging (see Chapter 3). Furthermore, users are restricted to the predefined surface shapes, and cannot create ones of their choice. In contrast, my novel sketching tool, presented in Chapter 4, empowers users to author 3D surfaces and detail them using a mix of 2D and 3D sketching. This makes my tool more suitable for design sketching.

Systems aimed at the creation of CAD models in VR employ various curve and surface fitting techniques for precise 3D modelling [76, 255]. Gravity Sketch [91] is a recent commercial tool with bimanual interactions to support a CAD-like workflow in VR. While these systems target precise CAD models, I present two sketching tools with different goals. For the SymbiosisSketch system described in Chapter 4, the focus is early concept design, epitomized by loosely constrained sketches with rich illustrative detail. SymbiosisSketch surfaces lie in-between surface patch networks used in CAD, and very loose hand-swept surfaces created by existing VR/AR drawing tools such as Surface Drawing [204]. On the other hand, my CASSIE concept modelling system (Chapter 6) aims at bringing detailed concept modelling closer to freehand 3D sketching. I further discuss related work in this domain in Section 2.3.

2.2.2 Immersive Sketching

Immersive creation has a long history in computer graphics. As noted above, immersive 3D sketching was pioneered by the HoloSketch system [58], which used a 6-DoF wand as the input device for creating polyline sketches, 3D tubes, and primitives. In a similar vein, various subsequent systems have explored the creation of freeform 3D curves and swept surfaces [71, 87, 131, 204]. While directly turning 3D input to creative output is acceptable for ideation, the inherent imprecision of 3D sketching is quickly apparent when more structured creation is desired.

The perceptual and ergonomic challenges in precise control of 3D input is well-known [130, 151, 152, 257], resulting in various methods for correcting 3D input. Input 3D curves have been algorithmically regularized to snap onto existing geometry, as with the FreeDrawer [255] system, or constrained physically to 2D input with additional techniques for "lifting" these curves into 3D [112, 141, 174]. Haptic rendering devices [119, 130] and tools utilizing passive physical feedback [92] are an alternate approach to tackling the imprecision of 3D inputs. Further, my experiments in Chapter 3 demonstrate the difficulty of creating curves that lie exactly on virtual surfaces in VR, even when the virtual surface is a plane. This observation directly motivates my exploration of techniques for projecting 3D strokes onto surfaces (Chapters 4, 6), instead of coercing users to awkwardly draw exactly on a virtual surface.

While my work on 3D sketching targets modern head-mounted displays, it relates to early efforts on supporting 3D drawing and painting with a variety of virtual reality devices.

2.2.3 Sketching Curves on, near, and around Surfaces

Curve creation and editing on or near the surface of 3D virtual objects is fundamental for a variety of artistic and functional shape modelling tasks. Functionally, curves on 3D surfaces are used to model or annotate structural features [81, 225], define trims and holes [211], and to provide handles for shape deformation [120, 165, 220], registration [83] and remeshing [138, 231]. Artistically, curves on surfaces are used in painterly rendering [85], decal creation [207], texture painting [5], and even texture synthesis [78]. Desktop-based curve on surface creation is typically realized via a *view-centric* WYSIWYG projection of on-screen sketched 2D strokes. That is, sketched points are raycast in the view direction, towards the

target surface. While the sketching view-point in these interfaces is interactively set by the user, there has been some effort in automatic camera control for drawing [172], auto-rotation of the sketching view for 3D planar curves [157], and user assistance in selecting the most *sketchable* viewpoints [22]. Immersive 3D drawing enables direct, viewpoint-independent 3D curve sketching, and is thus an appealing alternative to these 2D interfaces. This observation motivates my exploration into techniques for drawing curves on surfaces in VR, described in Chapter 5.

This work is also related to drawing curves *around* surfaces. Such techniques are important for a variety of applications: modelling string and wire that wrap around objects [52], curves that loosely conform to virtual objects [139], clothing design on a 3D mannequin [245], layered modelling of shells and armour [57], and the design and grooming of hair and fur [80, 205, 266]. Some approaches such as SecondSkin [57] and Skippy [139] use insights into spatial relationship between a 2D stroke and the 3D object, to infer a 3D curve that lies on and around the surface of the object. Other techniques like Cords [52] or hair and clothing design [266] present generic techniques that drape 3D curve input on and around 3D objects using geometric collisions or physical simulation. In a similar vein, my contribution in Chapter 5 addresses the general problem of projecting a drawn 3D stroke to a real-time inked curve on the surface of a 3D object. While this work does not address curve creation with specific geometric relationships to the object surface (like distance-offset curve), the proposed techniques can be extended to incorporate geometry-specific terms.

2.2.4 Evaluation of Sketching Ability

There have been many studies on human drawing ability in the fields of motor control, psychology, and HCI. Cohen and Bennett [51] attributed the misperception of the target object as a major reason for drawing inaccuracies. Schmidt et al. [208] focused on expert performance in drawing simple curves in various different projections, and found that even expert artists fail to perceive and/or draw on 2D projections of 3D objects accurately. Fitzmaurice et al. [79] discuss how artists reach optimal orientations in pen-and-paper drawing by rotating the paper, and how digital interfaces for 2D drawing can support this interaction. I present three user studies in Chapter 3 that extend these findings to freehand 3D sketching.

In the VR domain, Keefe et al. [130, 132] studied the impact of force feedback on 3D sketching accuracy by utilizing a Phantom Omni haptic device. In contrast, I compare mid-air sketching to the natural, passive feedback provided by a physical surface, thus better mimicking the traditional drawing setting. Further, unlike a Phantom Omni, this does not limit input range. My experiments also quantify the influence of various factors in isolation, and present novel results on surface-projected sketch accuracy. Jackson and Keefe [111] and Kühnert et al. [140] performed design studies to explore sketching over physical props for rapid prototyping. These works inspired and informed my exploration of visual guidance to improve sketching precision (Section 3.3). Perhaps the investigation closest in application to mine is that of Wiese et al. [257], who studied the learnability of mid-air sketching in a CAVE environment. My research contributes new findings on how factors such as physical constraints, visual guidance, orientation, and scale affects mid-air drawing ability in a head-mounted VR environment. Motor control studies, such as the one by Abend et al. [2], have studied the speed and position profiles of the human hand for various target motions. In the HCI community, various models for understanding speed and accuracy have been proposed for 2D gestures [38, 109], but work in 3D has been limited to low-level motor control evaluations [123]. Concurrent to the work presented here, Barrera-Machuca

et al. [151, 152] have also conducted fundamental explorations into precision and control issues in 3D sketching, evaluating, for instance, the geometric accuracy and aesthetic quality of shapes made of various grid-aligned lines and circles. This is related but distinct from my experiments which study sketching imprecision in lines and circles drawn in isolation. My work also relates to that of Herman and Hutka [98], who discuss the impact of such imprecision on artists transitioning from 2D sketching to VR/AR.

2.3 Sketch-based Modelling

Sketch-based 3D modelling is a rich ongoing area of research (see surveys by Olsen et al. [169] and Kazmi et al. [129]). Typically, desktop-based systems interpret 2D sketch inputs for various 3D shape modelling tasks. One could categorize these modelling approaches as single-view (akin to traditional pen on paper) [12, 40, 209, 267] or multi-view (akin to 3D modelling with frequent view manipulation) [22, 72, 73, 107, 165]. Single-view techniques use perceptual insights and geometric properties of the 2D sketch to infer its depth in 3D, while multi-view techniques explicitly use view manipulation to specify 3D curve attributes from different views. While my focus is on mid-air 3D stroke input, the inherent imprecision in 3D sketching (Chapters 3, 4), corrections required to create connected curve networks (Chapter 6), and the ambiguity of projection onto surfaces (Chapter 5) connects it to the interpretative algorithms designed for sketch-based 3D modelling. My tools aim to take advantage of the immersive interaction space by allowing view manipulation as and when desired, independent of geometry creation.

2.3.1 Immersive 3D Modelling

The 3-Draw system [200] decouples the act of drawing the curve from the act of positioning it with respect to other curves. Keefe et al. [130] use haptic feedback from a Phantom device and a two-hand interaction metaphor inspired by tape drawing [27] to separate drawing the curve from indicating its tangent directions. Other two-handed systems simulate deformable rods that users bend to model shape outlines and sharp surface features [250], or rely on a physical strip of sensors that provides bending and twisting controls on 3D curves [93]. These tools illustrate various solutions that have been proposed to overcome the challenges to VR sketching and aid 3D shape creation.

My CASSIE system (Chapter 6) addresses these challenges by automatically correcting imprecise strokes and by visualizing surfaces in-between strokes. This work is closer in spirit to FreeDrawer [255], where users first sketch a sparse network of feature curves, and then manually indicate curve cycles to form surface patches. Spacedesign [75] implements a similar concept but sketching is constrained to 2D curves projected onto virtual or physical planes. While CASSIE shares the design philosophy of FreeDrawer and Spacedesign, it utilizes a novel 3D optimization framework for curve network creation, as well as an algorithm for progressive, automatic surface patch creation, on current VR hardware. More importantly, I report on a comprehensive user study with design professionals and amateurs to evaluate the impact of these curve network and surface features on ideation and concept modelling in VR/AR.

This approach to 3D stroke neatening is inspired by sketch beautification and regularization frameworks originally developed to process 2D diagrams and drawings created with a computer mouse. Pavlidis and Van Wyk [181] introduced one of the first such systems, that takes a line drawing as input and outputs a drawing close to the input while satisfying geometric constraints. While this system was applied as a post-process on complete drawings, Iragashi et al. [106] proposed an interactive beautification

method that treats each new stroke as it is sketched. This iterative approach can cope with more complex sketches and geometric relationships between strokes. The interactivity of the system also gives more control to the user who can choose between multiple possible beautified results. More recently, Fišer et al. [77] proposed an interactive method called ShipShape, that supports Bézier curves as input. CASSIE draws inspiration from their approach, although I focus on a small set of geometric constraints tailored to the most frequent sensorimotor errors observed in 3D sketching, while they consider a larger variety of rules to beautify 2D drawings. In addition, while Fišer et al. [77] beautify each stroke by applying geometric rules in sequence, my technique casts stroke neatening and structuring as an energy minimization that balances the concurrent application of multiple geometric constraints with preservation of the input curve. This formulation is inspired by sketch-based modelling algorithms that seek to lift 2D strokes to 3D by enforcing geometric constraints, while making sure that the resulting curves re-project well on the input drawing [57, 209, 214, 267]. In particular, the interactive system by Schmidt et al. [209] lifts each new stroke by snapping it to the 3D curves inferred from previous strokes, which they achieve by balancing the satisfaction of snapping constraints with re-projection error.

2.3.2 Creating and Surfacing 3D Curve Networks

Surfaced curve networks form a compact and descriptive representation of 3D shapes [90], which has motivated the development of algorithms to automatically generate surfaces from sparse, designer-drawn 3D networks. A first challenge is to identify, among all closed cycles in the curve network, which cycles bound surface patches rather than internal cross-sections. Treating this problem globally is difficult due to the large search space of all possible curve cycles in a network, and due to the inherent ambiguity of the task [1, 201, 275]. Other approaches rely on assumptions about the network topology; for example, Orbay and Kara [171] assume that the curves form a connected graph. For my interactive curve neatening and surfacing scheme proposed in Chapter 6, I instead leverage the iterative nature of the workflow to surface the network progressively, only performing a local search for cycles around each newly-added stroke. The interactive context also allows us to let users add or remove cycles that might have been misinterpreted by the automatic algorithm. A second challenge is to generate the surface geometry that interpolates the cycle boundaries. I adopt the method by Zou et al. [276] for this task, which applies a dynamic programming algorithm to generate a triangulation that satisfies various geometric criteria. Alternative solutions include the generation of a quad mesh aligned with the input curves [32]. Finally, various surface fairing algorithms have been proposed to improve the quality of the generated surfaces [176, 226], and could be applied to CASSIE results in a post-process.

2.3.3 Modelling in Context

Objects are designed to serve a function, often interacting with people and objects in an environment. A number of 3D sketching systems [42, 43, 57, 117, 128, 137, 147, 174, 274] leverage existing visual media, 3D models, or spatial information of the environment to inspire and guide the design.

The success of a 3D sketching system using sketch planes depends to a large extent on the speed and accuracy with which users can choose desired sketch planes [23, 128]. A physical tablet has been shown to be effective in mapping directly to a subset of physical sketch planes in 3D [143]. In Chapter 4, I propose an indirect mapping of the tablet in AR to capture physical planes in arbitrary position, orientation, and scale, allowing users to draw comfortably in situ.

2.4 Animation Authoring Interfaces

While a large portion of the work described in this thesis is on immersive sketching and modelling, Chapter 7 focuses on the use of hand gestures for animation authoring. Here, I briefly describe techniques and interfaces for animation authoring and in particular for immersive animation.

2.4.1 Natural Interfaces for Animation

Most professional animators use specialized tools with complicated interfaces full of sliders and numeric entries for controlling thousands of different commands and parameters. An alternative approach is *performance-driven* animation [3, 97, 183, 264], which tracks human actors and maps their motion to digital characters. Over the last decade, researchers have also explored *direct manipulation* interfaces to make animation easy and accessible [56, 108, 125, 126, 127, 241, 265], allowing amateur users to rapidly adapt to their animation creation tools. Interestingly, the use of direct manipulation as well as performance for animation was pioneered in the 1960's in the foundational Genesys system [24].

Spatial and gesture-based interfaces have been defined for more restricted animation tasks. For instance, using Finger Walking [149] to author walking/running motion, motion capture widgets [62] to control a restricted set of DoFs of a biped or quadruped character, full-body motion [39] to animate non-humanoid characters with various topologies, and finger movements [164] to manipulate deformable, rigged drawings. While restricted to specific sub-domains of animation, these works show that gestures and spatial inputs can be used to simultaneously control multiple DoFs in animated scenes. However, in these cases, the mapping between the input and animation parameters is natural and well understood. In contrast, the mapping from hand gestures to the control of complex animated phenomena is unclear, and motivated me to conduct a new gesture elicitation study (Section 7.1). This work is also related to that of Jensen et al. [115], who extract physical properties of virtual objects from demonstrations.

Commercial VR animations tools, such as Quill [71] and Tvori [246], apply performance-based direct manipulation to animated strokes and articulated skeletons using hand-held controllers. In contrast to direct manipulation, the communicative aspects of hand gestures to articulate dynamic, physical phenomena remains an open and challenging question. In computer graphics, the development of algorithms to control and guide physical simulations—such as deformation, collision, flow, and fracture—is a long-standing area of research [25, 31, 177]. In Chapter 7, I investigate the mapping between spatial interactions (hand gestures) and physical animation parameters, and provide guidelines for future spatial animation interfaces.

2.4.2 Immersive Animation

Unlike the extensive work on 3D sketching and modelling in VR/AR, the literature on immersive animation tools is rather sparse. Nevertheless, the venerable HoloSketch system [58], already mentioned a few times in the previous sections, included basic tools for animating the sketched models. The interface focused on predefined motion widgets encoding transformations such as periodic rotational motions. In addition, curve primitives could define motion paths for other objects. Thalmann [234] discusses the use of various input mechanisms for driving VR animations, such as a tracked HMD for driving the camera motion trajectory and force transducers for physics-based animation. Recent commercial systems focus on performance-based animation [71, 167, 246] but rely on standard 6-DoF controllers and 3D widgets. These systems inspired my work on performance-based animation using hand gestures.

2.5 Design and Analysis of HCI Experiments

In this section, I will provide a brief introduction to the typical processes utilized for designing and analyzing quantitative experiments in HCI. Such experiments are utilized throughout this work, including in Chapters 3, 4, 5, and 6. The intention is to introduce these concepts to a reader who is not well-versed with quantitative experiments with human subjects. For an authoritative account, the reader is referred to the edited volume by Robertson and Kaptein [193] for an HCI-focused account and Wilcox's book [259] for a detailed description of hypothesis testing and the various statistical tests employed in the general scientific community.

2.5.1 Experiment Design

In a typical quantitative experiment in HCI, the experiment designer wishes to test whether two different experimental conditions *differ* from each other. For example, the creator of a new pointing device would design an experiment to test whether the new device is demonstrably better or worse than a mouse. The designer needs to determine a representative task which can reasonably be achieved using both devices, such as dragging and dropping icons from one window to the other. The choice of the experiment task is influenced by what the experimenter desires to measure. If he wishes to show that the new input device increases users' efficiency, the time taken to perform the task can be a reasonable measure. However, if the target is to demonstrate that the new device is ergonomically superior, then the measurement as well as the actual experiment task may need to be changed. Please note the the experiment variables are also called as independent variables or factors, and the measurements are also known as dependent variables. Different possible values taken by a factor are interchangeably called as conditions, levels, or values.

Once the task, the factors, and the measurement are determined, the experimenter can choose to design the experiment as a *between-subjects study*—where two different groups of users are selected and each device is tested by a group, or a *within-subject study*—where the same group of users tries out both the devices in sequence. All the quantitative experiments reported in this dissertation employ a within-subject design.

Of course, HCI experiments are not always designed to distinguish between two possible values of an experiment variable. Experiment variables, such as the pointing device in our running example, can have more than two values. Further, a single experiment may be designed to measure the influence of multiple variables. For example, the pointing device experiment may involve a graphic tablet as an additional device, and may consider dragging-and-dropping across targets of different sizes, respectively. Finally, multiple measures may be recorded for later analysis. For instance, in addition to measuring the time taken for the task, the experimenter may wish to use the same experiment to gain a measure of subjective user satisfaction. This can be achieved by presenting the user with a subjective survey question at the end of the experiment, where they choose a value among 'Very satisfied', 'Somewhat unsatisfied', 'Neutral', 'Somewhat satisfied' and 'Very satisfied' on a subjective rating scale. Such a scale is called a *Likert scale*.

In addition to defining the measurements, the experimenter needs to define one or more *null hypotheses* for the experiment. The typical null hypothesis in an HCI experiment is that all the tested conditions are the same. That is, the recorded measurement(s) do not differ between the conditions to a statistically significant degree. Then, the experimenter employs a statistical test to determine if the null

hypothesis can be rejected. In general, the researcher desires to show that the fancy new device, system, or algorithm they designed is indeed better than the state of the art, and therefore, the researcher would like to reject the null hypothesis. As HCI researchers, we need to be careful that our personal desires do not hijack the experiment design and we should always strive to follow the best principles for experiment design and analysis.

2.5.2 Experiment Analysis

Now we will look into various statistical tests for hypothesis testing. I will first formalize the notion of hypothesis testing. Without loss of generality, let us consider the case with a single experiment variable with two levels. Assume that n users tried the first condition and the measurements recorded were $\mathbf{x} = [x_1, \ldots, x_n]$, and m users tried the second condition and the recorded measurements were $\mathbf{y} = [y_1, \ldots, y_m]$. The null hypothesis states that the elements of $\mathbf{x}$ and $\mathbf{y}$ are sampled from the same underlying (but unknown) distribution. The statistical tests we will look at result in the probability, informally called the p-value of the test, that the two sets of samples are from the same distribution. In the HCI community, the generally accepted p-value for rejecting the null hypothesis is 5%. Thus, an outcome with $p < .05$ is termed to be a *statistically significant* one.

Broadly, statistical tests for hypothesis testing can be categorized either as *parametric*—which work under the assumption that the underlying probability distribution can be modelled with a fixed set of parameters, or as *non-parametric*—which make no such assumption. We will consider two classes of parametric tests, the Student's t-test and the Analysis of Variance (ANOVA), and one non-parametric test: the Wilcoxon signed-rank test.

Student's t-test. The t-test is used to check whether the difference between the means of two populations, such as $\mathbf{x}$ and $\mathbf{y}$ above, is a meaningful one. The test assumes that the samples from both the populations are distributed normally, and have similar variances. For hypothesis testing, a statistic, called the t-statistic is computed from the samples, and its value is compared to a one-parameter distribution called the Student's t-distribution. Similar to the normal distribution, the t-distribution is bell-shaped with a maximum at zero. Therefore, as a rule of thumb, higher absolute values of the t-statistic indicate a higher probability of rejecting the null hypothesis.

In a within-subject design, m is always equal to n. More importantly, the samples are *paired*. That is, corresponding to the i^{th} user, we have the samples x_i and y_i. In such a paired-sample scenario, a variant of the t-test, called the paired-samples t-test must be employed. Here, the t-statistic is computed based on the sample deviations $d_i = x_i - y_i$.

ANOVA. The ANOVA is an omnibus parametric test, extremely popular in the HCI community. As noted above, the t-test is limited to comparing between two levels of a single variable. If multiple levels or multiple experiment variables need to be considered, an ANOVA can be employed to test whether any of the experiment variables has a statistically significant effect on the measurement. Similar to the reliance of the t-test on the one-parameter t-distribution, the ANOVA computes a test statistic and compares it to a two-parameter F-distribution. Further, analogous to the use of a paired t-test, within subject designs require the use of a repeated measures ANOVA, abbreviated as RM-ANOVA or rANOVA.

Wilcoxon signed-rank test. Parametric tests such as the t-test or ANOVA place additional assumptions on the data. First, the dependent variable (the measurement) must admit arithmetic operations of addition and subtraction, allowing one to define the mean of the samples. This assumption is broken by, for instance, Likert scale data where the various possible responses are ordered, but addition or subtraction cannot be defined. Even in the cases where the first assumption is met, other assumptions such as normality or similar variance among populations may not be met. In such cases, we can employ non-parametric tests which do not rely on the same assumptions. A widely-used non-parametric test for paired samples is the Wilcoxon signed-rank test, which ignores the actual values of the samples, but only relies on how a paired sample is ordered; whether $x_i < y_i$, $x_i > y_i$, or $x_i = y_i$. After computing the test statistic, the probability of rejecting the null hypothesis can be computed by considering the likelihood of the observed statistic under the null hypothesis expectation that the orderings $x_i < y_i$ and $x_i > y_i$ are equally likely. In practice, the statistic is compared against a standard table of critical values.

Chapter 3

Mid-Air Drawing: Opportunities and Limitations

We started our discussion with the creative potential of mid-air interactions for 3D creation. In this chapter, we will dive deeply into a most basic mid-air interaction—drawing a 3D stroke that traces the motion of a device tracked in 3D space. We will delve into the expressive power of mid-air 3D sketching and compare its expressiveness to traditional sketching bound to a plane. We will also explore the novel problems encountered by artists when creating directly in 3D. Recognizing both the advantages of 3D sketching a nd the challenges associated with it will allow us to pinpoint the design applications which stand to benefit from this novel sketching paradigm. This in turn will guide the design of creative tools that help their users avoid these challenges while taking full advantage of the novel creative possibilities. The development of three such tools will be described in the later chapters (Chapters 4, 5, and 6) while this chapter explores the fundamentals of 3D sketching.

Sketching is a canonical task in many visual design pipelines due to its freeform and expressive nature. Since freehand sketching provides an intuitive method of conceptualizing ideas, there has been considerable research [26, 57, 107, 205, 267, 274] into using freehand sketches to create three-dimensional artifacts by lifting 2D sketches into the third dimension. The HCI community has complemented this research by producing interfaces that allow sketching directly in 3D [92, 131, 232]. Prior works [110, 232]

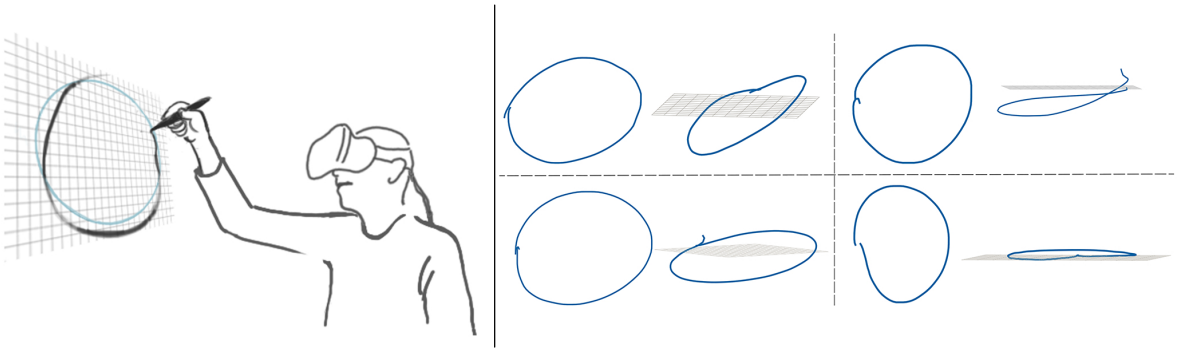


Figure 3.1: Virtual Reality affords the freedom to sketch in unconstrained 3D spaces (left). However, our experiments indicate that drawing accurately in VR is challenging. Inaccuracies in depth as well as the target planar projection (right) are common.

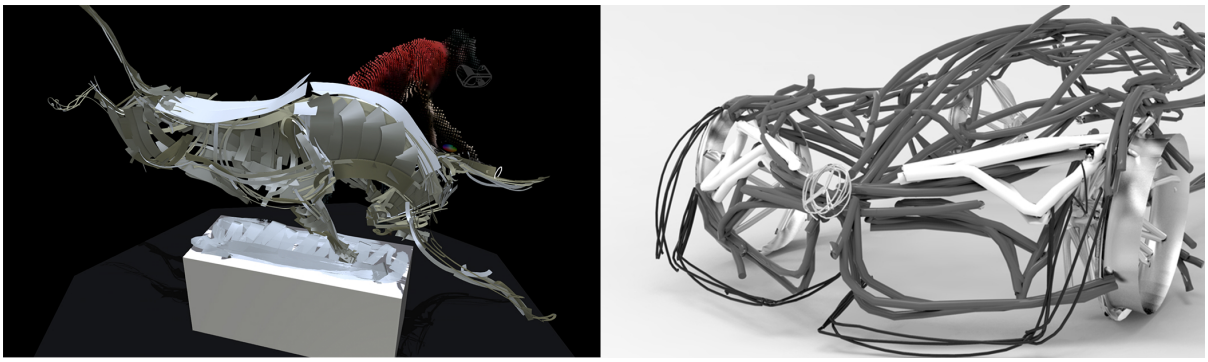


Figure 3.2: VR sketching using consumer-grade tools—Google Tilt Brush (left) and Gravity Sketch (right). © Google (CC BY 3.0) and Gravity Sketch (used with permission).

indicate that professional designers are excited by the possibility of using direct 3D input for sketching, but find it difficult, and get frustrated by the lack of control over their strokes (Figure 3.1). While there have been some quantitative studies exploring how 3D sketching capabilities improve with learning [257] and how force feedback affects precision [130], there has been limited work that evaluates and quantifies the factors which influence drawing in unconstrained, mid-air environments. In this chapter, I describe a set of formal experiments to study the human ability to draw in mid-air VR environments, and explore the impact that physical and visual guidance can have on 3D sketching accuracy. I first conducted a series of observational sessions with professional designers to understand how professionals approach mid-air sketching. Based on these sessions, I formulated an experiment aimed at quantifying drawing accuracy in VR and comparing it to traditional 2D drawing by switching between the presence and absence of a physical drawing surface, in three canonical orientations. I then conducted a second experiment to observe how visual guidance provided by the drawing system in VR impacts drawing accuracy for both planar and non-planar curves, across a greater range of surface orientations.

The first study showed that mid-air drawing accuracy in VR, measured via the mean deviation from a target stroke, decreased by 148% compared to traditional 2D drawing. However, by including a physical drawing surface within a VR environment, this loss of accuracy was reduced to 20%. The second study found that visual guidance can serve to improve mid-air sketching accuracy, but leads to worse aesthetic quality of strokes, as measured via curve fairness [74]. In both studies, surface orientation was a key factor of the above accuracy levels. Based on these results, I present a broad set of interaction guidelines that can help guide the future design of VR-based design tools.

3.1 Initial Observation Sessions

Before conducting the two controlled experiments, I first directly observed artists working within a VR 3D sketching system. The goal of these sessions was to obtain feedback on the challenges and opportunities of 3D freehand sketching. The resulting observations were then used to guide the areas of focus for the quantitative evaluation. I invited five expert designers (two industrial designers, two concept artists, and an architect) to participate in a design session using Tilt Brush [87] (v5.4), a VR sketching application. An HTC Vive [103] device was used for the sessions. Participants were asked to create any 3D sketch related to their domain of expertise in a 60-minute session.

3.1.1 Observations

In general, the participants were positive about the ability to draw in scale, directly in 3D, utilizing the immersive and freeform nature of VR sketching. Artists drew conceptual models of edge-heavy objects such as cars, an interior design of a room, as well as organic shapes such as shoes and humans. We observed that the drawing characteristics seemed to vary across various positions and orientations of the drawing plane. Participants stated that accuracy and precision of the strokes in 3D VR environments were more critical, compared to its 2D counterparts:

"In 3D situations, line accuracy is more important than 2D situations, since you're conveying more information to the viewer. In 2D, the viewer needs to make a mental leap to go from 2D to 3D. On the contrary, in 3D, because you get more info, you're looking more precisely at the quality of the line." *(P2)*

In general, participants felt that it could be challenging to depict a desired shape, noting in particular that curves that were meant to be planar often ended up as convex (P2, P5). Participants also felt they could not always achieve the intended result due to inaccurate positioning (P1, P2). They unanimously agreed that additional tools were required for precision and to improve stroke control for meaningful design tasks in VR. Participants suggested adding snapping tools (P2, P3), haptic feedback (P1, P4), and projection to existing planes and surfaces (P2, P3, P5). Some participants (P2–P5) were concerned about the possibility of ergonomic issues such as neck and shoulder pain when using VR over an extended period.

These observations are consistent with those that can be made from existing repositories of VR sketching workflows [88]. While artists were able to design successfully, there were certainly struggles when trying to depict a desired curve accurately in 3D space. In the following sections, I present two controlled experiments conducted to gain a better understanding the human limits of freehand drawing, which would eventually lead to design recommendations for VR sketching systems.

3.2 Experiment 1: VR vs. Traditional Drawing

While the ability to draw non-planar curves makes mid-air sketching unique, planar curves play an integral role for both shape perception and within the 3D design process, as evidenced by numerous previous works [22, 87, 157], as well as the initial observational sessions. In the first experiment, I sought to investigate how mid-air drawing ability of planar curves in VR compares to traditional drawing on a flat surface. In particular, I wanted to study how the presence or absence of a physical drawing surface affects drawing of planar curves. To this end, the participants were exposed to three main experimental conditions. In the *traditional* condition, participants drew on a physical surface. In the *VR* condition, participants used a VR head-mounted display and drew in 3D space. In the *hybrid* condition, participants drew on a physical surface while using the same VR headset.

3.2.1 Experimental Design

The experiment was designed as a $3 \times 3 \times 3 \times 3$ within-subjects study, with independent variables of *condition* (*traditional*, *hybrid*, *VR*), *drawing plane* (*horizontal*, *vertical*, *sagittal*), *shape* (*u-line*, *v-line*, *circle*), and *size* (*small*: 10cm, *medium*: 30cm, *large*: 60cm). The three drawing plane configurations are illustrated in Figure 3.3a. All of these planes were located at a comfortable position [34, 178]. For each of

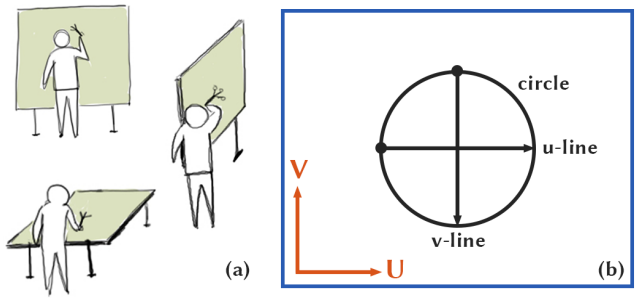


Figure 3.3: Alignment of the three drawing planes (a) clockwise from top: *vertical*, *sagittal*, and *horizontal*; and arrangement of the three stroke shapes (b) in Experiment 1.

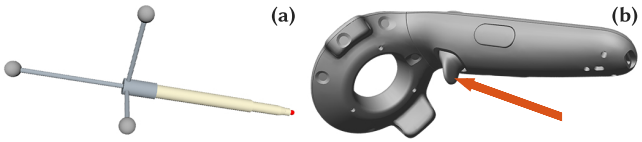


Figure 3.4: Apparatus: The motion-captured pen (a) and the HTC Vive controller (b) used for drawing.

these planes, we defined *U-V* coordinate axes that define the *u-line* and *v-line* orientations (Figure 3.3b).

Participants performed the experiment in a single session lasting 40–50 minutes. The order in which participants were exposed to each condition was counterbalanced via a Latin square. Within each *condition*, the order of appearance of the *drawing planes* was again dictated by a Latin square per participant. For each plane orientation, the stroke *shape* order was randomized. For each stroke *shape*, participants had to complete three sets of trials, each of which was a random permutation of the three stroke *sizes*. Overall, each participant drew 243 strokes. In addition, for each *condition*, participants drew a medium-sized stroke for each *shape* for practice.

3.2.2 Participants

In total, 12 able-bodied individuals (8 male, 4 female), aged 22 to 51, participated in the study. All participants were right-handed or ambidextrous, and used their right hand to draw. Participants were 164–183cm tall, and none had sketching experience using VR/AR devices, or professional drawing experience. Participants were paid for their time.

3.2.3 Apparatus

In the *traditional* condition, participants had access to a large (84") display, which showed the target strokes and was used as a physical drawing surface. Strokes were drawn using a dry-erase pen augmented with motion-capture markers (Figure 3.4a). However, data was only recorded when the participants pressed a trigger on a HTC Vive controller (Figure 3.4b) held in their left hand. In the *hybrid* condition, participants wore the HTC Vive HMD, and the large display was still used as the drawing surface. Simplified 3D models of the display and the tracked pen were rendered via the HMD. In the *VR* condition, no drawing surface was used or showed, and a matte grey background was rendered. In all the three conditions, the pen position and orientation were tracked at 60 Hz using OptiTrack motion capture cameras [170]. The HTC Vive [103] display had a refresh rate of 90Hz. The software for the

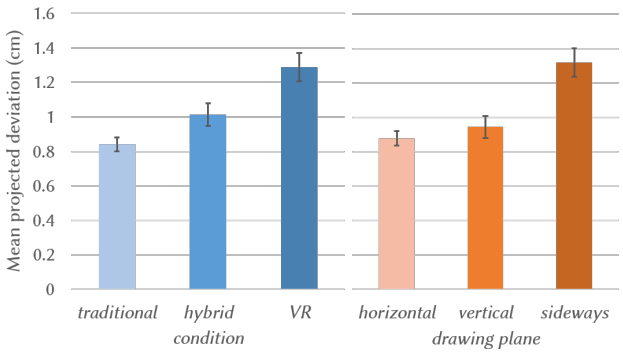


Figure 3.5: Main effects of *condition* (left) and *drawing plane* (right) for mean projected deviation.

experiment was implemented in C# using Unity [247] and SteamVR [248] for rendering and interaction, and OptiTrack's Motive software [170] for motion capture.

3.2.4 Procedure

Participants were instructed to stand at a fixed spot before starting to draw in a new drawing plane to ensure controlled results. Without this constraint, participants would be able to change their orientation relative to the drawing surface, thereby confounding the results. For the *horizontal* orientation, the drawing surface was kept at a height of 1m, while for the other two orientations, the surface was arranged so that the centre of the stroke was at a height of 1.4m. Following ergonomic guidelines [178], the target strokes were centred approximately 40cm in front of participants.

The trial started with the participant being shown the target stroke along with the starting point, both rendered in black. Participants were instructed to draw all circles in a clockwise direction, while the starting spots on lines were chosen so that the horizontal lines were drawn left-to-right or far-to-near, and the vertical lines top-to-bottom. The target stroke was shown until the participant pressed the trigger in their left hand to initiate drawing. Participants were told to execute the strokes as quickly and accurately as possible. Strokes were rejected automatically if any point on the input was over 20cm away from the target, and the trial was repeated.

For the *traditional* condition, the target strokes were shown on the large screen, and the marker left a visible ink trail. For the *hybrid* condition, the screen was tracked by motion capture markers, and rendered virtually at the same position in space so that participants actually drew on the physical surface. In the *VR* and *hybrid* conditions, participants used the same pen to draw, but the target and drawn strokes were shown via the HMD.

In the *traditional* condition, an experimenter erased the stroke drawn by the participant after each trial. In the *hybrid* and *VR* conditions, the input stroke was hidden 1 second after the participant finished drawing it.

3.2.5 Data Preparation

Strokes were recorded as sequences of 3D points sampled at 60Hz. The points were initially transformed to a local coordinate system defined relative to the target stroke. This coordinate system is defined for lines such that the line starts at $(0, 0, 0)$ and ends at $(l, 0, 0)$ and for circles such that the target stroke

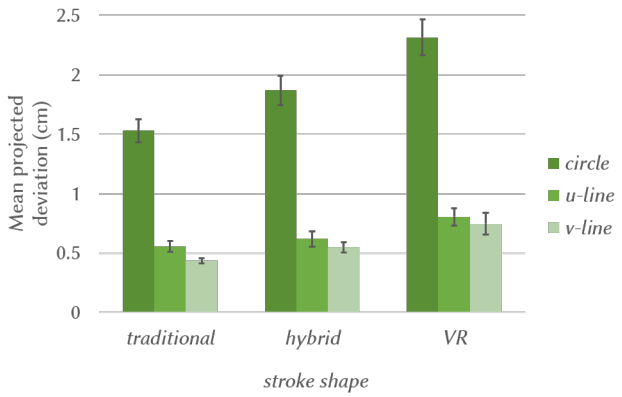


Figure 3.6: *Stroke shape* × *condition* interaction on mean projected deviation.

is centred at $(0, 0, 0)$ and the starting point is at $(-l/2, 0, 0)$, where l is the value of the *size* variable. Further, the local XY-plane for the stroke coincides with the drawing plane.

To preprocess the input data before error measures were computed, a median filter with a window size of 6 points (equivalent to 100ms) was applied to filter out any high frequency tracking noise. Some of the strokes in *traditional* and *hybrid* conditions had small tails sticking out of the plane at either ends of the stroke due to slight difference between the participants' intention of beginning/ending drawing and actually pressing/leaving the ON/OFF switch. These tails, found via deviation from mean z-coordinate of the stroke, were removed before further processing. Then, piecewise linear arclength approximation was used to resample all input strokes to 100 equidistant points. Finally, all the strokes were translated such that the starting point matched the starting point of the target stroke. This was done to minimize the impact of positional error caused by participants misjudging the exact position of the target stroke, as well as equipment error such as minor inaccuracies in calibrating the coordinate axes of the motion capture system and the HMD.

We denote a resampled and translated stroke as $\mathcal{P}$, represented via the sequence of points $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{100}$, where each point $\mathbf{p}_i = (p_i^x, p_i^y, p_i^z) \in \mathbb{R}^3$.

3.2.6 Measurements

Repeated Measures ANOVAs were performed to analyze various error measures. All post-hoc pairwise comparisons were performed using Bonferroni-corrected paired t-tests. The following measures were computed.

Mean Overall Deviation

Mean deviation is defined as the average distance of the (resampled) points of user-drawn stroke from the target stroke. Due to the resampling and positional correction in the preprocessing steps, this measure effectively computes the average distance of the drawn stroke from the target stroke. For a line, this is therefore defined as the average deviation from the local X-axis.

$$\mathrm{OD}_L(\mathcal{P}) = \frac{1}{n} \sum_{i=1}^{n} \sqrt{(p_i^y)^2 + (p_i^z)^2} \tag{3.1}$$

For circles, it is the average deviation from the target circle.

$$\mathrm{OD}_C(\mathcal{P}) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\left(\sqrt{(p_i^y)^2 + (p_i^z)^2} - \frac{l}{2}\right)^2 + (p_i^z)^2} \tag{3.2}$$

Mean Projected Deviation

Since two of the conditions involved a physical constraint that would eliminate any errors in the z direction, a more informative measure of accuracy can be computed from the deviation from the target after the drawn shape is projected onto the drawing plane. This projection is equivalent to simply setting the z term in the mean overall deviation equations to zero. Note that for *traditional* and *hybrid* conditions, overall and projected deviations are equivalent, as p_i^z is always zero.

3.2.7 Results

Mean Overall Deviation

The *condition* variable had a significant effect on the mean overall deviation ($F_{2,22} = 69.6, p < .001$). Statistically significant effects were also observed for *drawing plane* ($F_{2,22} = 8.09, p < .005$), *shape* ($F_{2,22} = 203, p < .001$) and *size* ($F_{2,22} = 169, p < .001$). The value for the *VR* condition (M = 2.08cm, SD = 1.71cm) was much higher than that for the *hybrid* (M = 1.01cm, SD = 1.09cm) and *traditional* (M = 0.84cm, SD = 0.87cm) conditions.

Post-hoc pairwise comparison showed that all three conditions were significantly different, with *VR* exhibiting over twice the inaccuracy of either of the other two conditions, showing that moving into the third dimension adds a large amount of inaccuracy. For brevity, I will skip further discussion on this measure and focus on mean projected deviation.

Mean Projected Deviation

Significant main effects were observed for all four factors—*condition* ($F_{2,22} = 23.8, p < .001$), *drawing plane* ($F_{2,22} = 27.5, p < .001$), *shape* ($F_{2,22} = 173, p < .001$), and *size* ($F_{2,22} = 133, p < .001$). Interaction effects were observed for both *condition* × *shape* ($F_{4,44} = 6.89, p < .001$) and *condition* × *size* ($F_{4,44} = 9.60, p < .001$), but not for *condition* × *plane* ($F_{4,44} = 1.43, p = .24$).

The *VR* condition resulted in the highest deviation (M = 1.29cm, SD = 1.32cm) followed by *hybrid* (M = 1.01cm, SD = 1.09cm), and *traditional* (M = 0.84cm, SD = 0.87cm). Post-hoc pairwise comparison showed significant differences between all *conditions* ($p < .05$) (Figure 3.5, left). Stated differently, there was a 20.2% decrease in accuracy by wearing an HMD and rendering the environment virtually, and a further 27.2% decrease by removing the supporting physical surface. These results demonstrate that the difficulty of in-air 3D sketching is two-fold, impacted by both motor and visual limitations. Visually, the user no longer has direct line of sight to their hand and pen, and must rely on virtual depth perception cues. From a motor standpoint, the lack of a physical surface introduces even greater inaccuracies.

With respect to orientation, the *sagittal* drawing plane had the largest mean projected deviation (M = 1.32cm, SD = 1.30cm), and was significantly different ($p < .01$) from both the *horizontal* (M = 0.95cm, SD = 0.94cm) and *vertical* (M = 0.88cm, SD = 0.98cm) planes which were not significantly different from each other (see Figure 3.5, right). The irregular movements required to draw on the *sagittal* drawing plane likely contributed to these results.

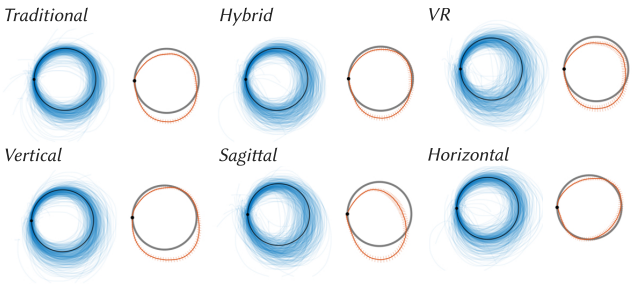


Figure 3.7: Visualization of projected deviation for *circles*. Circles drawn in different *conditions* (top), and *drawing planes* (bottom). For each row, all circles (left), and average circle with 95% confidence interval (right) are shown.

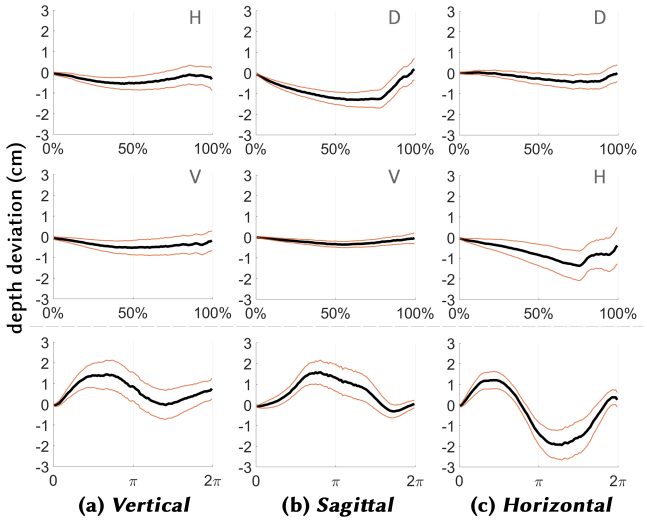


Figure 3.8: Depth deviation trends for different *drawing planes* and *shapes*. Row 1–2: lines, row 3: circles. Key for lines—H: horizontal left-to-right, V: vertical, and D: horizontal (depth) far-to-near.

Figure 3.6 shows the effects of *shape* for each *condition*. As one may expect, the *circle* caused the most difficulty, (M = 1.90cm, SD = 1.38cm) as compared to one-dimensional lines (combined M = 0.62cm, SD = 0.62cm). It can be seen that the effect is amplified in the *VR* condition, as indicated by the significant *condition* × *size* interaction. All values of *size* were significantly different ($p < .001$), with the means following the order *small*: 0.54cm (SD = 0.51cm), *medium*: 0.94cm (SD = 0.88cm), and *large*: 1.65cm (SD = 1.46cm). This indicates that drawing a stroke of larger size is inherently more difficult, hinting at a trend similar to that observed by Cao and Zhai [38] for 2D gestures of much smaller sizes. The trend could be due to the increasing effect of the stroke being dictated by the natural arcs of arm movement.

Shape Trends

Beside aggregated measures of accuracy, it is also interesting to observe trends in deviation from target as participants progressed through the strokes. We visualize mean projected deviation trends for circles in Figure 3.7 (top). A noticeable effect is the consistency of the average circle across the *conditions*. In contrast, the average circles for the three *drawing planes* are visibly different (Figure 3.7 bottom). This

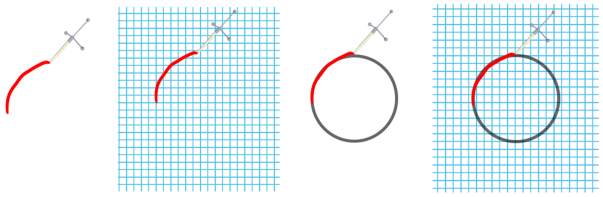


Figure 3.9: Visual guidance variants tested in Experiment 2. Left to right: *none*, *surface*, *stroke*, and *both*.

may be due to different muscle groups needed to draw on each of the planes.

For the *VR* condition, we can also look at the deviation in depth across the stroke progress. Figure 3.8 illustrates this effect of drift for each orientation and shape. In particular, it can be seen that the same lines drawn in different imaginary planes have very different characteristics. This may be an effect of pen grip, which was informally observed to change with the drawing planes, even in the *VR* condition.

3.3 Experiment 2: Factors of VR Drawing

The first experiment showed that there are both motor and visual challenges involved with drawing in VR. In this second experiment, I focus on the VR drawing condition in greater depth, and determine if some of the challenges can be alleviated with enhanced visual guidance. Expanding on Experiment 1, I also examine a larger variety of plane orientations and the effect of drawing non-planar strokes.

3.3.1 Visual Guidance

A central challenge of working in 3D is being provided with adequate depth cues and visual feedback. I therefore explored combinations of two forms of visual guidance resulting in four visual guidance conditions (Figure 3.9).

Surface Grid

Previous work in sketch-based modelling uses rendered surfaces to help users anchor their strokes in 3D [22]. I sought to examine the effect of visual guidance plane on the characteristics of sketched strokes. I use a grid line pattern as the rendered surface, as prior research [11, 26] suggests it as an effective texture for the perception of a surface.

Target Stroke

A more detailed form of visual guidance is to render and trace over the actual target stroke. In VR, this condition simulates drawing over virtual objects, inspired by existing literature [111]. Examples of this task include marking out boundaries of CT scans [122], or using an imported image or model as scaffolding to guide freehand strokes [92].

In addition to these two guidance conditions, I also tested a condition that combined them (*both*), and a condition that provided no visual guidance (*none*).

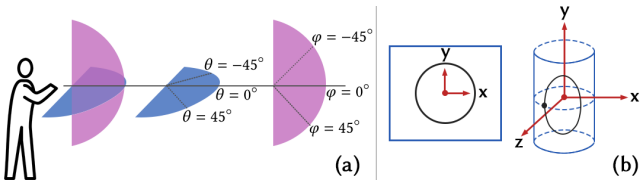


Figure 3.10: Orientations (a) and surface shapes: flat and curved (b) used in the Experiment 2.

3.3.2 Drawing Surface Orientation

To gain a deeper understanding of how the orientation of the drawing surface affects accuracy, I tested 13 different surface orientations. I positioned the drawing surfaces tangential to a hemisphere of radius 40cm, placed approximately 15cm in front of the participant's head. The orientation is defined using the pair of angles φ, θ by which the plane was rotated around the centre of the hemisphere along the vertical (down-to-up) and horizontal (left-to-right) axes, respectively (Figure 3.10a). Both φ and θ were defined to be zero for the vertical plane in front of the participant. The various orientations studied were

$$(\varphi = 0°, \theta \in \{0°, 45°, -45°, 90°, -90°\}),$$
$$(\varphi = 90°, \theta = 0°), (\varphi = -90°, \theta = 0°),$$
$$(\varphi = 45°, \theta \in \{0°, 45°, -45°\}), (\varphi = -45°, \theta \in \{0°, 45°, -45°\}).$$

3.3.3 Experimental Design

The experiment was designed as a $4 \times 13 \times 2$ within-subject study. The independent variables were *visual guidance* (*none*, *surface*, *stroke*, *both*), *surface orientation* (see above) and *surface shape* (*curved* or *flat*). The *curved* surface consisted of a cylindrical surface of radius $R_0 = 15cm$ (Figure 3.10c). The target shape was fixed as a 30cm circle.

The experiment was divided into four blocks, one for each of the *visual guidance* conditions. The ordering of *visual guidance* was counterbalanced using a balanced Latin square. For each level of *visual guidance*, the participants were exposed to 13 orientations of the drawing surface, ordered randomly. For each of these *orientations*, they had to complete three sets of trials, each of which involved drawing two circular strokes—one on a planar surface (*flat*), and the other on the cylindrical surface (*curved*). The surface shapes were ordered randomly. Overall, each participant executed 312 strokes, in a single session lasting 50–80 minutes.

3.3.4 Participants

We recruited 12 able-bodied participants (8 male, 4 female), aged 22 to 53. None of the participants had experience using VR/AR for sketching, or professional experience sketching or painting. Participants were paid for their participation.

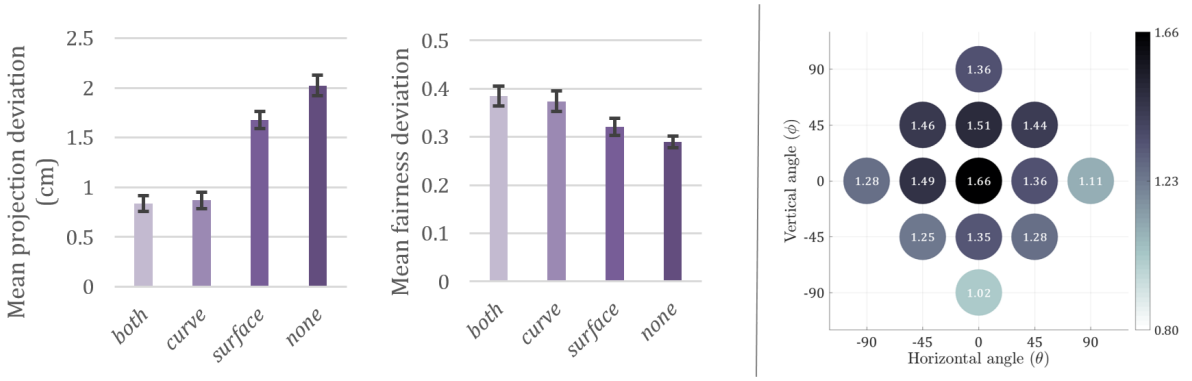


Figure 3.11: Main effects of *visual guidance* on mean projected deviation (left) and mean fairness deviation (centre), and of *surface orientation* on mean depth deviation. The observed trend for *visual guidance* was exactly the opposite for projection and fairness deviations. Depth deviation improved when moving away from the vertical plane in front of the participants.

3.3.5 Procedure

Participants were instructed to stand at a fixed spot for the duration of the experiment, and to avoid moving their lower body as much as possible. The procedure was otherwise the same as the first experiment—participants used a switch in their left hand to engage the pen, and the drawn stroke was rendered in real-time. Each trial consisted of drawing a circle of diameter 30cm as a single stroke. The apparatus was the same as the first experiment, using an HTC Vive and OptiTrack system for display and tracking, respectively.

3.3.6 Data Preparation

The same filtering procedure as Experiment 1 was performed to remove tracking noise from the data. However, strokes were not translated to align their starting points. This allows us to observe how the presence or absence of certain visual guidance affects depth perception and positioning accuracy.

For comparing stroke characteristics for the two values of the *surface shape* variable, the circles drawn on the *curved* (cylindrical) surfaces were transformed to a planar circle by unwrapping the surface [13] such that it becomes the XY plane in the new coordinate system, while the Z direction remains the same. This unwrapping transforms curves drawn in the *curved* condition into an equivalent coordinate system as the curves drawn in the *flat* condition.

3.3.7 Measurements

The main measurement is mean projected deviation, as defined for Experiment 1, which represent the components of the overall deviation in the local XY plane. I also look at depth deviation, which represents the overall deviation in the local Z direction. I do not discuss overall deviation, as its trends were very similar to that of mean projected deviation. In addition to these accuracy levels, I also examine the aesthetic quality of the drawn strokes, and the stroke execution time.

Mean Depth Deviation

The mean depth deviation refers to mean deviation of the stroke from the target in local Z-direction ($\frac{1}{n}\sum_{i=1}^{n} p_i^z$). This measure will help estimate the effectiveness of visual guidance to aid depth perception during the drawing task.

Mean Fairness Deviation

Fairness of a curve is an important measure of its aesthetic quality [74, 155], useful in many design applications. The fairness of a curve is defined using the smoothness of its curvature. We use a simple approximation to compute this. Formally, if the curvature of the input curve at point $\mathbf{p}_j$ is C_j, then we define mean fairness deviation as

$$\mathrm{FD}(\mathcal{P}) = \frac{1}{n-1}\sum_{i+1}^{n-1}\frac{|C_{i=1} - C_i|}{\mathrm{avg}(C_i, C_{i+1})}. \tag{3.3}$$

Intuitively, this measure penalizes frequent curvature changes in the input stroke, which are undesirable in design applications. A perfect circle has a fairness deviation of 0.

Stroke Execution Time

Stroke execution time is the total time spent in executing a stroke. This measure gives an idea of the extent to which strokes were performed as smooth arm movements in contrast to participants correcting themselves mid-stroke.

3.3.8 Results

Mean Projected Deviation

Significant main effects were observed for *surface shape* ($F_{1,11} = 59.3, p < .001$), *visual guidance* ($F_{3,33} = 81.8, p < .001$), as well as for *surface orientation* ($F_{12,132} = 2.74, p < .01$). The *surface orientation* × *surface shape* interaction was also significant ($F_{12,132} = 3.24, p < .001$).

Among the four *visual guidance* conditions, the greatest error was with *none* (M = 2.02cm, SD = 1.05cm), followed by *surface* (M = 1.67cm, SD = 0.72cm), *stroke* (M = 0.87cm, SD = 0.51cm), and *both* (M = 0.84cm, SD = 0.52cm) (Figure 3.11, left). Post-hoc comparisons revealed that all pairs other than (*stroke*, *both*) were significantly different. This suggests that the drawing surface provides useful guidance for participants, and showing the target stroke helps even further. However, showing the drawing surface when the target curve is already visible may not lead to additional accuracy.

For *surface shape*, mean projected deviation was higher for *curved* (M = 1.63cm, SD = 0.97cm), compared to *flat* (M = 1.07cm, SD = 0.71cm). This may be due to additional mental and motor overhead costs associated with drawing a non-planar stroke.

In contrast to the large effect of *drawing plane* observed in Experiment 1, the difference between various *surface orientations* was significant, but smaller. The maximum difference between any pair of surface orientation values was 0.32cm. This may be because the surfaces in this experiment were positioned to have projections of comparable sizes on the participants' eyes. Consistent with the first experiment, accuracy was greatest when the drawing surface was oriented horizontally.

Mean Depth Deviation

The mean depth deviation was also significantly affected by *visual guidance* ($F_{3,33} = 23.8, p < .001$), *surface orientation* ($F_{12,132} = 4.52, p < .001$), and *surface shape* ($F_{1,11} = 8.75, p < .05$). Post-hoc comparisons revealed that *none* was significantly less accurate than the other three *visual guidance* conditions. However, these three *visual guidance* conditions were not significantly different from one another (Figure 3.11, centre). The mean values followed the order: *none* (M = 2.11cm, SD = 1.02cm), *surface* (M = 1.37cm, SD = 1.11cm), *stroke* (M = 1.01cm, SD = 0.58cm), and *both* (M = 0.91cm, SD = 0.46cm).

For *surface shape*, the mean depth deviation for *curved* surfaces (M = 1.27cm, SD = 0.86cm) was actually lower than for *planar* surfaces (M = 1.43cm, SD = 1.04cm). However, there was a significant interaction effect observed between *surface shape* and *visual guidance* ($F_{3,33} = 4.93, p < .01$). *Curved* surfaces exhibited higher depth deviation than *flat* ones when no visual guidance was provided, but lower depth deviation when any additional visual guidance was provided. This shows that with adequate visual feedback, users may be able to draw non-planar curves as accurately as planar ones.

The mean depth deviation trend for *surface orientation* is visualized in Figure 3.11, right. This data shows that sketching in the vertical frontal orientation is the least accurate (M = 1.66cm, SD = 1.35cm), and accuracy gradually increases as the orientation moves away from the frontal plane in all directions. Consistent with the first study, accuracy is the highest for horizontal surfaces (M = 1.02cm, SD = 0.63cm).

Mean Fairness Deviation

Both *visual guidance* ($F_{3,33} = 20.6, p < .001$) and *surface shape* ($F_{1,11} = 22.6, p < .001$) had a significant effect on mean fairness deviation. Drawn curves were significantly fairer when the target curve was invisible when drawing ($M_{none} = 0.29$; $M_{surface} = 0.32$), as compared to when it was visible ($M_{stroke} = 0.37$, $M_{both} = 0.38$). This followed our informal observation, as participants focused on staying close to the target curve when that guide was displayed.

For surface shapes, we observed that circles drawn on *flat* surfaces (M = 0.32, SD = 0.10) were fairer than those on *curved* surfaces (M = 0.36, SD = 0.09). Lastly, *surface orientation* was found to be significant as well ($F_{11,132} = 7.73, p < .001$). In particular, drawing on the frontal plane ($(\varphi, \theta) = (0, 0)$) produced the fairest curves.

Stroke Execution Time

Total time spent executing curves on the two *surface shapes* was significantly different ($F_{1,11} = 34.3, p < .001$). This is expected since the planar circles can be completed with a simple motion, while non-planar strokes on the *curved* surface require more careful manipulation of the arm. The level of *visual guidance* also significantly influenced the curve execution time ($F_{1,11} = 20.2, p < .001$), with participants spending more time with increasing visual guidance ($M_{none} = 3.31$s, $M_{surface} = 4.11$s, $M_{stroke} = 5.58$s, $M_{both} = 6.02$s). All but the last two differences were statistically significant.

While curve tracing was slower than drawing without any guidance, similar to Viviani and Terzuolo's [253] observation for curves on a plane, the magnitude of the difference was much smaller than the tenfold difference they found.

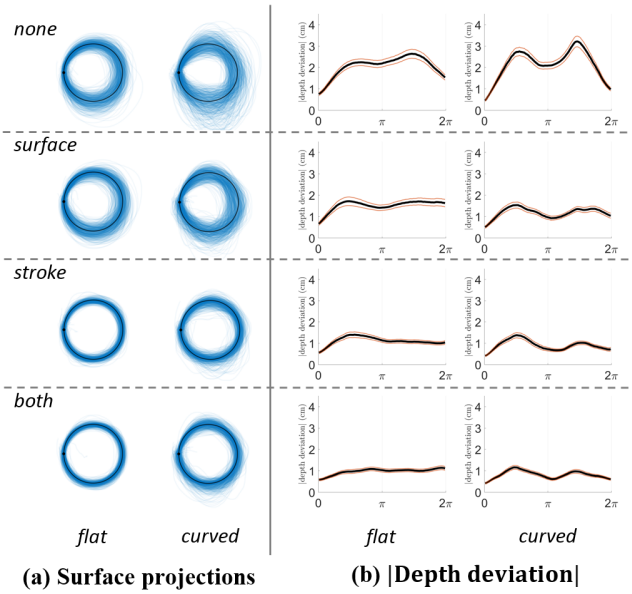


Figure 3.12: Effect of *visual guidance* and *surface shape*. Visualization of surface projections of all strokes (a), and trends of depth deviation with 95% confidence intervals (b). Notice the interaction effect between the two variables on the depth deviation measure.

Stroke execution time correlated strongly with mean fairness deviation ($r = .82, p < .001$), but negatively correlated weakly with mean overall deviation ($r = -.38, p < .001$). Contrary to the observed trend [38] for gestures, this indicates that spending more time on the stroke leads to marginally better accuracy. However, it leads to a high chance of poor curve quality. This is likely due to the difficulty in keeping a stylus stable during slow movements in 3D space.

Visual Trends

Similar to the first experiment, visualization of data reveals interesting trends in the drawing characteristics. I visualize the surface projections of all drawn circles in Figure 3.12a. In Figure 3.12b, I illustrate the associated depth deviations. This visualization reveals an interesting trend with respect to depth deviation: the maximum depth deviation lies around an angle of $\pi/2$ radians from the start, and the minimum occurs around π radians. For *curved* surfaces, another local maximum exists around $3\pi/2$ radians. A similar, albeit more noisy, trend of two characteristic jumps at $\pi/2$ and $3\pi/2$ radians was also seen for curvature. This leads to a hypothesis that participants followed an overly linear path for a quarter of the stroke before making a sudden direction change and continuing on towards linearity again, and repeated this process for the next half as well.

3.4 Discussion

The initial pilot observation clearly points out that while designers appreciate the freeform nature of 3D sketching in VR, precision and control is often required for meaningful design tasks. Designers expressed interest in switching back and forth between freeform and controlled sketching modes.

In my experiments, I observed that the lack of a physical drawing plane, surface position, shape and orientation, stroke size, and visual guidance were important factors affecting drawing ability in VR. While

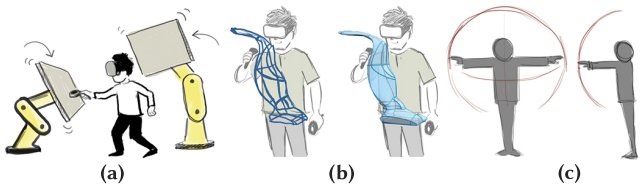


Figure 3.13: Examples of VR interface elements based on our design suggestions: use of an actuated physical support (a), automatic surfacing with semi-transparent rendering (b), and positioning strokes for maximum projection size (c).

a physical drawing surface improves accuracy by 22% as compared to unguided mid-air drawing, just a virtual surface, which can be easily incorporated in many design applications, can improve accuracy by as much as 17%. In applications where the target curve is known a priori, showing that curve can further boost accuracy, but may lead to aesthetically poor strokes. It should be noted, however, that these results characterize strokes drawn by amateurs with no prior VR sketching experience. It is possible that mid-air drawing precision improves with practice and experience. Wiese et al. [257] reported encouraging results on learnability of freehand 3D sketching in a short-term study conducted on design students. Future work on VR sketching learnability can gain from my experience by benchmarking against these results.

While primarily targeted towards sketching applications, my results for conditions with no visual guidance could also be utilized for estimating user precision in performing 3D gestures in VR. Thus, these results could apply to a wide variety of use-cases, such as mode switches, command invocations, or navigation gestures. These use cases result in potential applicability of my results to a number of VR domains such as motor skill training [82], animation and modelling [175, 246], and data visualization [60].

3.4.1 Physical Surface

During the observational sessions, designers recommended projecting their strokes into desired virtual planes for greater accuracy. However, my experimental studies indicate that, compared to accuracy achieved in traditional sketching, projecting to virtual planes in VR is 53% worse, while drawing on a physical surface in VR is only 20% worse. This is a strong motivation for devices combining existing drawing methods in VR. One potential approach is to hold a clipboard, or other rigid surface in the non-dominant hand to rest upon [200]. An alternative approach is dynamically configurable drawing surfaces actuated by robotic arms (Figure 3.13a). While the former potentially allows for higher mobility, the latter may be preferable in long design sessions where fatigue becomes a major factor. Such surfaces could potentially provide a more natural haptic feedback without limiting input range, unlike existing active haptic devices used to aid 3D sketching [130].

3.4.2 Visual Guidance

My experimental results indicate that visual guides, such as grids and scaffolding curves, can help position strokes more accurately, increasing accuracy by 17% and 57%, respectively. Visual guidance may be augmented by snapping techniques, such as the “Guides” feature in Tilt Brush [87], which snaps strokes to nearby surface widgets. However, such guides should be used with caution. In particular,

tracing over an existing curve adversely affects the drawing quality. Rendering a grid may provide the right balance between accuracy and aesthetics. These results are in contrast to assumptions for 2D sketching on which curve smoothing algorithms are based [238]. Therefore, novel stroke processing methods may be needed to account for the lack of curve fairness.

3.4.3 Depth Perception

Accurate depth perception is critical for VR design applications. The *hybrid* condition from the first experiment validated that the difficulties of drawing in VR are not solely due to the lack of physical constraint, but also due to the reduced visual fidelity as compared to the real world. For example, I observed that study participants were not able to match the endpoint of their drawn circles to the starting point. Participants in the initial observation sessions had similar complaints about depth perception, more so when the scene had many curves. Advanced visual guides, beyond the approaches tested in the second experiment (Section 3.3), could be explored. A common method employed in desktop-based modelling applications is fog rendering, to blur distant objects. In an immersive environment, depth may also be effectively conveyed via a 3D grid, which cannot be employed in 2D due to the resulting visual clutter. While drawing curve networks or concept sketches, automatic surfacing with semi-transparent rendering may also be useful for indicating relative depth (see Figure 3.13b).

3.4.4 Orientation, Position, and Navigation

The drawing plane orientation had a significant effect on accuracy in both experiments. In general, the familiar horizontal orientation was most accurate, with vertical orientations performing worse. The effect was most prominent in the first study when participants had to draw on their midsagittal plane (the body's plane of symmetry). In contrast, the drawing surfaces in experiment 2 had approximately the same projection on the user's eyes, and all of them were at a comfortable motor distance. Bae et al. [22] define a "sketchability" measure for 2D sketches, which states that a good viewpoint for sketching 2D strokes is one where the sketch surface has a large visible projection. The trend observed in my experiments hints that the same model applies for sketching in an immersive VR environment as well. Thus, it would be interesting to explore how VR design tools could equip designers with simple navigation tools that enable them to snap or project the drawing planes to positions and orientations with higher "sketchability" (Figure 3.13c).

3.4.5 Drawing Scale

My experiments provide evidence that drawing in large scale leads to a sharp increase in drawing inaccuracies. This can be partly attributed to the tendency of the human arm to follow a natural arc. This might lead one to the conclusion that, contrary to prior recommendations [110, 232], full-scale design in VR is not a good strategy, if precision is desired. Instead, I would suggest that VR design tools allow users to draw in a small manageable scale to improve accuracy and comfort, while allowing users to quickly switch to full-scale view to utilize the visualization benefits afforded by VR [110, 232].

3.4.6 Non-planar Strokes

While a large number of design tasks can be performed using planar strokes only [57, 267], previous work suggests that non-planar strokes occur commonly in scenarios such as automobile design [92] and organic shapes [112]. Since my experiments indicate that drawing on planar surfaces produces more accurate and fair curves, I believe that mid-air design tools could incorporate methods to allow planar curves to be projected onto curved surfaces. However, my results also showed that with adequate visual feedback, users may be able to accurately draw non-planar curves in mid-air. Previous work utilizing 2D displays have used orthographic projections of surfaces onto planes [92], or level set representations of 3D spaces around a surface [205]. How well these techniques relate to artist intent in VR is an interesting avenue to explore in the future. Finally, this work only studied a fundamental set of non-planar strokes—circles projected onto a constant curvature convex surface. While the impact of factors studied here is likely independent of shape complexity, more complex 3D strokes are an interesting topic for future investigations. For instance, it is unknown how surface curvature and torsion affect stroke performance.

3.5 Conclusion

By taking away the flatness of paper, VR sketching tools enable artists and designers to dive in and create in free space without any constraints. While this offers immense freedom and expressiveness, my observations indicate that sketching precise curves for design tasks in VR is challenging. This chapter quantitatively explored multiple factors affecting the imprecision in 3D sketching in VR, including scale, lack of physical surface, orientation, and planarity. I hope that this understanding would lay out the foundation for building tools to equip designers with the desired precision and controls, and I describe some of these tools in the later chapters. Based on my analysis, I also discussed a number of design and interaction guidelines that could potentially alleviate the precision problem for sketching in VR and warrant further investigation. However, further studies are required to understand the effectiveness of these solutions to respect artist intent and maintain fidelity. My hope is that this work would take this medium forward by making it more receptive to the needs of creative design.

An aspect of sketching which has not been touched by the quantitative analysis presented in this chapter is ideation sketching. Sketching is the dominant mode of ideation in various domains of art and design, and it remains to be seen if mid-air sketching can prove to be as effective as the time-tested technique of 2D sketching. As noted in the chapter's initial sections, virtual reality offers numerous advantages for design tasks—limitless scale, 3D visualization, immersion—which positively impact exploratory ideation. However, the ubiquity of sketching in design pipelines can partially also be attributed to the accessibility of 2D sketching, where the 3D domain may be no match to it. Buxton [36] ascribes this ubiquity to the low cost and wide availability of sketch creation, and its support for rapid expression. Virtual reality and 3D sketching cannot currently rival the 2D sketching tradition along these ideation ideals. I will further discuss ideation sketching in Chapter 4 when we discuss a hybrid 2D–3D sketching tool, and again in Chapter 6 where we target ideation and concept sketching in a single system.

Chapter 4

Combining 2D and 3D Sketching

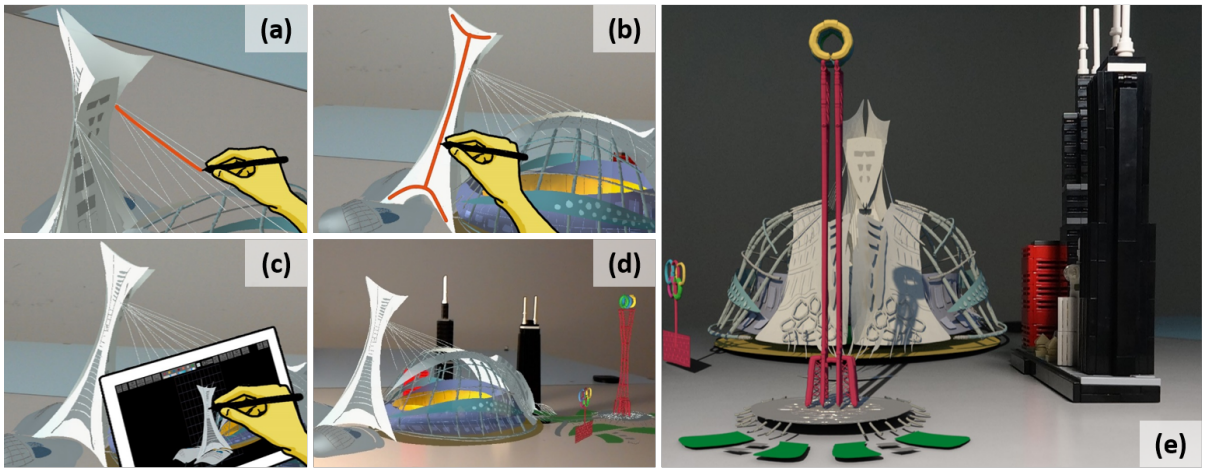


Figure 4.1: SymbiosisSketch combines mid-air 3D interactions (a) with constrained sketching. Users can create planar or curved *canvases* (b), and use a tablet (c) to sketch onto them. Designs are created in situ, in context of physical objects in the scene (d), allowing quick post-processing operations to seamlessly blend the virtual objects into the real world (e).

Sketching has been a design and conceptualization aid for centuries. While traditional 2D sketching is ingrained into our consciousness, abilities, and training from childhood [269], the growth of consumer grade augmented and virtual reality (VR/AR) devices has enabled us to transcend the limits of the sketch canvas digitally—equipping artists with the unprecedented ability to sketch 3D curves directly in the air [58, 131]. However, the previous chapter described challenges in control and precision, perception, and ergonomics faced by the users of 3D sketching systems. The chapter concluded with a list of design guidelines for 3D sketching systems. In the current chapter, I use those guidelines to design a novel 3D sketching tool: SymbiosisSketch.

Despite their novelty, current commercial mid-air 3D drawing systems [71, 87] lack the precision and constraints necessary for use in conceptual design. This is primarily due to our ergonomic inability to draw well and in detail at arbitrary 3D scales, orientations, and without a physical surface to support and steady our sketching motion (see Keefe et al. [130] and Chapter 3). Interestingly, traditional and digital 2D sketching systems address this issue well. They are ergonomically superior, not fatiguing to

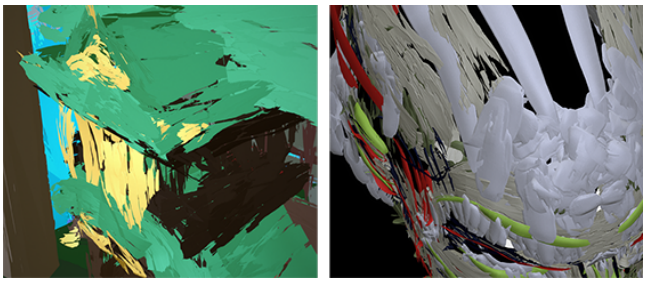


Figure 4.2: Lacking surfaces, Tilt Brush users employ many strokes to depict texture or the illusion of a surface. © Kamikakushi de Sen et Chihiro (left) and Christopher Watson (right). Used under CC-Attribution 3.0.

use, and afford greater control and precision over sketch strokes. Over the centuries, 2D sketching has further developed a vocabulary (see Figure 4.4) of illustration styles, textures, and rendering techniques to depict geometric detail, material, and lighting [66, 194]. While drawing such illustrative detail in 2D is as common and efficient as handwriting, executing regular stroke patterns mid-air in 3D is extremely challenging (see Figure 4.2).

The biggest problem with 2D sketching is conveying the absolute 3D depth from a single viewpoint using only a flat surface. Often multiple 2D sketches from disparate viewpoints, using construction lines [267], perspective grids [23] and scaffolds [209], are required in 2D to convey all visual aspects of a 3D design. In contrast, depth is trivially integral to mid-air 3D sketching.

Another common practice in 2D design sketching is the use of photographs or renderings of 3D objects as a reference to situate sketches in a 3D context [57]. Augmented Reality (AR) has the potential of incorporating a 3D context directly into the design process. However, AR exacerbates the problem of visual clutter when complex designs are viewed as a collection of 3D curves. Creating surfaces from 3D curves is typically treated as a separate and subsequent stage in conceptual design [32]. Integrating surfacing with curve sketching can provide occlusion primitives to reduce visual clutter from both the real environment and sketched 3D curves, as well as a local 3D surface to map 2D sketch strokes.

I distill these synergistic insights and the design guidelines developed in Chapter 3 into SymbiosisSketch, a system that judiciously combines 3D mid-air and 2D surface sketching to produce conceptual models comprising 3D curves and surfaces within an AR setting. The system equips the designer with a HoloLens, a spatially tracked 3D stylus, and a digital tablet to view and sketch both mid-air in 3D and on a 2D surface (see Figure 4.6). The user specifies design surfaces by sketching multiple strokes. Working in AR allows designers to work directly with existing 3D objects in situ. Finally, this system allows users to rescale work volumes, allowing drawings that range from miniature to larger than life in size.

The principal contribution is a novel hybrid sketching workflow in AR with tools and widgets that leverage the complementary strengths of 3D mid-air and 2D tablet sketching. Surface creation is integrated early in the drawing workflow, as *drawing canvases* for 2D sketching, and better visualization of the evolving design.

I performed a user study with both professional designers and experienced amateurs to evaluate the system. The evaluation and results confirm the hypothesized advantages of this hybrid workflow for 3D conceptual design.

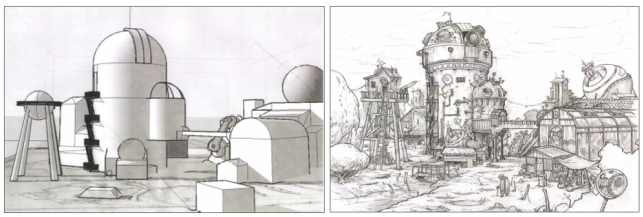


Figure 4.3: Design workflow using 3D and 2D tools. Rustam Hasanov modelled an underlay of an interior scene in a 3D modelling tool (SketchUp), and then drew details over the top of those underlays using a 2D interface and tools. © Rustam Hasanov. Used with permission.

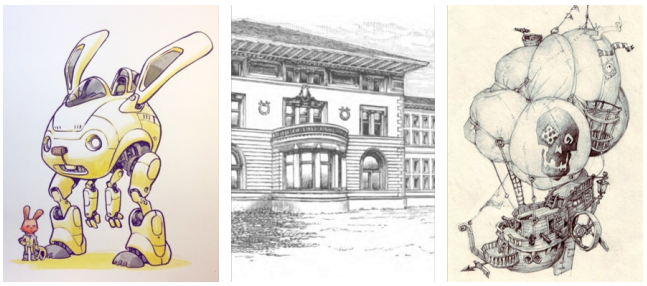


Figure 4.4: Details in 2D drawings: structural details, architectural embellishments, and textures. © Jake Parker (left) and Johannes Figlhuber (right). Used with permission.

4.1 Motivation: Hybrid Interfaces for Design

SymbiosisSketch is motivated by ideas from multiple and complementary disciplines—the wealth of 2D drawing practice, modern VR-based design tools, in situ 3D design research tools, and the affordances of modern AR hardware.

4.1.1 Portraying 3D shapes using 2D

3D Tools for Layout and Perspective

While 2D sketching is a ubiquitous medium for drawing and conceptualization, depicting 3D shapes in 2D requires expertise and experience. In particular, understanding, controlling, and portraying depth in 2D is difficult [210]. Experienced designers often employ the complementary strengths of 2D and 3D design tools in their workflows for detailed drawings [219]. For instance, 3D modelling programs can aid in the basic layout of perspective drawing grids, quickly blocking out proportion and composition, from a desired viewpoint, before exploring design details in 2D (see Figure 4.3).

2D for Surface-constrained Strokes, Texturing, and Details

As evident in numerous books on drawing and design [66, 194], 2D sketching incorporates a myriad of artistic styles and textures, that are used to convey structure, material properties, function, lighting, and even motion (see Figure 4.4). While commonplace in 2D, authoring such detail using freeform 3D curves without a constraining physical surface is tedious, tiring, and error-prone, further motivating the use of a physical tablet.

Figure 4.5: Post-processed results. HoloLens hardware limitations restrict us to basic Gouraud shading. With appropriate shading and occlusion, SymbiosisSketch designs can seamlessly blend into the real world. (Clockwise from top) war helicopter shooting at Captain America (Figure 4.14a), mechanical wing augmentation (Figure 4.14c), mini car (participant creation), Flintstones' house (author creation), and large fan (Figure 4.14e).

4.1.2 Designing in Situ

A central idea of this work is the use of AR to visualize and interact with a physical 3D stylus, a tablet, and objects in context with the utmost fidelity. Designing in context of the physical world also requires working with scales spanning from the miniature to the massive. To this end, the tool should allow designers to interact minimally with real world shapes and contours, minimize visual clutter, draw with a comfortable posture, and still visualize their creations 1:1 in the real world.

4.2 System Overview and Setup

The system requires three main hardware components: an AR-capable Head-Mounted Display (HMD), a tablet for 2D drawing, and a digital pen with 6-degree of freedom (DoF) motion tracking to draw mid-air, as well as on the tablet. The pen position and orientation is tracked using Vicon motion capture cameras [252]. The three devices communicate in real-time to ensure a unified, consistent, state of the world and the design elements. I used the Microsoft HoloLens 1 [160] as the AR HMD, whose spatial mapping capability provides a coarse 3D map of the physical world as a triangle mesh. The tablet for 2D interface is a Microsoft Surface Book [161] clipboard, detached from its keyboard.

Metaphorically, this setup is analogous to a painter's bimanual interaction with her tools. Typically, a painter reserves her dominant hand for holding the paintbrush for making marks on the canvas, and

the non-dominant hand to hold the colour palette. In my multimodal interaction system, the user holds the pen in the dominant hand, while the tablet is typically held in the non-dominant hand (but can also rest on a table like an easel). The tablet serves two purposes. First, it renders an orthographic view of the world, and users can draw on the tablet to project their strokes to a 2D *canvas* visible from this orthographic window (Figure 4.6a). Second, it also includes a function toolbar, accessible with pen or touch input (Figure 4.8). For a given *drawing canvas* planar projected on the tablet screen, the position of the orthographic camera remains fixed for comfortable sketching, regardless of the position and orientation of the tablet. Hence, we do need to not explicitly track the position and orientation of the tablet.

In my prototype, the buttons on a standard mouse are used to trigger these interactions. The mouse is magnetically fastened to the back of the tablet, allowing the user to interact with the mouse while holding the tablet (see Figure 4.6b). Alternatively, the user can also hold the mouse by itself in the non-dominant hand, while the tablet rests on a table.

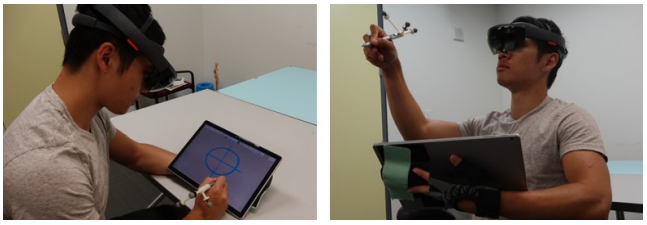


Figure 4.6: Setup: the user puts on the HoloLens and draws with a motion-tracked stylus, on a tablet (left), or mid-air (right) using a mouse affixed to the back of the tablet.

4.3 Components

I first introduce some terminology, and the basic interface and interaction components of our 3D design tool.

4.3.1 Strokes

3D strokes are commonly stored and rendered either as *ribbons* (flat narrow sheets), or as *tubes* (generalized cylinders). I use tube rendering for both 2D and 3D strokes to provide a homogeneous sketch appearance, and to distinguish curves from surface patches (*drawing canvases*).

4.3.2 Drawing Canvases

Most 3D objects are better represented using a set of surface patches, than just a collection of curves. In programs like Tilt Brush [87], artists tediously align many wide ribbon-shaped strokes next to each other to create an approximate impression of a surface (see Figure 4.2). Most 3D sketching tools do not address surface creation at all, and some define transient surfaces on which to draw curves and subsequently delete the surface [23]. In contrast, SymbiosisSketch's output is a collection of 3D curves and surface patches. In addition to defining the 3D conceptual design, the surfaces map to the physical tablet as *drawing canvases*. The design of the *drawing canvases* is motivated by similar concepts utilized in existing creative works. For example, the unique art style of the Academy award winning short

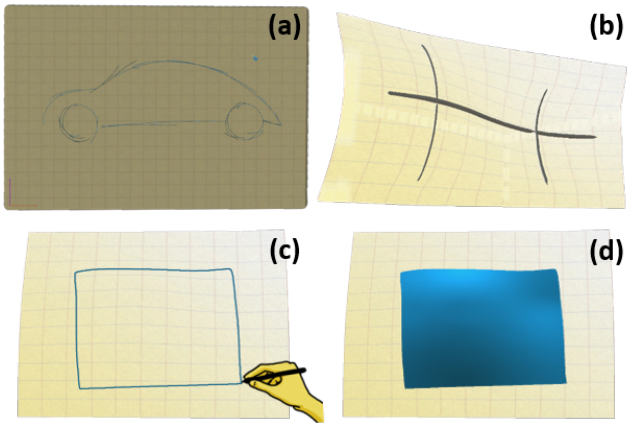


Figure 4.7: Strokes drawn using the 2D tablet are projected onto *drawing canvases*. (a) A *planar drawing canvas* is a rectangle with the same aspect ratio as the tablet. (b) Surfaces form *curved drawing canvases*. (c) Users can draw closed curves on *canvases* to define (d) *solid surfaces* which also lend occlusion, lighting, and shadows to the design.

film Paperman [256] was accentuated by sketching onto modelled surfaces. A similar concept was also employed by Bassett et al. [29] for authoring detailed painterly characters. Technically, a *drawing canvas* in my system is either a planar rectangle, or a surface patch represented as a height field over a planar rectangle.

4.3.3 Solid Surfaces

A *solid surface* is an enclosed region projected to a *drawing canvas*, with a uniform material appearance. Essentially, this is a triangulated mesh that aids in 3D model understanding by providing shading, occlusion, and shadows [49, 102]. In the absence of *solid surfaces*, depth perception can be poor and complex objects appear to be "stroke spaghetti" instead of coherent 3D objects, even with binocular cues (Section 3.4.3).

4.4 User Interface & Interactions

SymbiosisSketch is designed to enable a symbiotic relationship between 2D and 3D sketching, utilizing the strengths of both, while minimizing their respective shortcomings. Mid-air strokes and UI manipulations are executed by moving the pen with the dominant hand, while holding the mouse buttons with non-dominant hand. The 2D tablet renders the 3D world. Any stroke drawn in 2D is mirrored in the 3D world in real-time, and vice-versa. The 2D interface (see Figure 4.11) shows an orthographic projection of the active *drawing canvas*, a colour palette, and other functionality and configuration settings (also see Figure 4.8). Figure 4.1 demonstrates the overall workflow of the system.

4.4.1 Sketching in Freeform 3D

For freeform 3D strokes, the user presses and holds the left mouse button to initiate mid-air sketching. The stroke drawn traces the motion of the pen's tip. A virtual pen-tip is overlaid to aid positional perception.

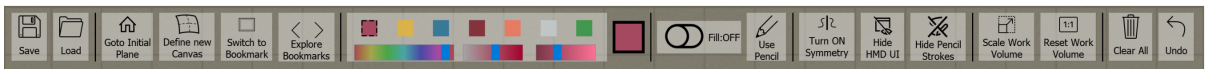


Figure 4.8: 2D UI toolbar. (left to right) save/load, initial *canvas*, create *canvas*, explore and select bookmarks, colour tools (palette, HSV sliders, indicator), fill, pencil/ink toggle, symmetry plane, UI and pencil toggles, workspace scaling and reset, clear all and undo stroke. Buttons are shown only if currently useful. For example, selecting the pencil hides colour tools. Icons © icons8.com, CC BY-ND 3.0.

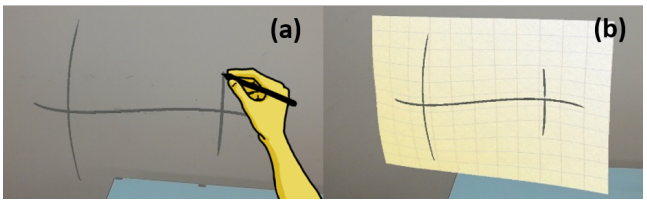


Figure 4.9: Creating a *curved drawing canvas*. (a) The user draws a few strokes using the 2D and/or 3D interface. (b) A surface patch is fit to these strokes.

4.4.2 Drawing Canvases

Creating and manipulating *drawing canvases* (both planar and curved) is a defining aspect of the system that allows users to sketch with greater precision. Users can define and use an arbitrary number of *drawing canvases* during their designs, but only one *canvas* is active at a time. The 2D tablet's viewport always projects an orthographic view of the active *drawing canvas*, allowing users to draw on the tablet and project their strokes onto that *canvas* in 3D.

Planar Drawing Canvas

Initially, the system displays a *planar drawing canvas* at a fixed location in 3D. The designer can thus sketch planar strokes without foreshortening on the *drawing canvas* by sketching on the tablet screen. The aspect ratio of the *planar drawing canvas* is constrained to match the tablet screen (Figures 4.6a, 4.7a). At any point in time, the designer can activate a *planar drawing canvas*, and position it in 3D space.

Curved Drawing Canvas

To create a *curved drawing canvas*, the user presses the "Define new canvas" button in the tablet toolbar. The user then draws a few canvas-defining 3D curves, coarsely specifying the intended surface in 3D space (Figure 4.9). Once completed, the user presses "Generate Canvas" to generate a curved surface that best fits the canvas-defining 3D strokes. The tablet's orthographic camera is positioned in front of the surface, ensuring the maximal projection area on the tablet plane, to maximize *sketchability* (see Figures 4.7b, 4.11a, Section 3.4.4, and Bae et al. [22]). It is worth noting that these canvas-defining strokes need not always be drawn mid-air. A user can freely combine canvas-constrained strokes (drawn using 2D tablet) with freehand mid-air 3D strokes to define an intended *drawing canvas*. This is very useful for creating *canvases* for depicting embossed or engraved details, which can be drawn efficiently over existing *canvases* (see an example in Figure 4.14a).

Technically, the *drawing canvas* is an open surface with a continuous boundary topologically equivalent to a disc. This class of surface patches are most common in 3D surface modelling, and further allows

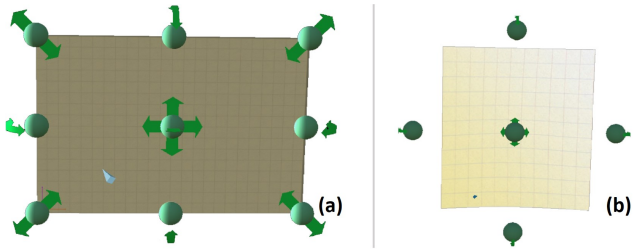


Figure 4.10: Widgets for direct 3D manipulation: translation and rotation (all *canvases*), and scaling (*planar canvases*): shown at the centre, edges, and corners, respectively.

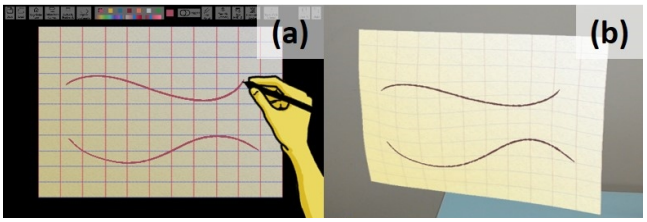


Figure 4.11: (a) The user sketches in the 2D view of the *canvas*. (b) The corresponding strokes in 3D.

the user to project strokes drawn on a tablet onto the entire *canvas* without frequent view manipulation.

By default, the *canvas* is clipped against the bounding rectangle of its canvas-defining strokes on the best-fit plane, giving it the appearance of a rectangle curved in 3D. However, if the projection of the largest enclosing canvas-defining stroke is approximately a closed simple curve, I further trim the *canvas* against this stroke. While the surface behaves just like a rectangular *canvas* for drawing, I hypothesized that the trimmed surface is visually more representative as part of the 3D design process.

Canvas Manipulation

The *canvas* manipulation tools enable designers to freely transform an active *drawing canvas* in 3D space (Figure 4.10). To move a *drawing canvas*, the user positions the pen's tip inside the translation widget, then presses and holds the right mouse button with non-dominant hand, and then freely moves it in 3D space. Similarly, widgets are provided along the *canvas* edges to scale (*planar canvas* only) and rotate the *canvas* about its local origin (see Figure 4.10). Since accurately rotating in 3D is difficult, I snap these rotations to multiples of 15°.

Canvas Bookmarking

The system automatically bookmarks all *drawing canvases* utilized during the design process. The toolbar UI allows switching to previously utilized *canvases* for design iterations.

4.4.3 Sketching on the 2D Tablet

My tools can project a broad class of curved surfaces or arbitrary planes in 3D to the tablet's 2D screen. The tablet displays an orthographic view of the active *drawing canvas* (see Figure 4.11a). Drawing on the tablet projects the input points onto the 3D *canvas* to create a new stroke. The ability to draw on an intended surface using the 2D physical surface discounts the depth error of mid-air 3D strokes (Section 3.2.7), which is a major source of drawing inaccuracy.

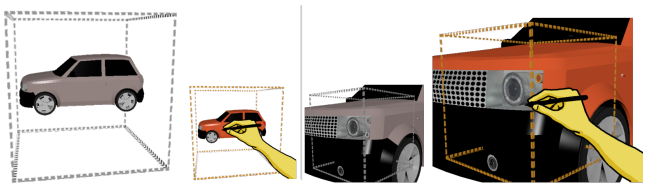


Figure 4.12: Workspace scaling tool for drawing a car. The user uses a large workspace to draw the shape of the car in a small, comfortable, scale (left). She then defines a small workspace centred on a headlight (right). This "zoom in" effect allows her to capture the details of the headlight's shape, define a new *canvas* on it, and draw highlights.

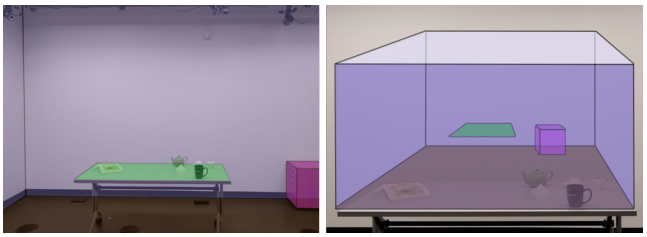


Figure 4.13: Physical planes detected by the HoloLens are automatically bookmarked (shown overlaid on the scene, left). With workspace scaling, users can visualize and work even with room-scale planes in an accessible position and comfortable scale (right).

When the pen is in proximity of the tablet's screen, a virtual pointer is displayed at the projected position in 3D. This indicator gives artists the ability to quickly judge the expected 3D position of their stroke when drawing in 2D. A grid texture on the *canvas* further helps users draw precisely. When the pen leaves the tablet's vicinity, the virtual pointer snaps back to the pen's tip.

Solid Surface Tool

The user can toggle ON the "Fill" button on the toolbar to activate the solid surface mode. In this mode, when the user draws a stroke on the tablet, the system automatically closes that curve (by connecting the first and last points), triangulates the enclosed region, and projects the shape onto the active drawing canvas with a selected solid colour (see Figure 4.7c–d). The solid surface tool is only accessible by the 2D tablet interface, since executing surface-constrained curves mid-air is challenging and error-prone.

4.4.4 Workspace Scaling

VR/AR environments allow designers to draw and visualize at an arbitrary scale. However, drawing at extreme scales—even if humanly possible—is detrimental to stroke quality (Section 3.2.7). However, when drawing in situ, the user often needs to design at such uncomfortable scales.

In order to mitigate these issues, we provide a workspace translation-cum-scaling tool. By default, the user works in a 1:1 scale, which is called the canonical workspace. On activating the tool using the tablet toolbar, the user sees two rays—one emanating from her head and another from the pen. Pressing the right mouse button creates a 3D point $\mathbf{q}_1$ at the approximate intersection point of the two rays. This is one corner of the intended workspace. Performing this action again defines the diagonally opposite corner ($\mathbf{q}_2$). This point selection method is inspired by Cheng and Tataksuk's virtual pointer [44].

This physical workspace is then mapped to the canonical volume, which provides a virtual view of

the world (orange dashed boxes in Figure 4.12). The user can now draw comfortably with respect to the canonical workspace, with results mapped to the defined workspace (grey dashed boxes in Figure 4.12).

4.4.5 Interaction with Physical Objects

Being an AR-based tool, SymbiosisSketch directly allows users to interact and draw relative to 3D objects in the physical world (see Figure 4.14). Drawing directly over many physical objects however, can be awkward, due to their steep curvature (see Figure 4.15c), softness (drapery or clothes, see Figure 4.14d), or size and reachability (buildings, ceilings, or cars, Figure 4.14e). In such cases, defining *curved canvases* using physical objects as proxies, or using workspace scaling, can greatly reduce the direct interaction required with the objects, improving final design outcomes.

The above capability is further bolstered by the HoloLens's capability to map the physical world as a 3D triangle mesh, which SymbiosisSketch utilizes to capture physical planes from the real world. Ideally, we would like to automatically use this mesh to generate *canvases* over all interesting physical objects. Unfortunately, the resolution of HoloLens's spatial mapping mesh is not fine enough to capture small objects or curved surface detail. Therefore, only the physical planes provided by the HoloLens Spatial Mapping API [162] are utilized. These physical planes are automatically bookmarked, and particularly useful for precise in situ architectural and interior design. The user can directly access physical landmarks such as the floor, ceiling, tables, and walls, and bring them into the canonical workspace for further sketching (see Figure 4.13).

4.4.6 Helper Tools

Pencil Tool

The pencil tool is used to draw strokes of a pre-defined thickness and colour. It works just like the regular stroke tool, but users have a switch to hide all pencil strokes. In conjugation with *canvas* bookmarking, it serves as a simple alternative to layers. For example, a user may use the pencil to draw out scaffolds and other construction lines, and finally hide the pencil strokes in the finished design.

Symmetry Plane

The symmetry plane mirrors the users' strokes about itself. Translation and rotation widgets on the symmetry plane are used for positioning and orienting it.

Additional Sketching Tools

SymbiosisSketch also provides a few essential sketching tools to help users utilize the system efficiently. Users can manipulate the stroke width using the mouse wheel. Stroke colour can be modified from a pre-defined palette, or using HSV sliders. Since the additive display of the HoloLens cannot render true black, the value slider is clipped to disallow very dark colours. An undo tool is also provided. Finally, the resulting designs can be exported as an `.OBJ` file for post-processing.

4.5 Implementation Details

The system is implemented in two parts: the tablet app is written in C++ using Qt and OpenGL, and runs on a Surface Book with a 2.60GHz Intel i7 CPU and Intel 520 GPU and support for pen and touch inputs. The second application runs on the HoloLens, and is written in C# using the Unity engine. While the HoloLens is capable of tracking objects, the latency and precision are not good enough for our task. The system is therefore augmented with a Vicon motion capture system connected to a computer to track the drawing pen in the air. The tablet app communicates with the motion capture server at 100Hz, and all information exchange between the tablet and HMD is carried out over a reliable low-latency TCP connection. I now provide technical details for implementing various pieces of the system.

4.5.1 Stroke Smoothing and Reparametrization

The input strokes are smoothed using a real-time smoothing procedure inspired by Elasticurves [238]. Let $\{\mathbf{s}_i\}$ be a sequence of input points, and $\{\mathbf{p}_i\}$ be the sequence of points forming the stroke after real-time smoothing. Then,

$$\mathbf{p}_i = \begin{cases} \mathbf{s}_i & \text{if } i \in \{0, n\}, \\ \mathbf{p}_{i-1} + f(\mathbf{s}_i - \mathbf{p}_{i-1}) & \text{otherwise.} \end{cases} \tag{4.1}$$

Here, n is the number of points in the sequence, and $f \in [0, 1]$ is the smoothing factor. The stroke starts and ends exactly at locations specified by the user, with smoothing in the middle modulated by f and the drawing speed. I use $f = 0.8$ for tablet strokes, and $f = 0.1$ for mid-air strokes (for less and more smoothing, respectively). This is followed by reparametrizing according to arc-length to make the strokes more suitable for *canvas* fitting and rendering. Finally, 10 iterations of bi-Laplacian smoothing remove additional jitter from the mid-air 3D strokes.

4.5.2 Rendering Curves and Solid Surfaces

For rendering curves as 3D tubes, we need to compute an orthogonal 3D frame at each segment of the stroke, where one of the three orthogonal directions is the tangent to the curve at that point. I start with an arbitrary frame at the first segment, and use parallel transport (Hanson and Ma [94]) to compute the Bishop frame [33] for the rest of the curve. The thickness of mid-air curves is given simply by the current stroke width setting set by the user. For tablet strokes, this value is multiplied by the normalized pressure value at each point, lending a typical *sketchy* look to the strokes.

When the "Fill" tool is active, trimmed solid surfaces are rendered by computing a Constrained Delaunay triangulation (CDT) with all the segments of the input stroke (including an one joining the end-points) as constraints. I compute the CDT using the standard technique (Cheng et al. [45]): first compute the CDT on a convex region covering the stroke, and then remove the triangles outside the region enclosed by the stroke. The vertices of the triangulation are then projected onto the active *canvas*, similar to the boundary curve points.

Figure 4.14: Sample results showing SymbiosisSketch's creative potential. Author creations: war helicopter shooting at a Captain America figure (a), logo drawn on a person's clothing (b), mechanical wing augmenting a robot figure (c, f), skirt drawn over a human model (d), and a large wall-fan (e, h). Participant creations: scary figure utilizing a mannequin head (g), basketball hoop design (i), and genie emanating out of a teapot (j) with details (k, l).

4.5.3 Curved Canvas Fitting and Drawing

The surface fitting function proceeds in two stages: finding the best-fit plane for the set of strokes used to create the surface, and then fitting a smooth height field to the strokes, relative to the best-fit plane. For the best-fit plane, I first uniformly sample a fixed number of points from each input stroke. I then solve for the best-fit plane in a least-squares sense [64]. This plane's local +Y and -Z axes then trivially define the orthographic view's up and forward directions.

To define the *curved canvas*, I then fit a height field (from the best-fit plane) to the input strokes by minimizing the field's thin plate spline energy. Since the input 3D curves can have inaccuracies, I use the approximate fitting method proposed by Donato and Belongie [61]. Their method utilizes a regularization parameter λ, where $\lambda = 0$ means that the solver tries to fit to the input points exactly, while a higher value implies sacrificing exact fit for smoothness.

Since we expect curves input via the 2D interface to be very precise, λ is set to zero for those, while for the curves drawn mid-air, $\lambda = 0.1$. Finally, the viewport of the tablet's camera is scaled such that the whole *canvas* is visible from it.

4.5.4 Workspace Translation and Scaling

For selecting 3D points for defining the workspace, approximate intersection between the pen and head rays is defined as the mid-point of the shortest line segment between the two rays. Once the workspace is defined, it maps an input position $\mathbf{p}_c$ in the canonical volume to the new volume as

$$\mathbf{p} = s\mathbf{p}_c + \mathbf{t}, \tag{4.2}$$

where $s = \|\mathbf{q}_1 - \mathbf{q}_2\|$, $\mathbf{t} = (\mathbf{q}_1 + \mathbf{q}_2)/2$.

4.6 User Evaluation

I evaluated the system with professional as well as amateur illustrators and designers. The goals of the study were to observe user workflow, test the system's core functionality, and to understand its limitations. A 3D mid-air drawing condition was also included to compare freeform 3D sketching alone with SymbiosisSketch.

4.6.1 Participants

Six participants (age 27–46, 2 female and 4 male) were recruited for the study. Participants were experienced with graphic design (*P1, P5*), 3D modelling (*P2, P6*), and VR-based design tools (*P1–2, P5–6*). Participants were compensated for their time.

4.6.2 Procedure

The study was conducted at the DGP lab. Each evaluation session was 90–120 minutes long, and consisted of four phases.

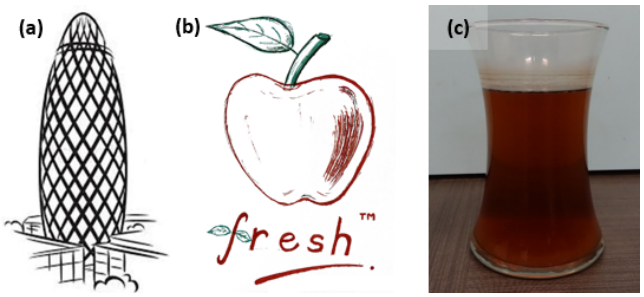


Figure 4.15: Task-1: drawing London's Gherkin building (a). Task-2: drawing a logo (b) on a curved physical surface of a vase (c).

Overview and training (10–20 minutes). Participants were introduced to the scope of the project and given time to get used to the HoloLens. Then, an experimenter explained the four core interactions: 3D drawing, 2D drawing, *canvas* creation, *canvas* manipulation, and other helper tools. Participants practised using the above tools until they felt confident. The workspace scaling features were not used in the study.

Fixed tasks (30–45 minutes). Two fixed tasks were designed to compare SymbiosisSketch to existing 3D sketching systems. The order of the systems was counterbalanced among the participants to eliminate learning effects.

The first task involved drawing the outline of a bullet shaped building with textured features (Figure 4.15a). Participants had to perform the task using two systems: *symbiosis*, and *mid-air only*, which did not afford *canvas* creation and stroke projection, therefore replicating existing mid-air drawing systems such as Tilt Brush [87]. For the former, participants were encouraged to utilize the *drawing canvas* tool.

Task-2 involved drawing a logo over a curved physical surface (see Figures 4.15b-c). Participants were encouraged to rest the tip of their pen on the physical surface. The motivation behind the second task was to measure the usefulness of the 2D interface even when the user can get better motor control (as compared to mid-air sketching) by resting on a physical surface.

Expanded overview & exploration (5–10 minutes). After completing the fixed tasks, participants were instructed about solid surfaces, workspace scaling, and the *canvas* bookmarking feature.

Freeform exploration and design (30–45 minutes). Finally, participants were free to explore the system on their own to create designs in the fourth phase. They were provided a host of physical props to inspire their designs. They completed a questionnaire after the study.

4.6.3 Results for Fixed Tasks

For the fixed tasks, the designs were evaluated using two quantitative measures: *task completion time* and *total strokes drawn*, defined simply as the number of strokes in the final design. The latter provides a proxy measure for the level of detail present in the designs.

For each participant, the *task completion time*, as well as the *total strokes drawn*, were higher when using the *symbiosis* system. In task-1, mean task completion time in the symbiosis condition (M=519s,

SD=115s) was over two times that of *mid-air only* (M=242s, SD=122s). A similar trend was observed for the second task: (M=281s, SD=123s) vs (M=148s, SD=85s).

The additional time taken with the combined 2D/3D interface of *symbiosis* was accompanied by additional strokes drawn. Users using *symbiosis* drew 75% more strokes (M=64, SD=24) than *mid-air only* (M=37, SD=14), for the first task. This difference increased to 140% for the texture detail–heavy second task: (M=67, SD=38) vs (M=28, SD=20). Paired t-tests reveal that the two conditions were statistically significant ($p < .05$) on the task completion time measure for both tasks, and the total strokes drawn measure for task-2. However, these statistics should be considered with caution since the focus of the study was qualitative measures, and the tasks were therefore not carefully controlled.

Figure 4.16: All participants' drawings for the fixed tasks. Row-wise from top: task 1 (Gherkin building) in *symbiosis* condition, the in *mid-air only* condition; task 2 (logo on a physical surface) in *symbiosis* condition, and in *mid-air only* condition.

My observations during the study suggest that these results can be explained by two main factors. First, participants took some time getting used to the *canvas* creation tool, which required more mental effort than mid-air sketching. However, once participants understood how to use the tool, they were able

to utilize it effectively, and drew detailed designs more accurately when they had access to the 2D UI. Figure 4.16 qualitatively demonstrates that the *symbiosis* condition produced drawings that were more precise, and better reproduced details shown in the target images.

4.6.4 Freeform Design and Qualitative Feedback

Participants were successful in utilizing SymbiosisSketch for a variety of design tasks. During the brief design sessions, participants attempted a design sketch of a car (*P2*), environment design involving physical props (*P4, P6*), character augmentations (*P3, P4*), and mystical art (*P5*). Figure 4.14 shows some of the resulting artifacts.

Overall, the qualitative feedback collected indicated that participants found SymbiosisSketch's toolset useful. This is evident in *P5*'s comment that the system is "a great way to bridge old methods with new technology" , but "[composing] a 3D image will require a lot of work and planning" . Below, I report the feedback received for individual components.

Drawing on Tablet to Create 3D Sketches

Participants found this feature to be extremely useful. On a 5-point Likert scale (1=not useful, 5=extremely useful), the median score was 4.5 for usefulness and 4.0 for usability. They appreciated that the hybrid interface helped replicate the degree of control in traditional 2D sketches to the 3D sketching scenario. Mapping a drawing surface to physical objects in the world was judged to be a useful and enjoyable feature.

"Drawing in 3D can feel unhinged and out of control, and this capability made me feel some of the enjoyment and control I feel when I practice traditional drawing." *(P3)*

Ability to Combine 2D and 3D Sketching in One Design

Participants responded positively to the multimodal interface for 3D design. They found the hybrid interface very useful (rated 4.5 out of 5), as well as easy to use (rated 4 out of 5).

"I really liked using the planes, and the combination of 2D & 3D. I really like the consideration of how working solely in a 3D space can become hard for most people." *(P5)*

An exception was *P2*, who, used to ZBrush [187] sculpting, finds freeform 3D sketching/modelling challenging.

Curved Canvas Creation and Stroke Projection

Users unanimously found the ability to create *drawing canvases* very useful (rated 5 out of 5).

"Mapping of the canvas in 3D was what I liked the most. Once I had defined a filled, curved region that I was happy with I found that it was very enjoyable to draw curves starting from that surface." *(P4)*

However, some participants thought the feature was difficult to use (median usability 3) due to control issues. In the future, we would like to add finer curvature control to the *curved canvas* tool to address these problems.

Use of Physical Props and Surfaces

Use of physical props was assessed to be both useful (median 4.5) and easy to use (median 4). In fact, all participants utilized this feature, and included physical objects in their creations. They also traced these objects to define *canvases*.

"Mapping a drawing surface to physical objects in the world was an immensely useful and enjoyable feature." *(P1)*

Miscellaneous Feedback

Participants provided feedback on other aspects of the system as well. In particular, they found *solid surfaces* to be useful and aesthetically pleasing. Participants (*P1, P3, P4*) also wanted non-planar closed curves drawn mid-air to be usable to trim a surface. This is an interesting technical challenge for future work: a possible implementation could project the curve onto the best-fit plane and then lift a 2D triangulation to 3D. Finally, *P2* suggested an additional interaction mode: drawing on the tablet and generating a swept surface by moving it in 3D. One can imagine this to be similar to, but more versatile than, Surface Drawing [204]. Controlling the tablet's motion and orientation mid-air, however, may be challenging.

4.7 Discussion

The evaluation suggested that participants enjoyed the core ideas of SymbiosisSketch, and were able to utilize the system successfully for creating the designs they desired. The artifacts created by the study participants and those created by the authors of the associated publication [17], confirm that SymbiosisSketch effectively mixes the traditional with the new to create a compelling and novel hybrid interface for 3D concept design. Along with the encouraging results, the user evaluation also showed some limitations, with opportunities to improve in the future.

4.7.1 Limitations and Future work

Drawing Canvases

Participants (*P1, P4*) expressed the desire to go beyond current surface patch topology, to create surfaces with multiple boundaries like open cylinders, or closed surfaces like spheres. Fitting a general surface poses new problems: inferring surface topology, defining a non-trivial mapping to a *drawing canvas*, and the need for view manipulation in the tablet to draw on such a *canvas*.

In Situ Design in the Wild and Other Hardware Limitations

P3 expressed the desire to using the system to plan and create large murals. Unfortunately, the reliance on a motion capture system to track the 3D stylus makes the current setup impractical outside a lab. The small field of view on the HoloLens also made it hard to view large strokes in totality (*P6*). Better 3D acquisition of the physical environment in AR can allow complex surfaces to become *drawing canvases*, but in turn poses the problem of scene surface segmentation.

Stroke & Canvas Manipulation for Iterative Design

While the Elasticurve [238] smoothing effectively removes tracking noise and motor control error from 3D strokes, it cannot remove *errors of intent*—caused by conflicting visual percepts or users changing their mind later. Non-rigid manipulation of *drawing canvases* and correction of freeform 3D curves by tablet-based oversketching are great avenues for future exploration. Effective manipulation and editing tools can improve the value of this system for the iterative design process.

Hybrid 2D–3D Interfaces for Functional Design Ideation

While SymbiosisSketch is built upon the idea that the *shape* of the designed object and its various parts dictate how 2D and 3D sketching are combined, a functionality-based distribution of design tasks between these two sketching paradigms can be an interesting area to explore. For instance, since immersion aids the designer's comprehension of spatial relationships [114], immersive mid-air sketching can be utilized to mark out prominent spatial features, whereas the design of aesthetic aspects may be largely delegated to 2D sketching. Such a symbiosis can be especially impactful for ideation where the aim is to use the minimal effort to externalize the core ideas [36]. The hybrid SymbiosisSketch interface and quick surface design in VR can also inspire future work in other domains where 3D shape information and computational support is crucial, such as the ideation of automobile lighting [67].

4.8 Conclusion

SymbiosisSketch is a hybrid system aiming to blend design knowledge and expertise from the traditional 2D sketching domain to the new and exciting world of direct 3D sketching in VR/AR. By devising a novel method for defining constrained *drawing canvases* in 3D and building tools to support interaction with physical objects, I built a system for 3D conceptual design in situ. The user evaluation confirmed that this toolset is useful, effective, and is able to support a variety of design tasks for users with diverse artistic backgrounds. I hope that this work guides future research in visual communication, furthering our ability to efficiently transform mental design concepts into digital models.

Chapter 5

Drawing on Meshes in Virtual Reality

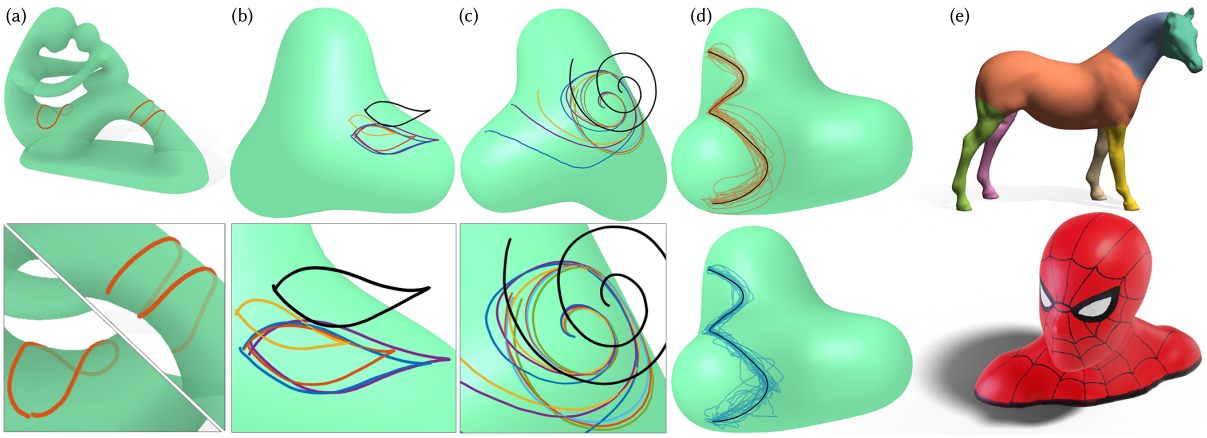


Figure 5.1: Drawing curves mid-air that lie precisely on the surface of a virtual 3D object in VR/AR is difficult (a). Projecting mid-air 3D strokes (**black**) onto 3D objects is an under-constrained problem with many seemingly reasonable solutions (b). We analyze this fundamental VR/AR problem of 3D stroke projection, define and characterize multiple novel projection techniques (c), and test the two most promising approaches—*spraycan* shown in **blue** and *mimicry* shown in **red** in (b)–(d)—using a quantitative study with 20 users (d). The user-preferred *mimicry* technique attempts to mimic the 3D mid-air stroke as closely as possible when projecting onto the virtual object. We showcase the importance of drawing curves on 3D surfaces, and the utility of our novel *mimicry* approach, using multiple artistic and functional applications (e) such as interactive shape segmentation (top) and texture painting (bottom).

In the previous chapters, we considered mid-air 3D sketching and sketching on physical 2D surfaces. We also utilized a physical surface for drawing onto virtual surface representable as height fields. Let's now consider the general problem of drawing curves onto virtual surfaces in immersive spaces.

Drawing or painting on 3D virtual objects is a critical task for interactive 3D modelling, animation, and visualization, where its uses include object selection, annotation, and segmentation [96, 116, 159]; 3D curve and surface design [107, 165]; strokes for 3D model texturing or painterly rendering [118] (Figure 5.1e). In 2D, digitally drawn on-screen strokes are mapped onto 3D virtual objects in a WYSIWYG

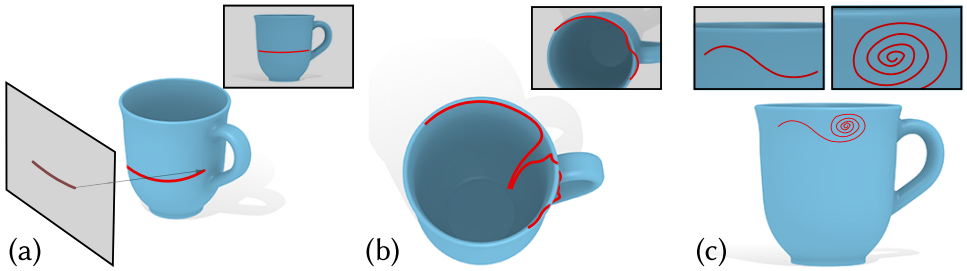


Figure 5.2: Stroke projection using a 2D interface is typically WYSIWYG: 2D points along a user stroke (a, inset) are ray-cast through the given view to create corresponding 3D curve points on the surface of 3D scene objects (a). Even small errors or noise in 2D strokes can cause large discontinuities in 3D, especially near ridges and sharp features (b). Complex curves spanning many viewpoints, or with large scale variations in detail, often require the curve to be drawn in segments from multiple user-adjusted viewpoints (c).

(what you see is what you get) manner, by projecting 2D stroke points through the given view onto the virtual object(s) (Figure 5.2a). Such WYSIWYG projection through an orthographic or perspective viewport was used in SymbiosisSketch (Chapter 4) and is also common in commercial desktop tools such as Substance Painter [5]. Unfortunately, when either the curve or the target object is reasonably complex, multiple view changes may be required and achieving the desirable results can be difficult (Figure 5.2b,c).

Given the numerous applications of drawing curves onto surfaces, a natural question to ask is whether we can use immersion and 3D interactions to improve the experience compared to 2D interfaces. In the previous chapters, we have seen that drawing directly on a virtual 3D object is nearly impossible without haptic constraints (Figure 5.3). Furthermore, unlike 2D drawing, where the WYSIWYG view-based projection of 2D strokes onto 3D objects is unambiguously clear, the user-intended mapping of a mid-air 3D stroke onto a 3D object is less obvious. In this chapter, I present the first detailed investigation into plausible user-intended projections of mid-air strokes on to 3D virtual objects.

Existing desktop interfaces that don't perform WYSIWYG projection either use perceptual insights or geometric assumptions like smoothness and planarity to project, neaten, or otherwise process sketched strokes. Some applications wait for user stroke completion before processing it in entirety, for example when fitting splines [22]. Our goal is to establish an application agnostic, base-line projection approach for mid-air 3D strokes. We will therefore assume a stroke is processed while being drawn and inked in real-time. That is, the output curve corresponding to a partially drawn stroke is fixed/inked in real-time, based on partial stroke input [238].

One might further conjecture that all "reasonable" and mostly continuous projections would produce similar results, as long as users are given interactive visual feedback of the projection. This is indeed true for tasks requiring discrete point-on-surface selection, where users can freely re-position the drawing tool until its interactively visible projection corresponds to user intent. Real-time curve drawing, however, is very sensitive to the projection technique, where any mismatch between user intention and algorithmic projection, is continuously inked into the projected curve (Figure 5.1d).

2D Strokes Projected onto 3D Objects. The standard user-intended mapping of a 2D on-screen stroke is a raycast projection through the given monocular viewpoint. Raycasting is WYSIWYG—the 3D curve visually matches the 2D stroke from said viewpoint (Figure 5.2a). Ongoing research on mapping

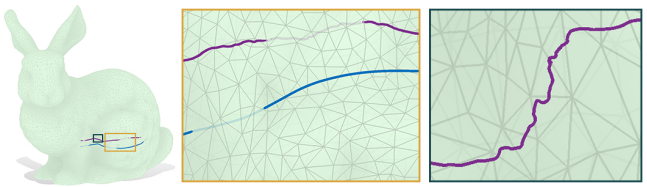


Figure 5.3: Mid-air drawing precisely on a 3D virtual object is difficult (faint regions of strokes are behind the surface), regardless of drawing quick smooth strokes (**blue**), or slow detailed strokes (**purple**). Deliberately slow drawing is further detrimental to stroke aesthetic (right).

2D strokes to 3D objects assumes this fundamental *view-centric* projection, focusing instead on specific problems such as creating curves around ridge/valley features where small 2D error can cause large 3D depth error (Figure 5.2b). Others focus on drawing complex curves with large scale variation where multiple viewpoint changes are needed while drawing, (Figure 5.2c). These problems are mitigated by the direct 3D input and viewing flexibility of VR/AR, assuming the mid-air stroke to 3D object projection matches user intent.

3D Strokes Projected onto 3D Objects. Physical analogies motivate existing approaches to defining a user-intended projection from 3D points in a mid-air stroke to 3D points on a virtual object (Figure 5.4). Graffiti-style painting with a *spraycan* is arguably the current standard, deployed in commercial immersive paint and sculpt software such as Medium [4] and Gravity Sketch [91]. A closest-point projection approximates drawing with the tool on the 3D object, without actual physical contact (used by the "guides" tool in Tilt Brush [87]). Like view-centric 2D stroke projection, these approaches are *context-free*: processing each mid-air point independently. The VR/AR drawing environment comprising six–degree of freedom controller input and unconstrained binocular viewing, is however, significantly richer than 2D sketching. The user-intended projection of a mid-air stroke (Section 5.1) as a result is complex, influenced by the ever-changing 3D relationship between the view, drawing controller and virtual object. This chapter therefore argue the need for historical context (i.e., the partially drawn stroke and its projection) in determining the projection of a given stroke point. The use of this historical context is balanced with the overarching goal of a general purpose projection that makes little or no assumption on the nature of the user stroke or its projection.

In the chapter, I will describe several *anchored* projection techniques—that minimally use the most recently projected stroke point—as context for projecting the current stroke point (Section 5.2). I evaluated various anchored projections, both theoretically and practically by pilot testing. The most promising and novel approach *anchored-smooth-closest-point*, also called *mimicry*, captures the observed tendency of a user stroke to mimic the shape of the desired projected curve. A formal user study in VR (Section 5.3) shows *mimicry* to perform significantly better than *spraycan* (the current baseline) in producing curves that match user intent (Section 5.4). While the formal evaluation is limited to VR, the fundamental problem could directly translate to AR scenarios as well. This chapter thus contributes, to the best of my knowledge, the first principled investigation of real-time inked techniques to project 3D mid-air strokes drawn in VR onto 3D virtual objects, and a novel stroke projection benchmark for VR—*mimicry*.

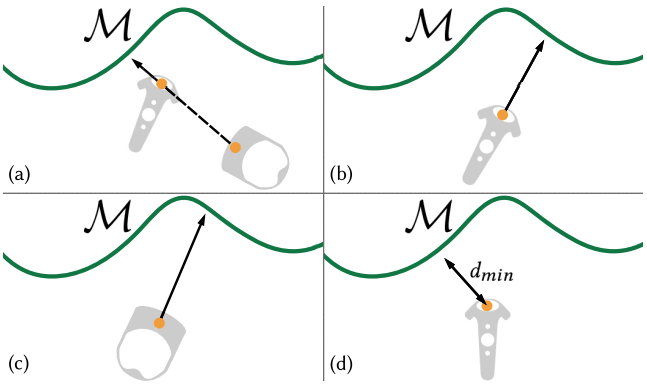


Figure 5.4: Context-free techniques: *occlude* projects points from the controller origin along the direction from the eye (HMD origin) to the controller (a); *spraycan* projects points from the controller origin in a direction defined by the controller's orientation (b); *head-centric*, akin to 2D projects points along the view direction defined by HMD orientation (c); *snap* projects points from the controller origin to their closest-point on $\mathcal{M}$ (d).

5.1 Projecting Strokes onto 3D Objects

I will first formally state the problem of projecting a mid-air 3D stroke onto a 3D virtual object. Let $\mathcal{M} = (V, E, F)$ be a 3D object, represented as a manifold triangle mesh embedded in $\mathbb{R}^3$. A user draws a piecewise linear mid-air stroke by moving a 6-DoF controller or drawing tool in VR. The 3D stroke $\mathcal{P} \subset \mathbb{R}^3$ is a sequence of n points $(\mathbf{p}_i)_{i=0}^{n-1}$, connected by line segments. Corresponding to each point $\mathbf{p}_i \in \mathbb{R}^3$, is a system state $S_i = (\mathbf{h}_i, \mathbf{c}_i, \hbar_i, c_i)$, where $\mathbf{h}_i, \mathbf{c}_i \in \mathbb{R}^3$ are the positions of the headset and the controller, respectively, and $\hbar_i, c_i \in \mathrm{Sp}(1)$ are their respective orientations, represented as unit quaternions. Also, without loss of generality, assume $\mathbf{c}_i = \mathbf{p}_i$. That is, the controller positions describe the stroke points $\mathbf{p}_i$.

We want to define a *projection* π, which transforms the sequence of points $(\mathbf{p}_i)_{i=0}^{n-1}$ to a corresponding sequence of points $(\mathbf{q}_i)_{i=0}^{n-1}$ on the 3D virtual object, i.e. $\mathbf{q}_i \in \mathcal{M}$. Consecutive points in this sequence are connected by geodesics on $\mathcal{M}$, describing the *projected curve* $\mathcal{Q} \subset \mathcal{M}$. The aim of a successful projection method is to match the undisclosed user-intended curve. The projection is also designed for real-time inking of curves: points $\mathbf{p}_i$ are processed upon input and projected in real-time (under 100ms) to $\mathbf{q}_i$ using the current system state S_i, and optionally, prior system states $(S_j)_{j=0}^{i-1}$, stroke points $(\mathbf{p}_j)_{j=0}^{i-1}$, and projections $(\mathbf{q}_j)_{j=0}^{i-1}$.

5.1.1 Context-Free Projection Techniques

Context-free techniques project points independent of each other, simply based on the spatial relationships between the controller, HMD, and the target object (Figure 5.4). These techniques can be further categorized as raycast or proximity based.

Raycast Projections

View-centric projection in 2D interfaces projects points from the screen along a ray from the eye through the screen point, to where the ray first intersects the 3D object. In an immersive setting, raycast approaches similarly use a ray emanating from the 3D stroke point to intersect 3D objects. This ray

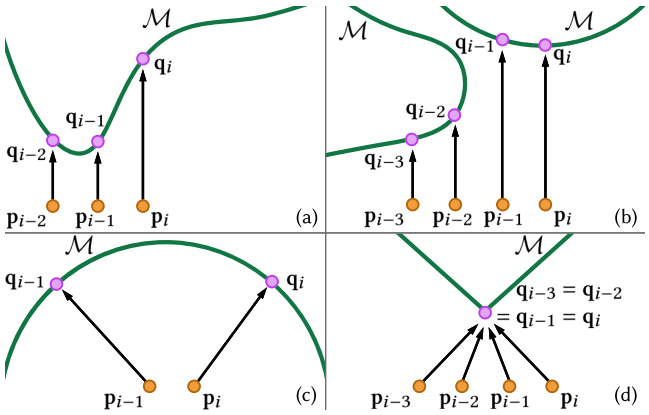


Figure 5.5: Context-free projection problems: large depth disparity (a), unexpected jumps (b), projection discontinuities (c), and undesirable snapping (d).

$(\mathbf{o}, \mathbf{d})$ with origin $\mathbf{o}$ and direction $\mathbf{d}$ can be defined in a number of ways. Similar to pointing behaviour, *occlude* defines this ray from the eye through the controller origin: $(\mathbf{c}_i, (\mathbf{c}_i - \mathbf{h}_i)/\|\mathbf{c}_i - \mathbf{h}_i\|)$, illustrated in Figure 5.4a. If the ray intersects $\mathcal{M}$, then the closest intersection to $\mathbf{p}_i$ defines $\mathbf{q}_i$. In case of no intersection, $\mathbf{p}_i$ is ignored in defining the projected curve, i.e., $\mathbf{q}_i$ is marked undefined and the projected curve connects $\mathbf{q}_{i-1}$ to $\mathbf{q}_{i+1}$ (or the proximal indices on either side of i for which projections are defined). The *spraycan* approach treats the controller like a spraycan, defining the ray like a nozzle direction in the local space of the controller (Figure 5.4b). For example, the ray could be defined as $(\mathbf{c}_i, \mathbf{f}_i)$, where the nozzle $\mathbf{f}_i = \mathbf{c}_i \cdot [0, 0, 1]^T$ is the controller's local z-axis (or *forward* direction). Alternately, *head-centric* projection can define the ray using the HMD's view direction as $(\mathbf{h}_i, \mathbf{h}_i \cdot [0, 0, 1]^T)$ (Figure 5.4c).

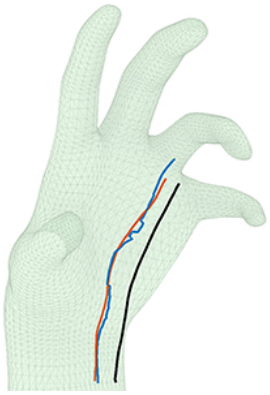
Pros and Cons. The strengths of raycasting are: a predictable visual/proprioceptive sense of ray direction; a spatially continuous mapping between user input and projection rays; and scenarios where it is difficult or undesirable to reach and draw close to the virtual object. Its biggest limitation stems from the controller/HMD-based ray direction being completely agnostic of the shape or location of the 3D object. Projected curves can consequently be very different in shape and size from drawn strokes (Figure 5.5a–b), and ill-defined for stroke points with no ray-object intersection.

Proximity-Based Projections

In 2D interfaces, the on-screen 2D strokes are typically distant to the viewed 3D scene, necessitating some form of raycast projection onto the visible surface of 3D objects. In contrast, VR/AR users are able to reach out in 3D and directly draw the desired curve on the 3D object. While precise mid-air drawing on a virtual surface is very difficult in practice (Figure 5.3), projection methods based on proximity between the mid-air stroke and the 3D object are certainly worth investigation.

The simplest proximity-based projection technique, *snap*, projects a stroke point $\mathbf{p}_i$ to its closest-point in $\mathcal{M}$ (Figure 5.4d).

$$\mathbf{q}_i = \pi_{snap}(\mathbf{p}_i) = \underset{\mathbf{x} \in \mathcal{M}}{\arg\min}\ d(\mathbf{p}_i, \mathbf{x}), \tag{5.1}$$



where $d(\cdot,\cdot)$ is the Euclidean distance between two points. Unfortunately, for triangle meshes, closest-point projection tends to snap to mesh edges (**blue** curve in the Figure on the right), resulting in unexpectedly jaggy projected curves, even for smooth 3D input strokes (**black** curve) [179]. These discontinuities are due to the discrete nature of the mesh representation, as well as spatial singularities in closest point computation even for smooth 3D objects. We can mitigate this problem by formulating an extension of Panozzo et al.'s *Phong* projection [179] in Section 5.1.2, that simulates projection onto an imaginary smooth surface approximated by the mesh. I will denote this *smooth-closest-point* projection as π_{SCP} (**red** curve).

Pros and Cons. The biggest strength of proximity-based projection is that it exploits the immersive concept of drawing directly on or near an object, using the spatial relationship between a 3D stroke point and the 3D object to determine projection. The main limitation is that since users rarely draw precisely on the surface, discontinuities in concave regions (Figure 5.5c) and undesirable snapping in highly-convex regions (Figure 5.5d) persist when projecting distantly drawn stoke points, even when using *smooth-closest-point*. In Section 5.2.1, I address this problem using stroke *mimicry* to anchor distant stroke points close to the object to be finally projected using *smooth-closest-point*.

5.1.2 Smooth-Closest-Point Projection

The goal of *smooth-closest-point* projection is to define a mapping from a 3D point to a point on $\mathcal{M}$ that approximates the closest point projection but tends to be functionally smooth, at least for points near the 3D object.

Phong projection, introduced by Panozzo et al. [179], addresses these goals for points expressible as weighted-averages of points on $\mathcal{M}$. I extend their technique to define a smooth-closest-point projection for points in the neighbourhood of the mesh. For completeness, I first present a brief overview of their technique.

Phong projection is a two-step approach to map a point $\mathbf{y}^3 \in \mathbb{R}^3$ to a manifold triangle mesh $\mathcal{M}$ embedded in $\mathbb{R}^3$, emulating closest-point projection on a smooth surface approximated by the triangle mesh. First, $\mathcal{M}$ is embedded in a higher dimensional Euclidean space $\mathbb{R}^d$ such that Euclidean distance (between points on the mesh) in $\mathbb{R}^d$ approximates geodesic distances in $\mathbb{R}^3$. Second, analogous to vertex normal interpolation in Phong shading, a smooth surface is approximated by blending tangent planes across edges. Barycentric coordinates at a point within a triangle are used to blend the tangent planes corresponding to the three edges incident to the triangle. I extend the first step to a higher dimensional embedding of not just the triangle mesh $\mathcal{M}$, but a tetrahedral mesh of an offset volume, or *shell*, around the mesh $\mathcal{M}$ (Figure 5.6). The second step remains the same, and I refer the reader to Panozzo et al. [179] for details.

For clarity, I refer to $\mathcal{M}$ embedded in $\mathbb{R}^3$ as $\mathcal{M}^3$, and the embedding in $\mathbb{R}^d$ as $\mathcal{M}^d$. Panozzo et al. compute $\mathcal{M}^d$ by first embedding a subset of the vertices in $\mathbb{R}^D$ using metric multi-dimensional scaling (MDS) [55], aiming to preserve the geodesic distance between the vertices. The embedding of the remaining vertices is then computed using LS-meshes [223].

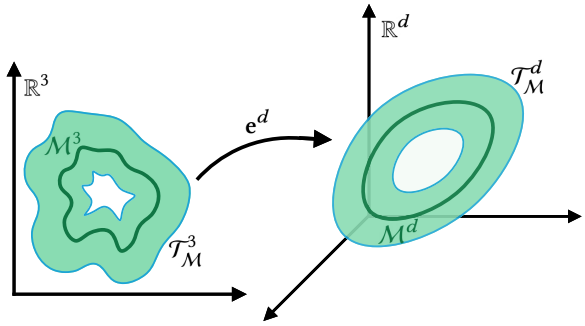
For the problem of computing weighted averages on surfaces, one only needs to project 3D points of the form $\mathbf{y}^3 = \sum w_i \mathbf{x}_i^3$, where all $\mathbf{x}_i^3 \in \mathcal{M}^3$. The point $\mathbf{y}^3$ is *lifted* into $\mathbb{R}^d$ by simply defining $\mathbf{y}^d = \sum w_i \mathbf{x}_i^d$,

$$\begin{array}{ccccc} \{\mathbf{x}^d\} \subset \mathcal{M}^d & \xrightarrow{\text{Def.}} & \mathbf{y}^d = \sum w_i \mathbf{x}_i^d & \xrightarrow{\mathcal{P}_{Phong}} & \mathbf{z}^d \in \mathcal{M}^d \\ \Big\uparrow \text{Bary}(\mathcal{M}),\ \mathbf{e}^d(\mathcal{M}) & & & & \Big\downarrow \text{Bary}(\mathcal{M}) \\ \{\mathbf{x}^d\} \subset \mathcal{M}^3 & \xrightarrow{\text{Def.}} & \mathbf{y}^3 = \sum w_i \mathbf{x}_i^3 & & \mathbf{z}^3 \in \mathcal{M}^3 \end{array}$$

(a) Computing weighted averages in Panozzo et al. [179].

$$\begin{array}{ccc} \mathbf{y}^d \in \mathcal{T}_{\mathcal{M}}^d & \xrightarrow{\mathcal{P}_{Phong}} & \mathbf{z}^d \in \mathcal{M}^d \\ \Big\uparrow \text{Bary}(\mathcal{T}_{\mathcal{M}}),\ \mathbf{e}^d(\mathcal{T}_{\mathcal{M}}) & & \Big\downarrow \text{Bary}(\mathcal{M}) \\ \mathbf{y}^3 \in \mathcal{T}_{\mathcal{M}}^3 & & \mathbf{z}^3 \in \mathcal{M}^3 \end{array}$$

(b) Computing smooth-closest-point projection.



(c) Computing a d-dimensional embedding for $\mathcal{M}$ and $\mathcal{T}_{\mathcal{M}}$.

Figure 5.6: Panozzo et al. [179] compute weighted averages on surfaces (a), while we want to compute a smooth closest-point projection for an arbitrary point *near* the mesh in $\mathbb{R}^3$ (b). I therefore embed $\mathcal{T}_{\mathcal{M}}$—the region *around* the mesh—in higher-dimensional space $\mathbb{R}^d$, instead of just $\mathcal{M}$ (c).

where $\mathbf{x}_i^d$ is defined as the point on $\mathcal{M}^d$ with the same implicit coordinates (triangle and barycentric coordinates) as $\mathbf{x}_i^3$ does on $\mathcal{M}^3$. Therefore, their approach only embeds $\mathcal{M}$ into $\mathbb{R}^d$ (Figure 5.6a,c). In contrast, we want to project arbitrary points *near* $\mathcal{M}^3$ onto it using the Phong projection. Therefore, I compute the offset surfaces at signed-distance $\pm\mu$ from $\mathcal{M}$. I then compute a tetrahedral mesh $\mathcal{T}_{\mathcal{M}}^3$ of the space between these two surfaces in $\mathbb{R}^3$. In the final precomputation step, I embed the vertices of $\mathcal{T}_{\mathcal{M}}$ in $\mathbb{R}^d$ using MDS and LS-Meshes as described above.

Now, given a 3D point $\mathbf{y}^3$ within a distance μ from $\mathcal{M}^3$, we can situate it within $\mathcal{T}_{\mathcal{M}}^3$, use tetrahedral Barycentric coordinates to infer its location in $\mathbb{R}^d$, and then compute its Phong projection (Figure 5.6b,c). We can fallback to closest-point projection for points outside $\mathcal{T}_{\mathcal{M}}^3$, since Phong projection converges to closest-point projection when far from $\mathcal{M}$. However, I set μ large enough to easily handle the smooth-closest-point queries in Section 5.2.1.

Projection Quality and Robustness Tests

Since the desirable properties of the Phong projection are not theoretically guaranteed for shapes with sharp features and noisy meshes [179], I experimentally measured the quality of the embedding by testing for extreme dihedral angles—below 5° or above 175°—resulting in sliver tets in the $\mathbb{R}^d$-embedding (Table 5.1). Further, for a direct measure of projection quality, I densely sampled points in $\mathcal{T}_{\mathcal{M}}$ (four points per tet) and projected each using both π_{snap} as well as π_{SCP}. Typically, we expect π_{SCP} to be a smoother version of π_{snap} (Section 5.1.1). Therefore, a π_{SCP} projection much farther from the input

Table 5.1: Embedding quality and π_{SCP} failure results. The former is indicated by the percentage of dihedral angles below 5° or above 175° in the $\mathbb{R}^d$-embedding, and the latter is defined in Equation 5.2 (lower values are desirable). Also shown are mesh sizes: (#vertices, #faces) for $\mathcal{M}$ and (#vertices, #tets) for $\mathcal{T}_\mathcal{M}$.

Shape	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{T}_\mathcal{M}\rvert$	**% slivers**	**% π_{SCP} fail**
Trebol	(1.2K, 2.3K)	(7.9K, 38K)	6.89	0.00
Cube	(1.5K, 3.0K)	(5.7K, 26K)	17.44	< 0.01
Torus	(1.7K, 3.5K)	(11K, 58K)	0.41	0.00
Spiderman	(3.3K, 6.6K)	(15K, 83K)	2.01	0.02
Hand	(4.2K, 8.5K)	(19K, 114K)	0.49	0.39
Fertility	(4.5K, 9.0K)	(21K, 124K)	0.61	0.29
Fandisk	(6.5K, 13K)	(23K, 133K)	4.75	0.96
Bunny	(7.1K, 14K)	(32K, 183K)	4.17	0.08
Horse	(7.7K, 15K)	(34K, 198K)	0.41	0.35
La Madeleine	(20K, 40K)	(68K, 421K)	0.83	0.24
Beast	(30K, 61K)	(132K, 837K)	0.57	< 0.01
Armadillo	(50K, 100K)	(229K, 1.4M)	0.63	< 0.01
Noisy-cube 5%	(1.5K, 3.0K)	(29K, 139K)	35.78	< 0.01
Noisy-cube 10%	(1.5K, 3.0K)	(30K, 149K)	30.07	0.05
Noisy-cube 15%	(1.5K, 3.0K)	(33K, 164K)	21.08	0.93
Noisy-cube 20%	(1.5K, 3.0K)	(33K, 167K)	10.95	3.73

than π_{snap} indicates a clear failure:

$$\|\mathbf{p} - \pi_{SCP}(\mathbf{p})\| > \|\mathbf{p} - \pi_{snap}(\mathbf{p})\| + \|\texttt{BBox}(\mathcal{M}^3)\|/20, \tag{5.2}$$

where $\|\texttt{BBox}(\mathcal{M}^3)\|$ is the length of the bounding box diagonal of $\mathcal{M}^3$. Table 5.1 shows that the projection works well for almost all the sampled points. I also practically tested all the shapes by drawing myriad curves on each, but did not notice any clear failures of π_{SCP}. Finally, the technique was stress-tested using noisy versions of the unit cube mesh. At extreme levels of noise, when each vertex is moved in the normal direction by up to 20% of the cube size, some clear failures showed up (Figure 5.17d). In practice, such failures can be detected heuristically and π_{snap} can be a drop-in replacement for such points. However, such a fix was not implemented the user study presented in Section 5.3, or for any of the results shown in this chapter.

5.1.3 Analysis of Context-Free Projection

The four different context-free projection approaches described in Figure 5.4 were informally tested by 4 users by drawing a variety of curves on the various 3D models shown throughout the chapter. Here are the main observations from the pilot sessions.

— *Head-centric* and *occlude* projections become unpredictable if the user is inadvertently changing their viewpoint while drawing. These projections are also only effective when drawing frontally on an object, like with a 2D interface. Neither exploits the potential gains of mid-air drawing in VR/AR.

— *Spraycan* projection was clearly the most effective context-free technique. It was observed, however, that consciously reorienting the controller while drawing on or around complex objects was both cognitively and physically tiring.

— *Snap* projection was quite sensitive to changes in the distance of the stroke from the object surface, and in general produced the most undulating projections due to closest-point singularities.

— All projections converge to the mid-air user stroke when it precisely conforms to the surface of the 3D object. But as the distance between the object and points on the mid-air stroke increases, their behaviour diverges quickly.

— While users did draw in the vicinity and mostly above the object surface, they rarely drew precisely on the object. While quantitative measurements were not made for the pilots, the average distance of stroke points from the target object was observed to be 4.8 cm in a subsequent formal study (Section 5.3).

The most valuable insight from the pilot sessions was that the user stroke in mid-air often tended to **mimic** the expected projected curve. Context-free approaches, by design, are unable to capture this mimicry, i.e., the notion that the change between projected point as we draw a stroke is commensurate with the change in the 3D points along the stroke. The inability of context-free approaches to capture a notion of stroke mimicry—due to a lack of curve history or context—materializes as problems in different forms.

Projection Discontinuities

Proximal projection (including *smooth-closest-point*) can be highly discontinuous with increasing distance from the 3D object, particularly in concave regions (Figure 5.5a). Mid-air drawing along valleys without staying in precise contact with virtual object is thus extremely difficult. Raycast projections similarly suffer large discontinuous jumps across occluded regions (in the ray direction) of the object (Figure 5.5d).

While this problem theoretically exists in 2D interfaces as well, it is less observed in practice for two reasons. First, 2D drawing on a constraining physical surface is significantly more precise than mid-air drawing in VR/AR (Section 3.2.7). Second, artists minimize such discontinuities by carefully choosing appropriate views (raycast directions) before drawing each curve. Automatic direction control of view or controller, while effective in 2D [172]), is detrimental to a sense of agency and presence in VR/AR.

Undesirable Snapping

Proximity-based methods also tend to get stuck on sharp (or high curvature) convex features of the object (Figure 5.5b). While this can be useful to trace along a ridge feature, it is problematic for general curve-on-surface drawing.

Projection depth disparity

The relative orientation between the 3D object surface and raycast direction can cause large depth disparities between parts of user strokes and curves projected by raycasting (Figure 5.5c). Such irregular bunching or spreading of points on the projected curve also goes against the observation of stroke mimicry. Users can arguably reduce this disparity by continually orienting the view/controller to keep the projection ray well aligned with object surface normal. However, continuous re-orientation can be tiring, ergonomically awkward, and deviates from 2D experience, where pen/brush tilt only impacts curve aesthetic, and not shape.

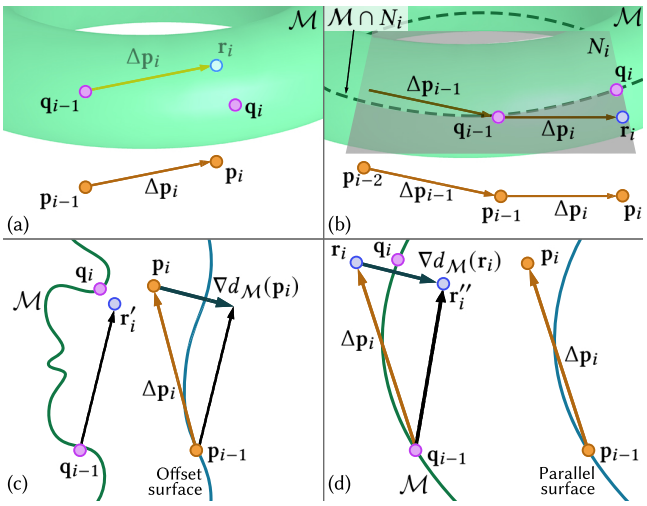


Figure 5.7: Anchored smooth-closest-point (a), and refinements: using a locally-fit plane (b), and anchor point constrained to an offset (c) or parallel surface (d). $\mathbf{q}_i$, is obtained by projecting $\mathbf{r}_i$ (a), $\mathbf{r}_i'$ (c), or $\mathbf{r}_i''$ (d) onto $\mathcal{M}$ via smooth-closest-point; or closest-point to $\mathbf{r}_i$ in $\mathcal{M} \cap N_i$ (b).

These observation motivated the design of projection methods that explicitly incorporate the shape of the mid-air stroke $\mathcal{P}$ and the projected curve $\mathcal{Q}$. I call these functions *anchored*.

5.2 Anchored Stroke Projections

The limitations of context-free projection can be addressed by equipping stroke point projection with the context/history of recently drawn points and their projections. In the techniques proposed here, only minimal context provided by the most recent stroke point $\mathbf{p}_{i-1}$ and its projection $\mathbf{q}_{i-1}$ are used to **anchor** the current projection.

Any reasonable context-free projection can be used for the first stroke point $\mathbf{p}_0$. I use *spraycan* π_{spray}, the preferred context-free technique. For subsequent points ($i > 0$), I compute

$$\mathbf{r}_i = \mathbf{q}_{i-1} + \Delta\mathbf{p}_i, \tag{5.3}$$

where $\Delta\mathbf{p}_i = (\mathbf{p}_i - \mathbf{p}_{i-1})$. I then compute $\mathbf{q}_i$ as a projection of the anchored stroke point $\mathbf{r}_i$ onto $\mathcal{M}$, that attempts to capture $\Delta\mathbf{p}_i \approx \Delta\mathbf{q}_i$. Anchored projections are motivated by the pilot observation that the mid-air user stroke tends to **mimic** the shape of their intended curve on surface. While users to do not adhere consciously to any precise geometric formulation of mimicry, it was observed that users often draw the intended projected curve as a corresponding stroke on an imagined offset or translated surface (Figure 5.7). A good general projection for the anchored point $\mathbf{r}_i$ to $\mathcal{M}$ thus needs to be continuous, predictable, and loosely capture this notion of mimicry.

5.2.1 Mimicry Projection

Controller sampling rate in current VR systems is 50Hz or more, meaning that even during ballistic movements, the distance $\|\Delta\mathbf{p}_i\|$ for any stroke sample i is of the order of a few millimetres. Consequently, the anchored stroke point $\mathbf{r}_i$ is typically much closer to $\mathcal{M}$, than the stroke point $\mathbf{p}_i$, making closest-

point *snap* projection a compelling candidate for projecting $\mathbf{r}_i$. Such an *anchored closest-point* projection explicitly minimizes $\|\Delta\mathbf{p}_i - \Delta\mathbf{q}_i\|$, but precise minimization is less important than avoiding projection discontinuities and undesirably snapping, even for points close to the mesh. The *smooth-closest-point* projection π_{SCP} formulated in Section 5.1.2 addresses these goals precisely. The *mimicry* projection is then defined as

$$\Pi_{mimicry}(\mathbf{p}_i) = \begin{cases} \pi_{spray}(\mathbf{p}_i) & \text{if } i = 0, \\ \pi_{SCP}(\mathbf{r}_i) & \text{otherwise.} \end{cases} \tag{5.4}$$

5.2.2 Refinements to Mimicry Projection

I now explore further refinements to *mimicry* projection, that could improve curve projection in certain scenarios.

Planar curves are very common in design and visualization [156]. Locally, planarity can be encouraged in mimicry projection by constructing a plane N_i with normal $\Delta\mathbf{p}_i \times \Delta\mathbf{p}_{i-1}$ (i.e. the local plane of the mid-air stroke) and passing through the anchor point $\mathbf{r}_i$ (Figure 5.7b). N_i is then intersected with $\mathcal{M}$. $\mathbf{q}_i$ is defined as the closest-point to $\mathbf{r}_i$ on the intersection curve that contains $\mathbf{q}_{i-1}$. Note that $\pi_{spray}(\mathbf{p}_i)$ is used for $i < 2$, and the most recently defined normal direction (N_{i-1} or prior) is retained when $N_i = \Delta\mathbf{p}_i \times \Delta\mathbf{p}_{i-1}$ is undefined. This method was found to work well for near-planar curves, but the plane is sensitive to noise in the mid-air stroke (Figure 5.9f), and can feel *sticky* or less responsive for non-planar curves.

Offset and parallel surface drawing captures the observation that users tend to draw an intended curve as a corresponding stroke on an imaginary offset or parallel surface of the object $\mathcal{M}$. While users cannot be expected to draw precisely on such surfaces, it is unlikely a user would intentionally draw orthogonal to them.

In scenarios when a user is sub-consciously drawing on an offset surface of $\mathcal{M}$ (an isosurface of its signed-distance function $d_{\mathcal{M}}(\cdot)$), one can remove the component of a user stroke segment along the gradient $\nabla d_{\mathcal{M}}$ when computing the anchor point (Figure 5.7c).

$$\mathbf{r}'_i = \mathbf{q}_{i-1} + \Delta\mathbf{p}_i - \Big(\Delta\mathbf{p}_i \cdot \nabla d_{\mathcal{M}}(\mathbf{p}_i)\Big)\nabla d_{\mathcal{M}}(\mathbf{p}_i). \tag{5.5}$$

Similarly, user strokes can be locally constrained to a parallel surface of $\mathcal{M}$ in Equation 5.6 as

$$\mathbf{r}''_i = \mathbf{q}_{i-1} + \Delta\mathbf{p}_i - \Big(\Delta\mathbf{p}_i \cdot \nabla d_{\mathcal{M}}(\mathbf{r}_i)\Big)\nabla d_{\mathcal{M}}(\mathbf{r}_i). \tag{5.6}$$

Note that the difference from Eq. 5.5 is the position where $\nabla d_{\mathcal{M}}$ is computed, as shown in Figure 5.7d.

A parallel surface better matched user expectation than an offset surface in pilot testing, but both techniques produced poor results when user drawing deviated from these imaginary surfaces (Figure 5.9g–l).

5.2.3 Anchored Raycast Projection

For completeness, I also investigated raycast alternatives to projection of the anchored stroke point $\mathbf{r}_i$. As with *mimicry* refinements, similar priors of local planarity and offset or parallel surface transport were used to define ray directions. Figure 5.8 shows two such options.

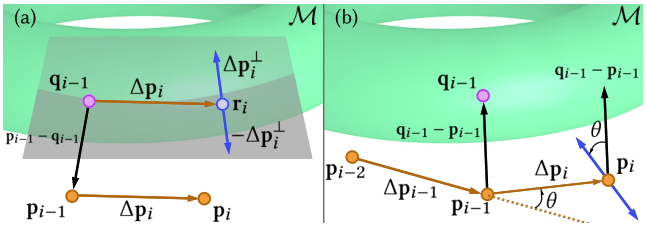


Figure 5.8: Anchored raycast techniques: ray direction defined orthogonal to $\Delta\mathbf{p}_i$ in a local plane (a); parallel transport of ray direction along the user stroke (b). The cast rays (forward/backward) are shown in blue.

In Figure 5.8a, a ray is cast in the local plane of motion, orthogonal to the user stroke, given by $\Delta\mathbf{p}_i$. The local plane of motion is defined as the plane containing $\mathbf{r}_i$ and spanned by $\Delta\mathbf{p}_i$ and $\mathbf{p}_{i-1} - \mathbf{q}_{i-1}$. Then, $\Delta\mathbf{p}_i^{\perp}$ is defined as the direction orthogonal to $\Delta\mathbf{p}_i$ in this plane. Since $\mathbf{r}_i$ may be inside $\mathcal{M}$, two rays are cast bi-directionally $(\mathbf{r}_i, \pm\Delta\mathbf{p}_i^{\perp})$, where

$$\Delta\mathbf{p}_i^{\perp} = \Delta\mathbf{p}_i \times \big(\Delta\mathbf{p}_i \times (\mathbf{p}_{i-1} - \mathbf{q}_{i-1})\big). \tag{5.7}$$

If both rays successfully intersect $\mathcal{M}$, $\mathbf{q}_i$ is chosen to be the point closer to $\mathbf{r}_i$. Unfortunately, as with locally planar mimicry projection, this technique suffered from instability in the local plane.

Motivated by mimicry, I also explored parallel transport of the projection ray direction along the user stroke (Figure 5.8b). For $i > 0$, the previous projection direction $\mathbf{q}_{i-1} - \mathbf{p}_{i-1}$ is parallel transported along the mid-air curve by rotating it using the 3D rotation that aligns $\Delta\mathbf{p}_{i-1}$ with $\Delta\mathbf{p}_i$. Once again, bi-directional rays are cast from $\mathbf{r}_i$, and $\mathbf{q}_i$ is set to the closer intersection with $\mathcal{M}$.

In general, all raycast projections, even when anchored, suffered from unpredictability over long strokes and discontinuities when there are no ray-object intersections (Figure 5.9n,o).

5.2.4 Final Analysis and Implementation Details

In summary, extensive pilot testing of the anchored techniques revealed that they were generally better than context-free approaches, especially when users drew further away from the 3D object. Among anchored techniques, stroke mimicry captured as an *anchored-smooth-closest-point* projection proved to be theoretically elegant and practically the most resilient to ambiguities of user intent and differences of drawing style among users. *Anchored closest-point* can be a reasonable proxy to *anchored smooth-closest-point* when pre-processing is undesirable. A pertinent application is real-time sculpting, where the object shape changes frequently.

These techniques are implemented in C#, with interaction, rendering, and VR support provided by the Unity Engine [247]. For the smooth closest-point operation, I modified Panozzo et al.'s [179] reference implementation, which includes pre-processing code in MATLAB and C++, and real-time code in C++. The real-time projection implementation is exposed to the C# application via a compiled dynamic library. In their implementation, as well as mine, $d = 8$; that is, $\mathcal{M}$ is embedded in $\mathbb{R}^8$. Offset surfaces are computed using `libigl` [113], with $\mu = \|\texttt{BBox}(\mathcal{M})\|/20$. I then improve surface quality using TetWild [104], before computing the tetrahedral mesh $\mathcal{T}_{\mathcal{M}}$ using TetGen [217].

Fast closest-point queries are supported using an AABB tree implemented in `geometry3Sharp` [206]. Signed-distance is also computed using the AABB tree and fast winding number [28], and gradient $\nabla d_{\mathcal{M}}$

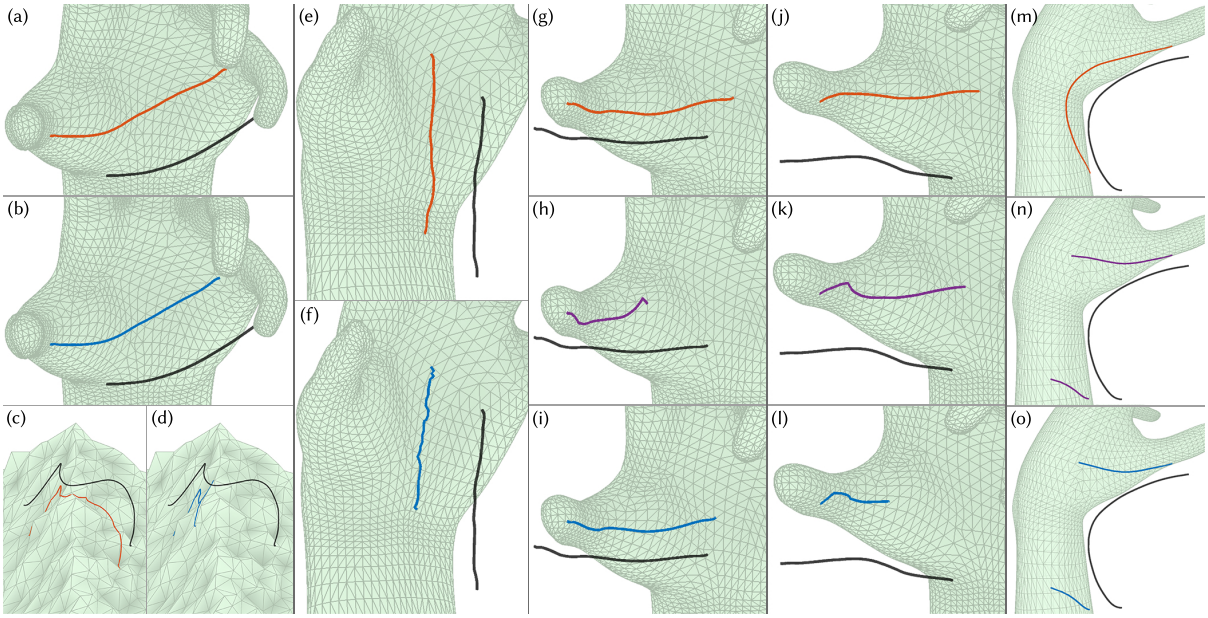


Figure 5.9: *Mimicry* vs. other anchored stroke projections: Mid-air strokes are shown in **black** and *mimicry* curves in **red**. Anchored closest-point (**blue**), is similar to *mimicry* on smooth, low-curvature meshes (a,b) but degrades with mesh detail/noise (c,d). Locally planar projection (**blue**) is susceptible local plane instability (e,f). Parallel (**purple** h,k) or offset (**blue** i,l) surface based projection fail in (h,l) when the user stroke deviates from said surface, while *mimicry* remains reasonable (g, j). Compared to *mimicry* (m), anchored raycasting based on a local plane (**purple** n), or ray transport (**blue** o) can be discontinuous.

computed using central finite differences.

To ease replication of our various techniques and aid future work, I have released our open-source implementation at `github.com/rarora7777/curve-on-surface-drawing-vr`.

I will now formally compare our most promising projection *mimicry* to the best state-of-the-art context-free projection *spraycan*.

5.3 User Study

I designed a user study to compare the performance of the *spraycan* and *mimicry* methods for a variety of curve-drawing tasks. Six shapes were selected for the experiment (Figure 5.10), aiming to cover a diverse range of shape characteristics: sharp features (*cube*), large smooth regions (*trebol*, *bunny*), small details with ridges and valleys (*bunny*), thin features (*hand*), and topological holes (*torus*, *fertility*).

Ten distinct curves were then sampled on the surface of each of the six objects. A canonical task in the study involved the participant attempting to re-create a given *target curve* from this set. Two types of drawing tasks were designed, as shown in Figure 5.11.

Tracing Curves, where a participant tried to trace over a visible target curve using a single smooth stroke.

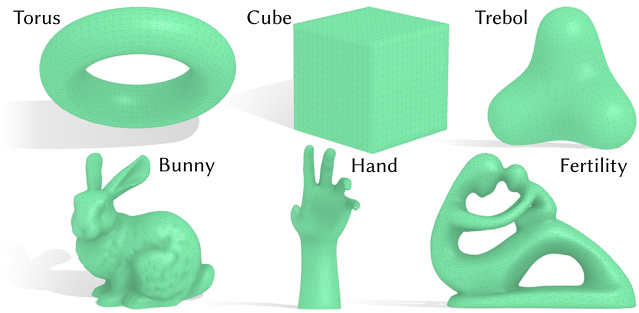


Figure 5.10: The six shapes utilized in the user study. The *torus* shape was used for tutorials, while the rest were used for the recorded experimental tasks.

Re-creating Curves, where a participant attempted to re-create a target curve from memory. The curve was initially shown, but was hidden as soon as the participant started to draw. An enumerated set of keypoints on the curve, however, remained as a visual reference, to aid the participant in re-creating the hidden curve with a single smooth stroke.

The rationale behind asking users to draw target curves is both to control the length, complexity, and nature of curves drawn by users, and to have an explicit representation of the user-intended curve. Curve tracing and re-creating are fundamentally different drawing tasks, each with important applications (Section 3.3). The curve re-creation task is designed to capture free-form drawing, with minimal visual suggestion of the intended target curve.

Further, the aim was to design target curves that could be executed using a single smooth motion. Since users typically draw sharp corners using multiple strokes [22], the target curves were constrained to be smooth, created using cardinal cubic B-splines on the meshes, computed using Panozzo et al. [179]. The length and curvature complexity of the curves was also controlled, as pilot testing showed that very simple and short curves can be reasonably executed by almost any projection technique. Curve length and complexity is modelled by placing spline control points at mesh vertices, and specifying the desired geodesic distance and Gauß map distance between consecutive control points on the curve.

A target curve is represented using four parameters $\langle n, i_0, k_G, k_N \rangle$, where n is the number of spline control points, i_0 the vertex index of the first control point, and k_G, k_N constants that control the geodesic and normal map distance between consecutive control points. The desired geodesic distance between consecutive control points is defined as $D_G = k_G \times \|\texttt{BBox}(\mathcal{M})\|$, where $\|\texttt{BBox}(\mathcal{M})\|$ is the length of the bounding box diagonal of $\mathcal{M}$. The desired Gauß map distance (angle between the unit vertex normals) between consecutive control points is simply k_N.

A target curve $\mathbf{C}_0, \ldots, \mathbf{C}_{n-1}$ starting at vertex $\mathbf{v}_{i_0}$ of the mesh is generated incrementally for $i > 0$ as

$$\mathbf{C}_i = \underset{\mathbf{v} \in V'}{\arg\min} \ \big(d_G(\mathbf{C}_{i-1}, \mathbf{v}) - D_G\big)^2 + \big(d_N(\mathbf{C}_{i-1}, \mathbf{v}) - k_N\big)^2, \tag{5.8}$$

where d_G and d_N compute the geodesic and normal distance between two points on $\mathcal{M}$, and $V' \subset V$ contains only those vertices of $\mathcal{M}$ whose geodesic distance from $\mathbf{C}_0, \ldots, \mathbf{C}_{i-1}$ is at least $D_G/2$. The restricted subset of vertices conveniently helps prevent (but doesn't fully avoid) self-intersecting or nearly self-intersecting curves. Curves with complex self-intersections are less important in actual application scenarios, and can be particularly confusing for the curve re-creation task. All the target curve samples

were generated using $k_G \in [0.05, 0.25]$, $k_N \in [\pi/6, 5\pi/12]$, $n = 6$, and a randomly chosen i_0. The curves were manually inspected for self-intersections, and infringing curves rejected.

Keypoints were then defined on the target curves as follows. First, curve endpoints were chosen as keypoints. This was followed by greedily picking extrema of geodesic curvature, while ensuring that the arclength distance between any two consecutive keypoints was at least 3cm. Positioning keypoints at curvature extrema ensured that curve re-creating tasks amounted to smoothly joining the keypoints, rather than testing participants' memory. The procedure concluded when the maximum arclength distance between any consecutive keypoints was below 15cm. The resulting target curves had between 4–9 keypoints (including endpoints).

5.3.1 Experiment Design

The main variable studied in the experiment was *projection method*—*spraycan* vs. *mimicry*—realized as a within-subjects variable. The order of methods was counterbalanced between participants. For each method, participants were exposed to all the six objects. Object order was fixed as torus, cube, trebol, bunny, hand, and fertility, based on our personal judgment of drawing difficulty. The torus was used as a tutorial, where participants had access to additional instructions visible in the scene and the strokes were not utilized for analysis. For each object, the order of the 10 target strokes was randomized. The first five were used for the tracing curves task, while the remaining five were used for re-creating curves.

The target curve for the first tracing task was repeated after the five unique curves, to gauge user consistency and learning effects. A similar repetition was used for curve re-creation. Participants thus performed 12 curve drawing tasks per object, leading to a total of 12×5 (objects) $\times$ 2 (projections) $= 120$ strokes per participant.

Owing to the COVID-19 physical distancing guidelines, the study was conducted on participants' personal VR equipment at their homes. A 15-minute instruction video introduced the study tasks and the two projection methods. Participants then filled out a consent form and a questionnaire to collect demographic information. This was followed by them testing the first projection method and filling out a questionnaire to express their subjective opinions of the method. They then tested the second method, followed by a similar questionnaire, and questions involving subjective comparisons between the two methods. Participants were required to take a break after testing the first method, and were also encouraged to take breaks after drawing on the first three shapes for each method. The study took approximately an hour, including the questionnaires.

5.3.2 Participants

Twenty participants (5 female, 15 male) aged 21–47 from five countries participated in the study. All but one were right-handed. Participants were not selected for artistic ability or prior VR experience, and exhibited a diverse range of self-reported artistic abilities (min. 1, max. 5, median 3 on a 1–5 scale) as well as varying degrees of VR experience, ranging from below 1 year to over 5 years. 13 participants had a technical computer graphics or HCI background, while ten had experience with creative tools in VR, with one reporting professional usage. Participants were paid CA$ 30 as a gift card.

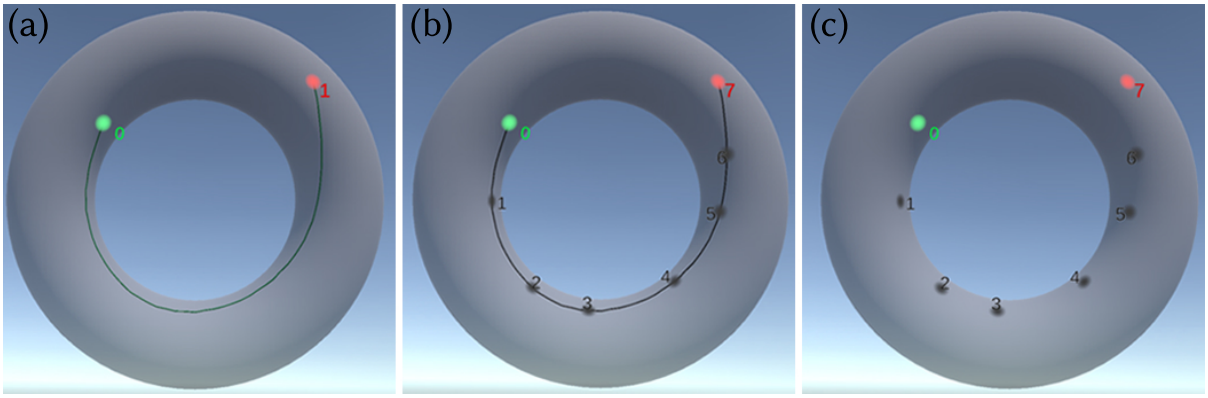


Figure 5.11: Study tasks—*curve tracing*: target curve is visible when drawing (a), and *curve re-creation*: target curve (b) is hidden when drawing (c).

5.3.3 Apparatus

As the study was conducted on personal VR setups, a variety of commercial VR devices were utilized—Oculus Rift, Rift S, and Quest using Link cable, HTC Vive and Vive Pro, Valve Index, and Samsung Odyssey using Windows Mixed Reality. All but one participant used a standing setup allowing them to freely move around.

5.3.4 Procedure

Before each trial, participants could use the "grab" button on their controller (in the dominant hand) to grab the mesh to position and orient it as desired. The trial started as soon as the participant started to draw by pressing the "main trigger" on their dominant hand controller. This action disabled the grabbing interaction—participants could not draw and move the object simultaneously. As noted earlier, for curve re-creation tasks, this had the additional effect of hiding the target curve, but leaving keypoints visible.

5.4 Study Results and Discussion

The study code recorded the head position $\mathbf{h}$ and orientation $\hbar$, controller position $\mathbf{c}$ and orientation ϵ, projected point $\mathbf{q}$, and timestamp t, for each mid-air stroke point $\mathbf{p} = \mathbf{c}$. In this section, I will refer to a target curve as $\mathcal{X}$, a mid-air stroke as $\mathcal{P}$, and a projected curve as $\mathcal{Q}$.

5.4.1 Data Processing and Filtering

Three criteria were formulated to filter out meaningless user strokes.

Short Curves. Projected curves $\mathcal{Q}$ that are too short as compared to the length of the target curves $\mathcal{X}$ were ignored. The criterion was set conservatively to ignore curves less than half as long as the target curve. While it is possible that the user stopped drawing mid-way out of frustration, it is more likely that they prematurely released the controller trigger by accident. Both curve lengths are computed in $\mathbb{R}^3$ for efficiency.

Stroke Noise. Strokes for which the mid-air stroke is too noisy were also ignored. Specifically, mid-air strokes with distant consecutive points ($\exists\ i$ s.t. $\|\mathbf{p}_i - \mathbf{p}_{i-1}\| > 5$cm) are rejected.

Inverted Curves. While the study program labelled keypoints with numbers and marked start and end points in green and red (Figure 5.11), some users occasionally drew the target curve in reverse. The motion to draw a curve in reverse is not symmetric, and such curves are thus rejected. Inverted strokes are detected by looking at the indices $i_0, i_1, \ldots, i_l$ of the points in $\mathcal{Q}$ which are closest to the keypoints $\mathbf{x}_{k_0}, \mathbf{x}_{k_1}, \ldots, \mathbf{x}_{k_l}$ of $\mathcal{X}$. Ideally, the sequence $i_0, \ldots, i_l$ should have no inversions, i.e., $\forall\ 0 \leq j < k \leq l,\ i_j \leq i_k$; and a maximum of $l(l+1)/2$ inversions if $\mathcal{Q}$ is aligned in reverse with $\mathcal{X}$. Curves with more than $l(l+1)/4$ (half the maximum) inversions were considered to be inadvertently inverted and reject them. As before, distances are computed in $\mathbb{R}^3$ for efficiency.

Despite conducting the experiment remotely without supervision, 95.8% of the strokes were found to satisfy the criteria and could be utilized for analysis. Out of the 102 strokes deemed unfit for analysis, 17 were too short, 66 were inverted, and 38 exhibited excessive tracking noise. For comparisons between π_{spray} and $\pi_{mimicry}$, stroke pairs where either stroke did not satisfy the quality criteria were rejected. Out of 1200 pairs (2400 total strokes), 1103 (91.9%) satisfied the quality criteria and were used for analysis, including 564 pairs for the curve re-creation task and 539 for the tracing task.

Table 5.2: Quantitative results (mean ± std. dev.) of the comparisons between *mimicry* and *spraycan* projection. All measures are analyzed using Wilcoxon signed-rank tests, lower values are better, and significantly better values ($p < .05$) are shown in **boldface**. Accuracy, aesthetic, and physical effort measures are shown with green, red, and blue backgrounds, respectively.

Tracing Curves				
Measure	**Spraycan**	**Mimicry**	***p*-value**	***z*-stat**
D_{ep}	2.31 ± 2.64 mm	**1.13 ± 1.11 mm**	< .001	8.36
D_{sym}	0.64 ± 0.66 mm	0.56 ± 0.44 mm	> .05	-0.09
K_E	280 ± 262 rad/m	**174 ± 162 rad/m**	< .001	15.59
K_g	249 ± 245 rad/m	**152 ± 157 rad/m**	< .001	15.42
F_g	394 ± 413 rad/m	**248 ± 285 rad/m**	< .001	14.82
T_h	0.81 ± 0.70	**0.58 ± 0.40**	< .001	7.93
R_h	1.63 ± 2.18 rad/m	**1.18 ± 1.63 rad/m**	< .001	4.82
T_c	**1.05 ± 0.36**	1.10 ± 0.29	< .001	-3.36
R_c	5.12 ± 5.88 rad/m	**3.79 ± 4.84 rad/m**	< .001	5.51
τ	**4.69 ± 1.85 s**	5.29 ± 2.17 s	< .001	-7.32
Re-creating Curves				
Measure	**Spraycan**	**Mimicry**	***p*-value**	***z*-stat**
D_{ep}	2.34 ± 2.49 mm	**2.24 ± 23.32 mm**	< .001	8.63
D_{sym}	0.75 ± 0.65 mm	1.12 ± 11.51 mm	> .05	0.55
K_E	254 ± 236 rad/m	**155 ± 127 rad/m**	< .001	14.70
K_g	223 ± 219 rad/m	**132 ± 123 rad/m**	< .001	14.95
F_g	348 ± 371 rad/m	**215 ± 227 rad/m**	< .001	14.11
T_h	0.72 ± 0.54	**0.54 ± 0.35**	< .001	6.78
R_h	1.50 ± 2.19 rad/m	**1.32 ± 1.99 rad/m**	.002	3.07
T_c	**1.05 ± 0.37**	1.11 ± 0.23	< .001	-5.94
R_c	5.23 ± 6.36 rad/m	**3.63 ± 5.13 rad/m**	< .001	4.00
τ	**4.33 ± 1.57 s**	4.92 ± 1.89 s	< .001	-7.12

5.4.2 Quantitative Analysis

Ten different statistical measures were defined (Table 5.2) to compare π_{spray} and $\pi_{mimicry}$ curves in terms of their accuracy, aesthetic, and effort in curve creation. I consistently use the non-parametric Wilcoxon signed rank test for all quantitative measures instead of a parametric test such as the paired t-test, since the recorded data for none of the measures was found to be normally distributed (normality hypothesis rejected via the Kolmogorov-Smirnov test, $p < .005$). In addition, I analyze users' tendency to mimic the target strokes and consistency between repeated strokes.

Curve Accuracy

Accuracy is computed using two measures of distance between points on the projected curve $\mathcal{Q}$ and target curve $\mathcal{X}$. Both curves are densely re-sampled using $m = 101$ sample points equi-spaced by arc-length.

Given $\mathcal{Q} = \mathbf{q}_0, \ldots, \mathbf{q}_{m-1}$ and $\mathcal{X} = \mathbf{x}_0, \ldots, \mathbf{x}_{m-1}$, the *average equi-parameter distance* D_{ep} is computed as

$$D_{ep}(\mathcal{Q}) = \frac{1}{m} \sum_{i=0}^{m-1} d_E\left(\mathbf{q}_i, \mathbf{x}_i\right), \tag{5.9}$$

where d_E computes the Euclidean distance between two points in $\mathbb{R}^3$. The second accuracy measure, *average symmetric distance* D_{sym}, is computed as

$$D_{sym}(\mathcal{Q}) = \frac{1}{2m} \sum_{i=0}^{m-1} \left(\min_{\mathbf{x} \in X} d_E\left(\mathbf{q}_i, \mathbf{x}\right) \right) + \frac{1}{2m} \sum_{i=0}^{m-1} \left(\min_{\mathbf{q} \in Q} d_E\left(\mathbf{q}, \mathbf{x}_i\right) \right) \tag{5.10}$$

In other words, D_{ep} computes the distance between corresponding points on the two curves and D_{sym} computes the average minimum distance from each point on one curve to the other curve.

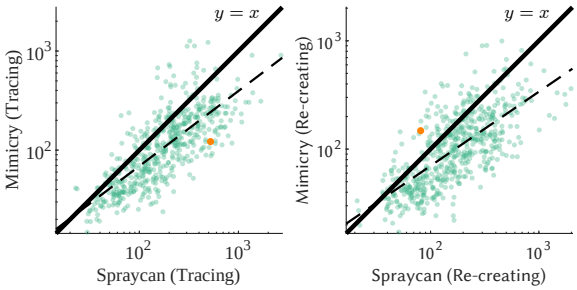
For both tracing and re-creation tasks, D_{ep} indicated that *mimicry* produced significantly better results than *spraycan* (see Table 5.2, Figure 5.1c, 5.12). The D_{sym} difference was not statistically significant, evidenced by users correcting their strokes to stay close to the intended target curve (at the expense of curve aesthetic).

Curve Aesthetic

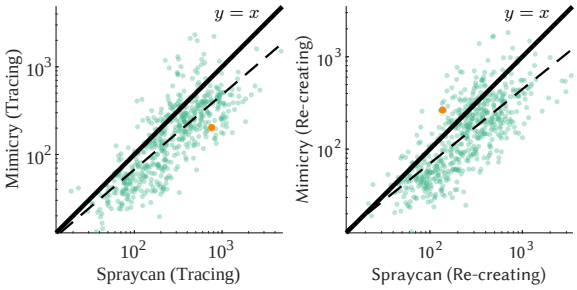
For most design applications, jagged projected curves, even if geometrically quite accurate, are aesthetically undesirable [155]. Curvature-based measures are typically used to measure fairness of curves. Here, I report three such measures of curve aesthetic for the projected curve $\mathcal{Q} = \mathbf{q}_0, \ldots, \mathbf{q}_{n-1}$. First, $\mathcal{Q}$ is refined by computing the exact geodesic on $\mathcal{M}$ between consecutive points of $\mathcal{Q}$ [229], to create $\widehat{\mathcal{Q}}$ with points $\widehat{\mathbf{q}}_0, \ldots, \widehat{\mathbf{q}}_{k-1}$, $k \geq n$. The curvature measures are normalized using $L_{\mathcal{X}}$, the length of the corresponding target stroke $\mathcal{X}$. The *normalized Euclidean curvature* for $\mathcal{Q}$ is defined as

$$K_E(\mathcal{Q}) = \frac{1}{L_{\mathcal{X}}} \sum_{i=1}^{k-2} \theta_i \tag{5.11}$$

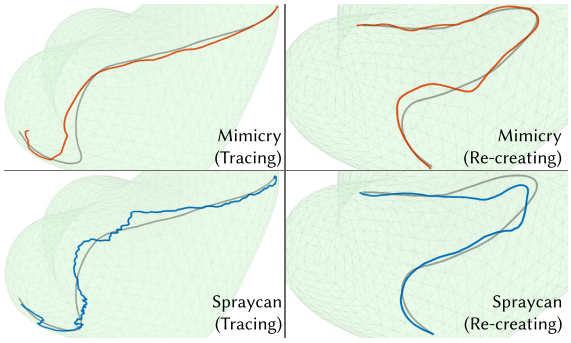
where θ_i is the angle between the two segments of $\widehat{\mathcal{Q}}$ incident on $\widehat{\mathbf{q}}_i$. Thus, K_E is the total discrete curvature of $\widehat{\mathcal{Q}}$, normalized by the target curve length.



(a) Normalized geodesic curvature K_g.



(b) Normalized fairness deficiency F_g.



(c) Example strokes, **orange** points in (a, b) above.

Figure 5.12: Curvature measures (a,b) indicate that *mimicry* produces significantly smoother and fairer curves than *spraycan* for both tracing (left) and re-creating tasks (right). Pairwise comparison plots between *mimicry* (y-axis) and *spraycan* (x-axis), favour *mimicry* for the vast majority of points (points below the $y = x$ line). A linear regression fit (on the log plots) is shown as a dashed line. Example curve pairs (**orange** points) for curve tracing and re-creating are also shown with the target curve $\mathcal{X}$ shown in gray (c).

Since $\widehat{\mathcal{Q}}$ is embedded in $\mathcal{M}$, we can also compute discrete *geodesic* curvature, computed as the deviation from the straightest geodesic on a surface. Using a signed θ_i^g defined at each point $\widehat{\mathbf{q}}_i$ [188], the *normalized geodesic curvature* is computed as

$$K_g(\mathcal{Q}) = \frac{1}{L_{\mathcal{X}}} \sum_{i=1}^{k-2} |\theta_i^g|. \tag{5.12}$$

Finally, similar to Section 3.3.7, fairness is approximated via first-order variations in geodesic curvature, thus defining the *normalized fairness deficiency* as

$$F_g(\mathcal{Q}) = \frac{1}{L_{\mathcal{X}}} \sum_{i=2}^{k-2} |\theta_i^g - \theta_{i-1}^g|, \tag{5.13}$$

For all three measures, a lower value indicates a smoother, pleasing, curve. Wilcoxon signed-rank tests on all three measures indicated that *mimicry* produced significantly smoother and better curves than *spraycan* (Table 5.2).

Physical Effort

The amount of head (HMD) and hand (controller) movement, and stroke *execution time* τ provide quantitative proxies for physical effort.

For head and hand translation, the position data is first filtered with a Gaussian-weighted moving average filter with $\sigma = 20$ms. The *normalized head/controller translation* measures T_h and T_c are then defined as the length of the poly-line swept by the filtered head/controller positions, normalized by the length of the target curve $L_{\mathcal{X}}$.

An important ergonomic measure is the amount of head/hand rotation required to draw the mid-air stroke. The forward and up vectors of the head/controller frame are first denoised or filtered, using the same filter as for positional data. The frames are then re-orthogonalized and the length of the curve defined by the filtered orientations in SO(3) is computed, using the angle between consecutive orientation data-points. The *normalized head/controller rotation* R_h and R_c are defined as the respective orientation curve lengths, normalized by $L_{\mathcal{X}}$.

Table 5.2 summarizes the physical effort measures. Lower controller translation (effect size $\approx 5\%$) and execution time (effect size $\approx 12\%$) were observed for *spraycan*; while head translation and orientation were lower (effect sizes $\approx 36\%, 26\%$) for *mimicry*. Noteworthy is the significantly reduced controller rotation using *mimicry*, with *spraycan* unsurprisingly requiring 35% (tracing) and 44% (re-creating) more hand rotation from the user.

Quantifying Users' Tendency to Mimic

The study also provided an opportunity to test if the users actually tended to mimic their intended curve $\mathcal{X}$ in the mid-air stroke $\mathcal{P}$. To quantify the "mimicriness" of a stroke, $\mathcal{P}$ and $\mathcal{X}$ are subsampled to m points as in Section 5.4.2. Then, using a correspondence as in Eq. 5.9 and looking at the variation in the distance (distance between the closest pair of corresponding points subtracted from that of the farthest pair) as a percentage of the target length $L_{\mathcal{X}}$ gives the *mimicry violation* measure for a stroke. Intuitively, the lower the *mimicry violation*, the closer the stroke $\mathcal{P}$ is to being a perfect mimicry of $\mathcal{X}$, going to zero if it is a precise translation of $\mathcal{X}$. Notably, users depicted very similar trends to mimic for both the techniques—with 86% of *mimicry* and 80% of *spraycan* strokes exhibiting *mimicry violation* below 25% of $L_{\mathcal{X}}$, and 71%, 66% below 20% of $L_{\mathcal{X}}$—suggesting that mimicry is indeed a natural tendency.

Consistency across Repeated Strokes

Recall that users repeated 2 of the 10 strokes per shape for both the techniques. To analyze consistency across the repeated strokes, the values of the stroke accuracy measure D_{eq} and the aesthetic measure F_g between the original stroke and the corresponding repeated stroke were compared. Specifically, the relative change $|f(i) - f(i')|/f(i)$ was measured, where (i, i') is a pair of original and repeated strokes,

and $f(\cdot)$ is either D_{eq} or F_g. Users were fairly consistent across both the techniques, with the average consistency for D_{eq} being 35.4% for *mimicry* and 36.8% for *spraycan*, while for F_g, it was 36.5% and 34.1%, respectively. Note that the averages were computed after removing extreme outliers outside the 5σ threshold.

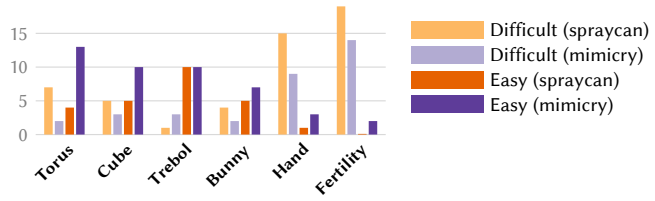
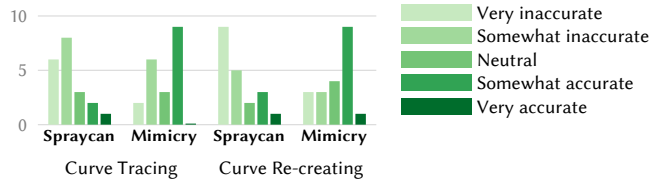
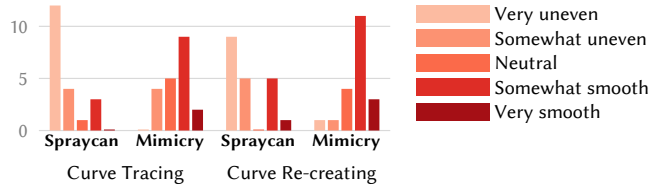


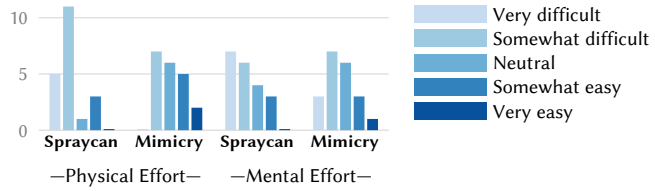
Figure 5.13: Perceived difficulty of drawing for the six 3D shapes in the study.



(a) Perceived accuracy.



(b) Perceived smoothness.



(c) Physical and mental effort ratings.

Figure 5.14: Participants perceived *mimicry* to be better than *spraycan* in terms of accuracy (a), curve aesthetic (b), and user effort (c).

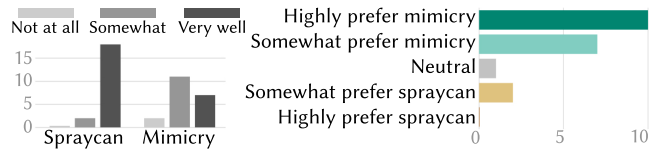


Figure 5.15: Participants stated understanding *spraycan* projection better (left); 17/20 users stated an overall preference for *mimicry* over *spraycan* (right).

5.4.3 Qualitative Analysis

The mid- and post-study questionnaires elicited qualitative responses from participants on their perceived difficulty of drawing, curve accuracy and smoothness, mental and physical effort, understanding of the projection methods, and overall method of preference.

Participants were asked to specify the objects which they found especially easy or difficult to draw on, when using either of the two projection methods. In general, the shapes shown earlier were judged to be easier to work with (Figure 5.13), validating the ordering of shapes in the experiment based on expected drawing difficulty. Importantly, this observation also suggests a lack of any learning effects caused by the fixed object ordering.

Accuracy, smoothness, physical/mental effort responses were collected on 5-point Likert scales. For the purpose of reporting the qualitative results, the choices have been consistently ordered from 1 (worst) to 5 (best) in terms of user experience. All summary scores reported below represent the median (M) response. *Mimicry* was perceived to be a more accurate projection (tracing, re-creating $M = 3, 3.5$) compared to *spraycan* ($M = 2, 2$), with nine participants perceiving their traced curves to be either *very accurate* or *somewhat accurate* with *mimicry*, compared to only two for *spraycan* (Figure 5.14a). Perception of stroke smoothness was also consistent with the quantitative results, with *mimicry* (tracing, re-creating $M = 4, 4$) clearly outperforming *spraycan* (tracing, re-creating $M = 1, 2$) (Figure 5.14b). Lastly, with no need for controller rotation, *mimicry* ($M = 3$) was perceived as less physically demanding than *spraycan* ($M = 2$), as expected (Figure 5.14c).

The response to understanding and mental effort was more complex. *Spraycan*, with its physical analogy and mathematically precise definition was clearly understood by all 20 participants (17 very well, 3 somewhat) (Figure 5.15a). *Mimicry*, conveyed to the participants as "drawing a mid-air stroke on or near the object as similar in shape as possible to the intended projection", was not as well understood (7 very well, 11 somewhat, 3 not at all). Despite not understanding the method consciously, the 3 participants were able to create curves that were both accurate and smooth. Further, participants perceived *mimicry* ($M = 2.5$) as less cognitively taxing than *spraycan* ($M = 2$) (Figure 5.14c). This may be because users were less prone to consciously controlling their stroke direction and rather focused on drawing. The tendency to mimic may have thus manifested sub-consciously, as had been observed in pilot testing.

The most important qualitative question was user preference (Figure 5.15b). 85% of the 20 participants preferred *mimicry* (10 highly preferred, 7 somewhat preferred). The remaining users were neutral (1/20) or somewhat preferred *spraycan* (2/20).

5.4.4 Participant Feedback

Participants were also asked to elaborate on their stated preferences and ratings. Participants (*P4,8,16,17*) noted discontinuous "jumps" caused by *spraycan*, and felt the continuity guarantee of *mimicry*: "seemed to deal with the types of jitter and inaccuracy VR setups are prone to better" *(P6)* ; "could stabilize my drawing" *(P9)* . *P9,15* felt that *mimicry* projection was smoothing their strokes. In reality, no smoothing was actually employed. This may be the effect of noise and inadvertent controller rotation, which can cause large variations with *spraycan* but is completely ignored by the *mimicry* projection, being perceived as curve smoothing.

Some participants (*P4,17*) felt that rotating the hand smoothly while drawing was difficult, while

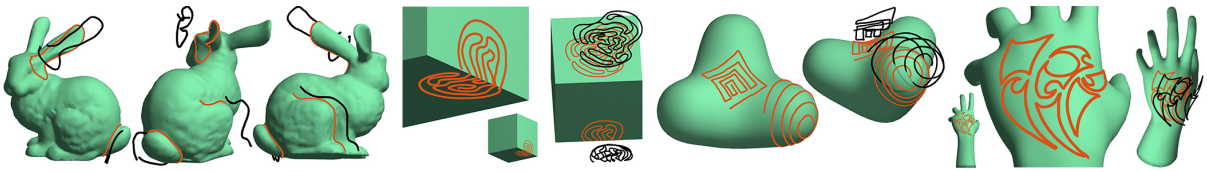


Figure 5.16: Gallery of free-form curves in **red**, drawn by the author of this thesis using *mimicry*. (Left to right) tracing geometric features on the bunny, maze-like curves on the cube, maze with sharp corners and a spiral on the trebol, and artistic tattoo motifs on the hand. Some mid-air strokes (**black**) hidden for clarity.

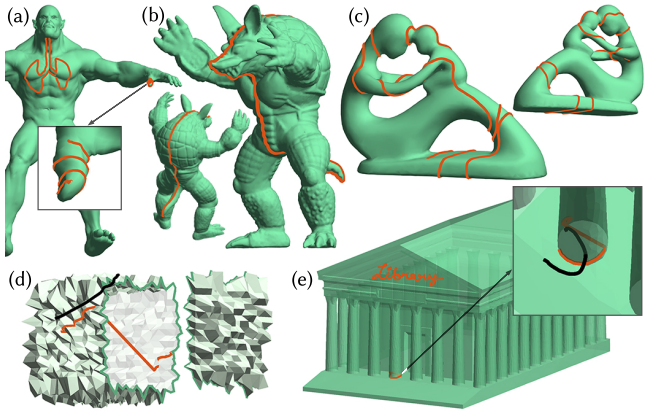


Figure 5.17: *Mimicry* can be used to draw long, complicated curves on complex high-resolution meshes. Here, I show strokes on high-resolution meshes (a, b, e), a long stroke bisecting a model (b), and a single stroke winding around a topologically non-trivial object multiple times (c). However, excessive noise in the input mesh can break the underlying π_{ACP} assumptions, resulting in catastrophic failure (d). Mesh has been cut open for visualization. Large meshes with many sharp features and topological complexity can also show smaller local failures in the form of unexpected jumps when drawing close to the sharp features (e, inset).

others missed the *spraycan* ability to simply use hand rotation to sweep out long projected curves from a distance (*P2,7*). Participants commented on physical effort: "Mimicry method seemed to required [sic] much less head movement, hand rotation and mental planning" *(P4)* .

Participants appreciated the anchored control of *mimicry* in high-curvature regions (*P1,2,4,8*) and also noted that with *spraycan*, "the curvature of the surface could completely mess up my stroke" *(P1)* . Some participants did feel that *spraycan* could be preferable when drawing on near-flat regions of the mesh (*P3,14,19,20*).

Finally, participants who preferred spraycan felt that mimicry required more thinking: "with mimicry, there was extra mental effort needed to predict where the line would go on each movement" *(P3)* , or because mimicry felt "unintuitive" *(P7)* due to their prior experience using a *spraycan* technique. Some who preferred *mimicry* found it difficult to use initially, but felt it got easier over the course of the experiment (*P4,17*).

5.5 Applications

Complex 3D curves on arbitrary surfaces can be drawn in VR with a single stroke, using *mimicry* (Figure 5.16). Drawing such curves on 3D virtual objects is fundamental to many applications, including

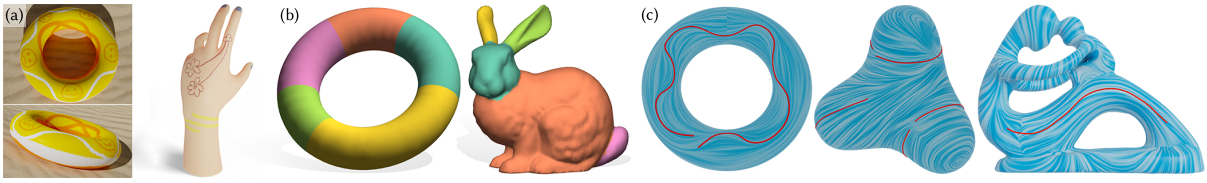


Figure 5.18: Applications of *mimicry* projection. Texture painting (a), interactive segmentation by drawing curves onto meshes (b), and providing constraints (**red** curves) to guide the vector field generation of Fisher et al. [78] (c).

direct painting of textures [207]; tangent vector field design [78]; texture synthesis [144, 244]; interactive selection, annotation, and object segmentation [41]; and seams for shape parametrization [146, 189, 203], registration [83], and quad meshing [242]. I showcase the utility and quality of *mimicry* curves within example applications.

I also stress-tested the technique by drawing curves on complex models (Figure 5.17a,b,e) and drawing a single long curve looping around the fertility model multiple times (Figure 5.17c). Finally, I show a failure case discussed in Section 5.1.2—the *mimicry* projection fails catastrophically due to problems in the underlying π_{ACP} projection when the mesh is perturbed with excessive random noise (Figure 5.17d). Smaller local jumps can also occur when the model is both highly detailed and contains many sharp features (see inset in Figure 5.17e).

Texture Painting. Figures 5.1e, 5.18a show examples of textures painted in VR using *mimicry*. The long, smooth, wraparound curves on the torus, are especially hard to draw with 2D interfaces. To create brush strokes or geometric stamps on the object, my implementation uses Discrete Exponential Maps (DEM) [207] to compute a dynamic local parametrization around each projected point $\mathbf{q}_i$.

Mesh Segmentation. Figures 5.1e, 5.18b show interactive segmentation using *mimicry*. In my implementation, users draw an almost-closed curve $\mathcal{Q} = \{\mathbf{q}_0, \ldots, \mathbf{q}_{n-1}\}$ on the object using *mimicry*. Points $\mathbf{q}_i$ are snapped to their nearest mesh vertex, and Dijkstra's shortest path algorithm is used to connect consecutive vertices and to close the cycle. While easy in VR via *mimicry*, drawing similar strokes in 2D for selection/segmentation would require multiple view changes.

Vector Field Design. Vector fields on meshes are commonly used for texture synthesis [244], guiding fluid simulations [224], and non-photorealistic rendering [99]. *Mimicry* curves can provide soft constraints to guide the vector field generation of Fisher et al. [78]. Figure 5.18c shows example vector fields, visualized using Line Integral Convolutions [37] in the texture domain.

5.6 Conclusion

In this chapter, I presented a detailed investigation of the problem of real-time inked drawing on 3D virtual objects in immersive environments. I have shown the importance of stroke context when projecting mid-air 3D strokes, and explore the design space of anchored projections. A 20-participant study showed *mimicry* to be preferred over the established *spraycan* projection for projecting 3D strokes onto objects in VR. Both *mimicry* projection and performing VR studies in the wild do have some limitations.

Further, while user stroke processing for 2D interfaces is a mature field of research, mid-air stroke processing for VR/AR is relatively nascent, with many directions for future work. This study contributes a high-quality VR data corpus comprising ≈ 2400 user strokes, projected curves, intended target curves, and corresponding system states, useful for future data-driven techniques for mid-air stroke processing.

"In the wild" VR Study Limitations. Ongoing pandemic restrictions (during summer 2020) presented both a challenge and an opportunity to remotely conduct a more natural study in the wild, with a variety of consumer VR hardware and setups. The enthusiasm of the VR community allowed me to readily recruit 20 diligent users, albeit with a bias towards young, adult males. While the variation in VR headsets seemed to be of little consequence, differences in controller grip and weight can certainly impact mid-air drawing posture and stroke behaviour. Controller size is also significant: a larger Vive controller, for example, has a higher chance of occluding target objects and projected curves, as compared to a smaller Oculus Touch controller. The impact of controller size could be mitigated by rendering a standard drawing tool, but preference was instead given to render the familiar, default controller that matched the physical device in participants' hands. Further, no participant explicitly mentioned the controller getting in the way of their ability to draw.

Mimicry Limitations. The lack of a concise mathematical definition of observed stroke mimicry, makes it harder to precisely communicate it to users. While a precise mathematical formulation may exist, conveying it to non-technical users can still be a challenging task. *Mimicry* ignores controller orientation, producing smoother strokes with less effort, but can give participants a reduced sense of sketch control (*P2,3,6*). I hypothesize that the reduced sense of control is in part due to the tendency for anchored smooth-closest-point to shorten the user stroke upon projection, sometimes creating a feeling of lag. *Spraycan* like techniques, in contrast, have a sense of amplified immediacy, and the explicit ability to make lagging curves catch-up by rotating a controller in place.

5.6.1 Future work

In this chapter, my goal was to develop a general real-time inked projection with minimal stroke context via anchoring. Optimizing the method to account for the entire partially projected stroke may improve the projection quality. Relaxing the restriction of real-time inking would allow techniques such as spline fitting and global optimization that can account for the entire user stroke and geometric features of the target object. Local parametrizations such as DEM (Section 5.5) can be used to incrementally grow or shrink the projected curve, so it does not lag the user stroke. Hybrid projections leveraging both proximity and raycasting are also subject to future work.

On the interactive side, I experimented with feedback to encourage users to draw closer to a 3D object. For example, I tried varying the appearance of the line connecting the controller to the projected point based on line length; or providing aural/haptic feedback if the controller got further than a certain distance from the object. While these techniques can help users in specific drawing or tracing tasks, they were found to be distracting and harmful to stroke quality for general stroke projection. Bimanual interaction, such as rotating the shape with one hand while drawing on it with the other (suggested by *P3,19*) can also be explored. To generalize this work to AR, the impact of rendering quality and perception of virtual models also needs to be studied in the future. Drawing on physical objects in AR is another related research direction.

Application-dependent optimizations to encourage closed strokes, snapping to geometric features, or alignment with existing user-drawn curves can also be explored in the future. Further, the user study only focused on smooth curves. While I show author-drawn example of curves with sharp features (Figure 5.16), formally testing the *mimicry* technique for drawing such curves and potentially optimizing the projection to deal with sharp features is an important future direction. But perhaps the most exciting area of future work is data-driven techniques for inferring the intended projection, perhaps customized to the drawing style of individual users. The study code and data has been made publicly available at `github.com/rarora7777/curve-on-surface-drawing-vr` to aid in such endeavours.

Chapter 6

Ideation and Concept Modelling in Virtual Reality

Contribution Statement

This work is part of a collaboration. Emilie Yu is the primary contributor and the author of this dissertation is the secondary. The author of this dissertation contributed to the following.

— Design of the user interactions (Section 6.1).

— Development of the continuous optimization for the curve network solver (Section 6.2).

— Design and conduction of the user experience study (Section 6.4).

— Using the system for creating concept models and generating the downstream application results (Section 6.5, Figures 6.14, 6.16).

— Writing the associated peer-reviewed publication [271].

Inclusion of this work in this thesis has been permitted by Emilie Yu and her Master's supervisor, Dr. Jakob Andreas Bærentzen, Associate Professor at Danmarks Tekniske Universitet (the Technical University of Denmark).

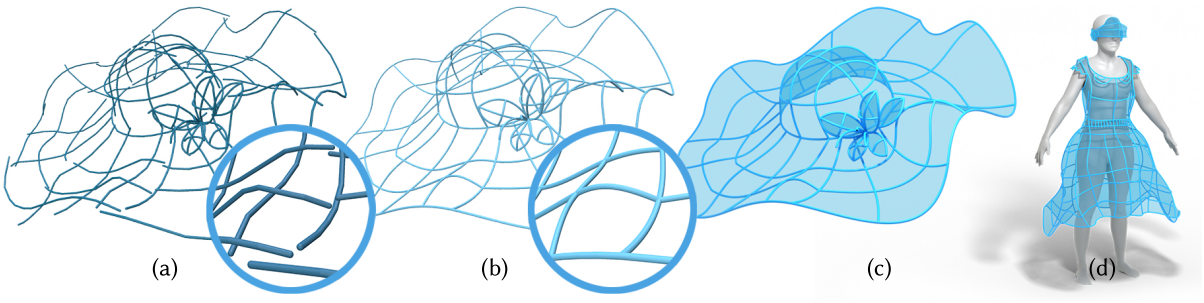


Figure 6.1: Freehand 3D sketching in VR/AR allows rapid conceptualization of design ideas (a). However, 3D inputs are prone to large inaccuracies (a, inset) and sketches cannot be utilized in downstream design pipelines. Our novel 3D sketching system, CASSIE, allows the creation of clean, well-connected 3D curve networks by performing automatic stroke neatening (b). These curve networks are augmented by our on-the-fly cycle detection and surfacing method (c) which improves shape perception by providing occlusion cues. We evaluated CASSIE with 12 users and utilized it for creating 3D concepts for a variety of application domains (d).

The digital 3D realization of a mental design construct typically progresses from exploratory *ideation* to a *concept model* which is subsequently processed for presentation, structural analysis, or manufacturing [65, 184].

Freehand sketching, traditionally on pen and paper, dominates the ideation process. A number of digital 2D drawing interfaces support the creation [22, 63, 174] and exploration [14] of ideation sketches. These sketches serve as a visual reference for concept 3D CAD modelling [192], though recent works like Analytic Scaffolding [209] and True2Form [267] have shown that 2D design sketches can be lifted into 3D curve networks, effectively bridging 2D interfaces for design sketching and 3D concept modelling.

This chapter describes a novel VR creation tool called *CASSIE*. CASSIE stands for curve and surface sketching in immersive environments. It allows *freehand* 3D ideation, as well as automatic optimization of 3D strokes to create curve network *armatures* and predictively surfaces *patches*, thus enabling both ideation and concept modelling through a single interface.

Existing research and commercial software for 3D modelling in VR is based on disparate metaphors of free-form sculpting [71], CAD-like curve and surface creation [91], curve networks [255], swept surfaces [58], and even coarse-to-fine poly modelling [86]. In contrast, CASSIE is the first general ideation and concept modelling system that progressively transforms mid-air sketch strokes into patched 3D curve networks. Admittedly, the distinction between 3D ideation and concept models becomes subtle in VR when both are produced via a sketching tool such as CASSIE. In the subsequent text, we will use the following definitions:

– An *ideation* sketch serves as an externalization of a mentally evolving design. In our system, such sketches consist of independent strokes from which a surface cannot trivially be inferred.

– A *concept model* expresses the design as a well-defined shape that designers use to communicate their idea to others. It loses part of the ambiguity present in ideation, thereby gaining structure which enables refinement and reuse. In our system, concept models consist of connected curves which form a network that aids us in computing meaningful surface patches.

Inspired by interactive sketch beautification [77, 106], we automatically neaten strokes as they are drawn to detect and enforce intersections with nearby curves and 3D grid points, tangent continuity,

axis-alignment, and planarity—features that are common in man-made objects [267]. Uniformly applying multiple and potentially conflicting sketch constraints can cause neatened strokes to deviate from their design intent. We thus cast the selection of these criteria as selective 3D optimization, balancing geometric constraint satisfaction with fidelity to the sketched stroke. We further propose a real-time algorithm, inspired by Stanko et al. [226], to progressively construct surface patches across predicted cycles of network curves. The resulting surfaces serve a triple purpose: they incrementally bolster the 3D shape providing important depth and occlusion cues [16]; they serve as virtual canvases on which to draw or project additional detail curves; and they act as an ongoing incentive to draw consistent curve networks, naturally guiding the user from ambiguous ideation towards a well-defined 3D concept model.

While CASSIE embodies functionality to support both 3D ideation and conceptual modelling, we formally evaluate its salient features as three independent systems.

— *Freehand* mid-air drawing with minimal and independent smoothing of 3D user strokes.

— *Armature*, which enables overall stroke optimization to aid the creation of precise curve networks.

— *Patch*, which further enables the automatic detection and surfacing of appropriate curve network cycles.

We conducted a within-subjects study of the 3 systems with 12 users (design professionals or enthusiasts). The study validated the functional need for all aspects of CASSIE—*freehand* sketching for ideation, and *armature* and *patch* for concept modelling. Critically, the study showed that the complexity and constraints of our concept modelling functionality were achieved with an agency and overall user experience comparable to unconstrained ideation sketching. While our study data shows strong support for freehand ideative exploration (Figure 6.11), it equally validates our hypothesis that intermingled curve network and surface creation is valuable and encourages the creation of consistent 3D conceptual models.

Some of our professional study participants were keen to continue experimenting with CASSIE after the study, and we show a gallery of compelling and diverse 3D concept models created by a variety of users (Figure 6.14). We also demonstrate the suitability of these models for downstream sculpting, engineering, and fabrication applications (Figure 6.16).

This chapter thus makes the following three **contributions**.

— We present a homogeneous ideation and concept modelling system for design sketching in VR, based on a novel 3D curve and surface optimization framework.

— We report on a detailed comparative evaluation between ideation and conceptual modelling in VR, with 12 users.

— We provide a corpus of 3D stroke data for future design analysis and data-driven sketch processing.

6.1 User Interface and Workflow

We first present the drawing interface offered by our system and the workflow it enables. We describe the technical components necessary to achieve this workflow—stroke neatening and surfacing—in Sections 6.2 and 6.3 respectively.

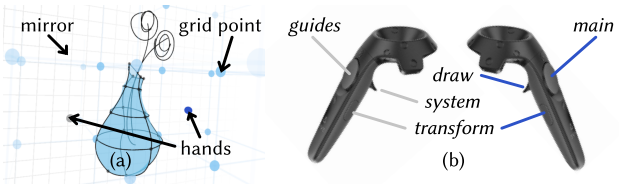


Figure 6.2: Our minimal interface is designed for simplicity, allowing the user to focus on creation (a). In a similar spirit, the controls are kept minimal and easily generalize to varied hardware (b). Here we show the controls mapped to an HTC Vive controller.

6.1.1 Interface Elements

We endeavoured to keep the interface minimal and user interactions simple, to keep the user focused on the creative task at hand. As a secondary objective, we only assumed standard interface features, common across modern VR setups, to ease remote evaluation on a variety of consumer VR hardware.

Figure 6.2a illustrates our drawing interface as seen in the VR headset. In addition to the 3D shape being created, our interface visualizes the ambient workspace with an axis-aligned 3D grid for better depth perception. The workspace also includes an optional mirror plane, which greatly facilitates the design of symmetric objects. Users interact with the system using two 6-DoF (degree of freedom) controllers, each of which must provide three push-buttons. Figure 6.2b shows all the controls on the HTC Vive controller, but the interactions can be similarly mapped to other common devices.

Users can draw strokes using the *draw* button. We represent each stroke as a smooth cubic poly-Bézier curve or a straight line segment, from which we remove unintended "hooks" at extremities as done by Liu et al. [148]. Double-clicking the *main* button on the dominant-hand controller deletes selected curves or surface patches. Selection is implicit, based on proximity to the curve/surface. Single-clicking the *main* button manually creates surface patches where automatic detection fails. Holding the *transform* button on the dominant or non-dominant controller allows uniformly scaling or rigidly transforming the workspace, respectively. Clicking or double-clicking the *guides* button on the non-dominant controller toggles the grid or the mirror plane, respectively. Finally, users can completely disable the automatic curve regularization and surfacing by clicking the *system* button, and use the same button to toggle the predictive features on again. Note that the functionalities attributed to the *draw* and *transform* buttons are very similar to commercial VR applications [4, 87], and therefore, users can quickly grasp their function.

Visible interface elements are similarly designed to be simple and unintrusive (see Figure 6.2a). Both the controllers are rendered as small spheres, differentiated by colour. The grid is rendered as semi-transparent lines and spheres. Curves are rendered in a visually-dominant black, while nodes of the curve network (intersections) are shown as faint spheres. The latter serves as an important visual confirmation of a successful connection between curves. Patches are shown as a faint blue to provide occlusion, thus improving depth perception (see Figure 3.13), without obstructing important design elements or breaking the creative flow. Finally, implicit selections are indicated by a colour change. We do not provide features to edit the strokes or surfaces. Once sketched, it is only possible to delete elements, once they are created.

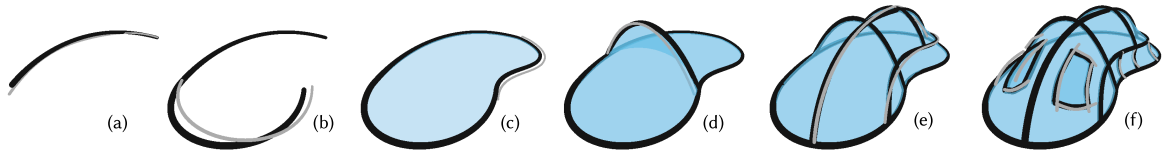


Figure 6.3: Typical workflow to create a 3D concept model with CASSIE. User-sketched strokes are shown in grey, and neatened strokes in **black**. Our system automatically connects user strokes as they are drawn to form long curves (a–c) and curve networks (d–e). We detect closed cycles in this network to form surface patches (c–e), which improves depth perception, supports the creation of surface details (f), and results in models ready for downstream applications.

6.1.2 User Workflow

Figure 6.3 illustrates a typical concept modelling session with our system. For every newly-drawn stroke, our system automatically identifies regularization criteria with respect to already-created 3D curves, as well as to the ambient 3D grid. In particular, the system aligns the new stroke, in position as well as tangent, to nearby curves to ease the creation of long curves (Figure 6.3a–c) and well-connected curve networks (Figure 6.3d–f). The system also snaps strokes to nearby grid nodes, facilitating the use of the grid as a precise scale reference.

For every new curve added to the network, our system searches for closed cycles in its vicinity to form surface patches or to modify existing ones (Figure 6.3c–e). These patches greatly contribute to the perception of the 3D shape by occluding background curves. They can also serve as a sketching support, on which users can draw surface details or connect extruding parts (Figure 6.3f). While the surface patches are found automatically, users can delete any patch to create holes, or add a patch where the automatic algorithm might have missed one due to ambiguity.

From the user perspective, the curve network and associated surface patches behave like a soap film, which the user shapes progressively by adding one curve at a time to form complete concept models (Figure 6.3f). Note that strokes on which we detect no constraint to apply are left unchanged. Alternatively, users can also toggle all the predictive features off to fall back to a freehand sketching workflow more suitable to preliminary ideation.

6.2 Creating Curve Networks

Designers often depict 3D shapes by sketching networks of curves running along surface boundaries and lines of curvature [65, 90]. These curves aid the creation of 3D meshes in downstream 3D modelling [231], and several algorithms exist to automatically surface 3D curve networks [32, 201, 275]. Unfortunately, unlike 2D sketching, 3D user strokes rarely intersect perfectly, making freehand drawing of 3D curve networks in VR difficult. We facilitate the creation of such curve networks by automatically detecting and enforcing proximal curve intersections. However, complex drawings contain cluttered regions, where enforcing all such candidate intersections for a single stroke can distort the stroke dramatically. Inspired by related work on 2D sketch beautification [77, 106] and sketch-based modelling [209, 267], we propose a mechanism to connect each new 3D stroke to as many nearby 3D curves as possible (discrete hard constraints) while staying close to its original trajectory, and compensating for drawing inaccuracy by favouring geometric features—planarity of curves and tangent continuity (continuous soft constraints). Solving such a problem is a mix of a potentially exponential search for the optimal subset of discrete

hard constraints to satisfy, and function minimization for the continuous soft constraints.

6.2.1 Algorithm Overview

Our solution has three steps. Given a sketched 3D stroke, we first find all candidate hard intersection constraints using the distance between points on the stroke and existing 3D curves. We then find an approximately optimal subset of discrete constraints using a greedy linear search, and formulate a least squares minimization for the continuous constraints. Our overall algorithm is thus efficient, robust, and works well in practice without any noticeable lag.

We next describe our algorithm in a bottom-up fashion. We first detail our 3D curve optimization that minimizes soft constraints and remains as close as possible to the input stroke, while satisfying a prescribed subset of hard constraints (Section 6.2.2). We then describe our strategy to select this optimal subset (Section 6.2.3) We then detail how the various terms of the optimization are formulated (Section 6.2.4), and describe how the optimization problem is solve in Section 6.2.5.

6.2.2 Enforcing Intersections via Continuous Optimization

Let us first assume that we have selected a set of intersections $\mathcal{I}$ between the user stroke and existing curves in the 3D sketch. We associate each intersection with a constraint $c_{i\in\mathcal{I}}$ that needs to be satisfied for the user stroke to intersect the corresponding curve exactly. However, the input stroke needs to be deformed to satisfy these constraints. We measure the amount of deformation with an energy that we denote E_{fidelity}. In addition, we also seek to encourage regularization criteria such as curve planarity and tangential alignment to intersected curves, which we also express as energy terms E_{planar} and E_{tangent}. We express these criteria as soft energy terms rather than hard constraints because they correspond to aesthetic properties that are desirable but that can be balanced against other desiderata. In contrast, we need intersections to be satisfied exactly to form a well-connected curve network. Finally, we also ensure that the deformed stroke preserves the original G^1 continuity between successive curve segments, which we express as a set of constraints g_k. Given an input stroke S, we compute the deformed stroke S' that satisfies all constraints while minimizing deformation and best satisfying other criteria by solving the *continuous* optimization

$$\begin{aligned} \underset{S'}{\text{minimize}} \quad & E_{\text{fidelity}}(S, S') + E_{\text{planar}}(S') + \sum_{i\in\mathcal{I}} E_{\text{tangent}}(S', i) \\ \text{subject to} \quad & c_{i\in\mathcal{I}}(S') = 0, \\ & g_k(S') = 0. \end{aligned} \tag{6.1}$$

Recall that our strokes are represented using cubic Bézier segments. We formulate the soft constraints as quadratic energy functions of the Bézier control points, and hard constraints as linear functions, allowing us to solve this constrained least squares problem as a single linear solve as detailed in Section 6.2.5.

6.2.3 Selecting Intersections via Discrete Optimization

New strokes often run near multiple curves, especially on complex drawings with high stroke density. Our next challenge is to identify which of these curves should intersect the input stroke. On one hand, we would like the curve network to be as connected as possible. On the other hand, we don't want to deviate too much from the original user intent. We cast these competing objectives as two terms in a

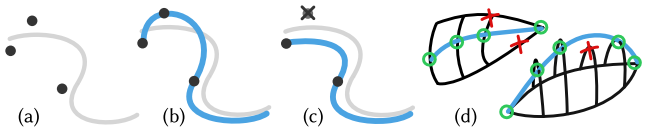


Figure 6.4: An input stroke with all detected potential intersections (a). Enforcing all intersections distorts the curve (b). Enforcing the optimal subset keeps the curve close to the input (c). Typical configurations where too many intersections are detected (d): in areas with high stroke density (top) or when the input stroke is smooth but the candidate intersections do not form a smooth path (bottom). Our algorithm selects a subset of intersections (**circled**) and rejects others (**crossed out**) to preserve the shape of the original curve (**blue**).

discrete optimization, where we model the activation of each constraint c_i with a binary variable b_i set to 1 if the intersection is selected and to 0 otherwise. The optimization problem is

$$\underset{i \in \mathcal{I}}{\text{minimize}} \quad \lambda E_{\text{fidelity}}\big(S, S'(b)\big) + (1-\lambda) E_{\text{connectivity}}(b) \tag{6.2}$$

where the deformed stroke $S'(b)$ is computed using Equation 6.1 and the selected intersections, E_{fidelity} measures the deviation of S' from the input stroke, $E_{\text{connectivity}}$ is an energy term that decreases as more intersections are selected, and $\lambda = 0.6$ is a parameter that balances the two objectives.

Unfortunately, finding the optimal subset would require solving Equation 6.1 for all $2^{|\mathcal{I}|}$ combinations of binary variables b_i, which is impractical. Yet, we observed that the detected intersections are most often valid, and only a few superfluous ones need to be rejected. This observation motivated us to minimize Equation 6.2 using a greedy strategy, where we first select all intersections, and then test all subsets obtained by removing one intersection. If the best of these subsets yields a lower energy than the full set, we keep it and repeat the process by removing another intersection, until the energy doesn't decrease anymore. Figure 6.4 illustrates the result of this selection mechanism on a 2D curve, and on typical 3D curve configurations. The number of subsets that must be tested for each stroke is determined by the total number of detected constraints $|\mathcal{I}|$, and the number of rejected constraints $|\mathcal{I}_r|$ as: $1 + (|\mathcal{I}_r| + 1)(2|\mathcal{I}| - |\mathcal{I}_r|)/2$.

The efficiency of our greedy strategy was validated during a subsequent user study (see Section 6.4), where we counted an average of 5 subsets tested for each stroke, with a median of only 2 subsets. In practice, even this count is an upper bound as we further prune the number of subsets tested when some constraints are deemed incompatible, such as when a point is constrained to multiple positions.

6.2.4 Constraints and Energy Terms

We now describe how we detect and express the constraints and energy terms that form Equation 6.1 and Equation 6.2.

Interactive neatening. Since we target an interactive sketching application, our method optimizes each stroke as soon as it is sketched, in the context of previously-drawn curves. We do not revisit the strokes that have already been optimized, as this would turn neatening into a global optimization problem that would become increasingly costly as more strokes are added. As a curve is sketched, it is first fit to a poly-Bézier curve $B(t)_{t\in[0,1]}$ using the method described by Miller [163]. The control points of the i^{th} cubic Bézier segment are denoted as $P_0^i, P_1^i, P_2^i, P_3^i$, in order.

Distance and angular thresholds. Many of our regularization constraints and objectives depend on a notion of proximity between curves, in space and/or angle. In addition, we wish to make the sensitivity of our beautification process adaptive to the drawing scale, such that the beautification is less aggressive when users zoom-in to draw details. We achieve this goal by defining a spatial proximity threshold $\delta(s) = \frac{\delta^1}{s}$, where s denotes the scale of the drawing space and $\delta^1 = 4$cm was fixed experimentally for appropriate beautification. In the initial zoomed out state, $s = 1$.

We set the angular threshold to $\theta = \frac{\pi}{6}$, under which we consider that two lines or curves are parallel.

Intersections constraint. We detect intersections between the input stroke and any nearby curve, grid nodes, or the mirror plane by testing if the brush tip comes within a radius $r_{\text{proximity}} = \delta(s)$ of such elements while sketching.

To constrain the poly-Bézier curve $B(t)_{t\in[0,1]}$ to intersect a point p_{target}, we first compute the closest point on the curve from p_{target}

$$p^* = B(t^*); \quad \text{where } t^* = \arg\min_t \|B(t) - p_{\text{target}}\|. \tag{6.3}$$

Here, p_{target} may be a grid point, or the closest point to $B(t)$ on the mirror plane or an existing curve. If we are close to an existing control point P_0^κ, such that: $\|P_0^\kappa - p^*\| < \delta(s)/2$, we constrain that control point. Otherwise, we split the input curve at t^* using de Casteljau's algorithm. This yields a new control point on the curve $P_0^\kappa = p^*$. In both cases, we express the hard constraint c as

$$c = P_0^\kappa - p_{\text{target}} = 0. \tag{6.4}$$

We rely on the fidelity energy (described later in this section) and our greedy constraint selection strategy to reject constraints that would deform the input too much, and to select which constraint to apply at a control point if there are more than one.

Closed curve constraint. We express the constraint that enables the creation of closed loops the same way as an intersection constraint, by forcing the endpoints of the curve to coincide. If P_0^0 is the first control point and P_3^{K-1} the last (K being the total number of cubic Bézier in the poly-Bézier curve), the constraint is

$$c = P_0^0 - P_3^{K-1} = 0. \tag{6.5}$$

G^1 continuity constraint. We want $B(t)_{t\in[0,1]}$ to have G^1 continuity even after the beautification process. That is, we want to satisfy some control point equality and tangent alignment between successive Bézier segments of the curve.

$$g = P_3^{k-1} - P_0^k = 0\ , \quad \text{for } k \in \{1, \dots, K-1\}, \text{ and} \tag{6.6}$$

$$g = \frac{(P_3^{k-1} - P_2^{k-1})}{\|P_3^{k-1} - P_2^{k-1}\|} - \frac{(P_1^k - P_0^k)}{\|P_1^k - P_0^k\|} = 0\ , \quad \text{for } k \in \{1, \dots, K-1\}. \tag{6.7}$$

For the C^0 constraint (Equation 6.6), we simply remove all control points variables P_0^k, for $k \in \{1, \dots, K-1\}$ from the optimization.

For the G^1 constraint (Equation 6.7), we linearize this constraint by approximating the norms of the control polygon edges by the initial norms.

$$\|P_i^k - P_{i-1}^k\| \approx \|\bar{P}_i^k - \bar{P}_{i-1}^k\|. \tag{6.8}$$

Making this constraint linear enables us to solve the optimization efficiently and gives reasonable results in our use case.

Tangent alignment constraint. At each intersection, we compare the tangents between the two curves. If the angle between both tangents is under the angular threshold θ, we encourage the tangent of the new curve to align with the other (say T_{target}) using an energy term E_{tangent} that measures the norm of their cross product.

$$E_{\text{tangent}} = \|(P_1^k - P_0^k) \times T_{\text{target}}\|^2. \tag{6.9}$$

Otherwise, we set E_{tangent} for the curve to zero.

Planarity constraint. We first compute the best-fit plane to the control points using linear least squares. If the distance from the plane to the farthest control point is below the threshold $r_{\text{proximity}}$, we encourage all control points to lie in the plane using an energy E_{planar} that minimizes the dot product between the plane normal $\vec{n}$ and the vectors formed by pairs of successive control points.

$$E_{\text{planar}} = \sum_{i\in\{0,1,2\},\ k\in[0,K-1]} \|(P_{i+1}^k - P_i^k) \cdot \vec{n}\|^2. \tag{6.10}$$

We also test whether the plane normal is within an angle θ to one of the three orthogonal grid directions, in which case we first snap the plane to be corresponding coordinate plane.

For non-planar curves, we set E_{planar} to zero.

Fidelity to input. Our fidelity energy E_{fidelity} measures deviation between the beautified curve and the input one. Our formulation of the fidelity energy extends the *projection accuracy* energy described by Xu et al. [267] to compare 3D curves instead of 2D curves.

We want to minimize both variation in absolute position of the control points P_i^k from the input positions $\bar{P}_i^k$ and variation in the Bézier polygon edges $e_{ik} = P_{i+1}^k - P_i^k$ (see Figure 6.5). This leads us to define the *fidelity energy* as

$$E_{\text{fidelity}} = \frac{1}{|P_i^k|\delta^2} \sum_{i,k} \|P_i^k - \bar{P}_i^k\|^2 + \frac{1}{|e_{ik}|} \sum_{i,k} \frac{\|(P_{i+1}^k - P_i^k) - (\bar{P}_{i+1}^k - \bar{P}_i^k)\|^2}{\|\bar{P}_{i+1}^k - \bar{P}_i^k\|^2}. \tag{6.11}$$

We normalize each term by, respectively, the total number of control points $|P_i^k| = 3K + 1$ and the total number of Bézier control polygon edges $|e_{ik}| = 3K$, K being the number of cubic Bézier segments. As our Bézier curves are all arbitrarily subdivided, depending on both input stroke curvature variations and the number of intersection constraints applied, we normalize these terms so that we can later compare fidelity energy between different candidate results. To control the scale of the energy, we normalize the first term by the proximity threshold δ and the second term by the initial lengths of the

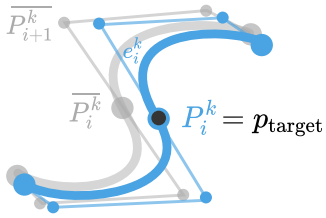


Figure 6.5: Input stroke (**gray**) and optimized stroke (**blue**) to match the intersection constraint p_{target}.

control polygon edges $\bar{P}_{i+1}^k - \bar{P}_i^k$. This also accounts for the uneven lengths of the Bézier control polygon edges, allowing greater deviation for longer edges, as in Xu et al. [267].

Connectivity. The goal of $E_{\text{connectivity}}$ is to favour the selection of as many intersections as possible, as long as they do not overly deform the input. We model it with an exponential that increases as fewer constraints are selected

$$E_{\text{connectivity}}(b) = \exp\left(-\left(\frac{\sum_{i\in\mathcal{I}} b_i w_i}{\sum_{i\in\mathcal{I}} w_i}\right)^2\right), \tag{6.12}$$

where w_i denotes the weight associated to intersection i, and b_i equals 1 if the intersection is selected, 0 otherwise. Following Schmidt et al. [209], we vary this weight depending on the nature of the intersection, being equal to 1 if the curve intersects a grid node or the mirror plane along its trajectory, 1.25 if the curve intersects a grid node at its endpoint, 1.5 if the curve intersects another curve, and 2 if the curve passes through an existing intersection.

Sharp features. We support strokes with sharp features by representing them with poly-Bézier curves without G^1 continuity at the junctions between successive Bézier segments. For such junctions, we do not enforce the hard constraint g_k from Equation 6.1. We empirically determine that an input stroke intends to represent a sharp junction if we detect a strong variation of tangent directions ($> 45°$, as in Rosales et al. [197]).

6.2.5 Solving the Optimization Problem

Our complete optimization problem that solves for the control points position variables $\{P_i^k\}$, $i \in [0, 3]$, $k \in [0, K-1]$, while applying the m_h hard constraints and the m_s soft constraints is

$$\underset{\{P_i^k\}}{\text{minimize}} \quad E_{\text{fidelity}}(\{P_i^k\}) + \sum_{j\in\{0,\dots,m_s-1\}} E_j(\{P_i^k\}) \tag{6.13a}$$

$$\text{subject to} \quad c_l(\{P_i^k\}) = 0\ , \ \text{for } l \in \{0, \dots, m_h - 1\}. \tag{6.13b}$$

We define P as a column vector of size $3(3K+1)$, corresponding to the control points position variables, with one entry per control point coordinate.

For each hard constraint $l \in \{0, \dots, m_h - 1\}$, we have n linear equations, such that we can write the constraint as: $c_l = C_l P = 0$, C_l being an $n \times 3(3K+1)$ matrix.

The objective function has a linear gradient, and all hard constraints are linear so we can find the solution to this optimization problem by solving a linear system (Equation 6.14), using the Lagrange multipliers method [166]:

$$\begin{bmatrix} A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} P \\ \Lambda \end{bmatrix} = \begin{bmatrix} B \\ 0 \end{bmatrix}. \tag{6.14}$$

We build A and B such that $AP - B = \frac{\partial}{\partial P}(E_{\text{fidelity}} + \sum E_j)$. The matrix C corresponds to the vertically stacked C_l blocks of the hard constraints. Λ is the vector of Lagrange multipliers.

For the linear solve, we use an LU factorization implemented by the MathNET Numerics library [198]. On Intel machines, we use the Math Kernel Library [54] as our BLAS backend to speed up the linear algebra computations.

6.2.6 Special Case: Beautification of Straight Lines

We employ a simpler algorithm to beautify straight strokes.

Selecting the Constraints. Since a line is uniquely defined by two points, we cannot apply more than two intersection constraints along a line segment. We use a simple heuristic to select the best two constraints: we don't expect users to draw a stroke longer than it needs to be. Following this assumption, we select the two constraints that are closest to the stroke endpoints. In addition, we use the angular threshold θ to determine whether a line segment is close enough to one of the three world axes to be constrained to align with it.

Applying the Constraints. To constrain a line segment to one or two intersection constraints, we simply update one or both endpoints such that the line segment passes through the constraint(s). If the constraint is close enough to one of the endpoints (within a distance δ), we set the endpoint to be at the constraint position. If there are two intersection constraints and they are not near the endpoints, we project both endpoints on the line formed by the intersection constraints. Finally, if there is one intersection constraint that is not near one of the endpoints, we translate the line segment so that it passes through the constraint.

6.3 Surfacing

Our stroke neatening and structuring algorithm greatly facilitates the creation of well-connected curve networks (Figure 6.6), which in turn enables the automatic detection of curve cycles (Figure 6.6b) bounding surface patches (Figure 6.6a). We first describe how we identify these cycles before detailing how we generate the surface they bound and how we exploit these surface patches for drawing on-surface details.

6.3.1 Cycle Detection

A number of existing cycle detection algorithms perform a global optimization over the complete curve network [1, 275]. Unfortunately, the space of cycles of a given curve network is very large, making a global search computationally demanding and unnecessarily repetitive in an interactive system. Instead,

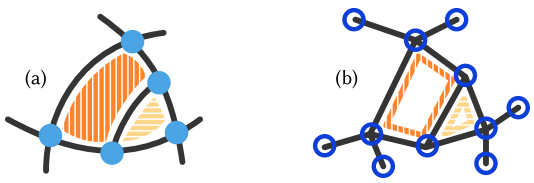


Figure 6.6: After enforcing intersections (Section 6.2), we obtain a result such as in (a). The sketch is composed of curves that meet at **intersections**. The sparse strokes represent surface patches (bright and light **orange**) that we create automatically. To do so, we reason on a curve network representation of the sketch (b) where each curve is represented by one or multiple segments connected at **nodes**. On this curve network we search for cycles corresponding to the patches.

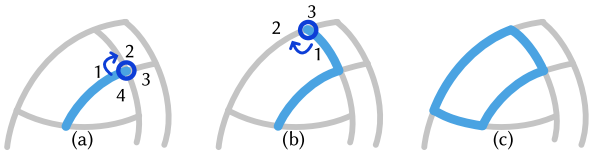


Figure 6.7: Local cycle detection. We sort all segments incident to an intersection according to the surface normal at that intersection (a). We walk around the cycle by selecting at each node the next segment in clock-wise order (b), until we obtain a closed cycle (c).

we exploit the fact that after the user has sketched a new stroke, any new patches should only be created for cycles in parts of the network affected by the stroke. We thus apply a local search strategy, which is more suitable for real-time, incremental construction of the sketch. Our approach is inspired by the algorithm of Stanko et al. [226], which relies on the surface normal at each intersection to sort all segments incident to that intersection, such that cycles can be formed by walking around the curve network in a fixed order as illustrated in Figure 6.7.

However, while Stanko et al. [226] were reconstructing real-world surfaces for which the normal was measured, we need to infer the surface normal from the input curves. We obtain this normal estimate at each intersection by assuming that the intersecting curves lie on a smooth surface, whose tangent plane is spanned by the curve tangents. Yet, this tangent plane only defines the normal direction, not its orientation. We remove this sign ambiguity at the scale of a cycle by comparing normal orientations between successive nodes. Along a curve, we can compute a family of vectors by parallel-transporting the normal from one of its nodes [94]. These vectors are a good approximation of the normals of the underlying surface if the curve lies on this surface without significant geodesic torsion, which is the case for curvature lines that form a large part of designer-drawn curves [105, 176, 214].

Based on this observation, parallel-transporting the normal $\vec{N}_A$ from node A to node B gives a

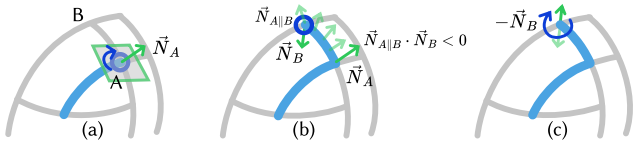


Figure 6.8: Local normal orientation. We compute the surface normal $\vec{N}_A$ at node A by fitting a plane to the tangents of incident curve segments (a). We parallel-transport this normal to node B to obtain $\vec{N}_{A\|B}$, which we compare to $\vec{N}_B$ (b). Since the two vectors point to opposite directions, we flip $\vec{N}_B$ before proceeding (c).

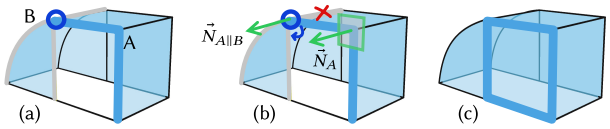


Figure 6.9: Dealing with sharp surface features. The surface normal is ill-defined at node B, as one of the incident segments does not lie in the plane formed by the other ones (a). We exclude the segment that is not in the plane defined by the parallel-transported normal $\vec{N}_{A\|B}$ (b, **crossed out**) and select the next segment in clockwise order around $\vec{N}_{A\|B}$ among the remaining options to form the cycle (c). In cases where the starting node A is also ill-defined, we compute the surface normal $\vec{N}_A$ as the cross-product of two successive segments (b, **blue**) among the segments at A sorted according to the normal of the best-fit tangent plane.

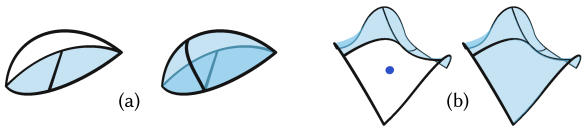


Figure 6.10: The automatic cycle detection might fail. Adding strokes resolves the issue by creating additional intersections (a). It is also possible to manually add a patch by clicking (b).

vector $\vec{N}_{A\|B}$ that should closely match the direction of normal $\vec{N}_B$. If the two vectors point in opposite directions, we flip $\vec{N}_B$ before proceeding with the next node along the cycle, as illustrated in Figure 6.8.

We apply this cycle detection on both sides of each new curve segment, taking care of removing any existing cycle when a curve is drawn across it. We also run the cycle detection on every segment that intersects user-deleted curves to update the surface after such edits.

While simple and efficient, the above algorithm is limited to smooth surfaces for which the tangent plane is well-defined at each curve intersection. If a node lies on a sharp surface feature, such as the corner of a cube, then several surface normals exist at that node (Figure 6.9). We detect such cases by checking whether some curve tangent at this node lies far from the best fit tangent plane, by assessing the dot products of all the incident curve tangents $\{\vec{t}_i\}_i$ with the plane normal $\vec{n}_P$. We consider that a node is on a sharp feature if $\max_i \vec{n}_P \cdot \vec{t}_i > \cos\left(\frac{\pi}{2} - \theta\right)$, with θ the angular threshold defined in Section 6.2.4.

On encountering such a node, we rely on the last well-defined normal along the cycle, which we transport to the node to sort its segments. We also take care of excluding the segments that do not lie close to the plane defined by this normal. In the rare case where the starting node of a cycle is itself sharp, we first resort to the normal of the best-fit plane to sort the segments incident to that node, and then for each successive pair of segments we look for a cycle with starting normal defined as the cross product between these two segments.

Nevertheless, our automatic algorithm sometimes fails to detect cycles, especially on sparse curve networks as in Figure 6.10a. Furthermore, our assumption that the curves exhibit little geodesic torsion does not always hold.

While adding new curves to form a denser network often resolves these issues, we also allow users to explicitly trigger the creation of a surface patch by clicking near the centre of the patch they envision (Figure 6.10b). We then detect the closest stroke segment to the click and walk along the curve network by always selecting the segment that is closest to the click, as measured by comparing distances from the click to the closest point on each segment. We also allow users to delete unwanted surface patches, for instance to create holes.

6.3.2 Surface Geometry

Once a cycle is detected, we generate a triangle mesh over it with the method by Zou et al. [276], using their public implementation. We then refine the generated surface by applying isotropic re-meshing followed by curvature-flow smoothing [59, 124] to generate minimal (soap-film) surfaces. Both steps are performed using CGAL [236].

6.3.3 Sketching on Surfaces

The surface patches generated by our method not only enhance the perception of the created shape (see Section 3.4.3), they can also serve the role of a canvas on which users can draw additional details. We consider that a stroke should lie on a surface patch if a) all of its control points are within a distance $r_{\text{proximity}}$ from that patch; and b) the curve does not split the cycle bounding the patch. When these criteria are fulfilled, we project all control points of the curve onto the surface patch so that the user stroke appears to stick to the object. Advanced techniques such as the *mimicry* technique described in Chapter 5 could also be implemented in the future.

6.4 User Experience Study

We formally evaluated the novel features of CASSIE (*armature*, *patch*), using *freehand* as a baseline, by re-configuring CASSIE as three isolated sub-systems with a common interface. Our study goals were twofold. One, evaluating whether the user experience achieved with stroke neatening and surfacing is comparable to free-form sketching, making CASSIE a homogeneous sketching interface capable of supporting ideation as well as more refined concept modelling. Two, evaluating the effectiveness of our algorithms in neatening the user strokes and creating well-connected models. Specifically, our study participants tried and evaluated three versions of the tool: a baseline *freehand* version which only contained mid-air sketching features, an *armature* version which implemented our curve neatening and structuring featureset, and a *patch* version which also included the automatic and manual surfacing features, and the ability to draw on surface patches. We performed this study with expert and novice users who are all familiar with VR interfaces in general.

In addition to this formal study, in Section 6.5.1 we report on an informal comparison of CASSIE vis-à-vis Gravity Sketch [91].

6.4.1 Participants

We recruited 12 participants (10 male, 2 female) aged 25–50 for the study. Six of the participants were professional artists or designers, while the others had at least basic experience in CAD modelling or creating design sketches as part of their jobs in graphics, HCI, or related fields and as hobbyists. All but two had used VR for over a year, and most (9/12) had utilized VR for creative tasks, using Tilt Brush [87], Medium [4], Masterpiece Studio [154], Blocks [86], AnimVR [167], or research prototypes. Participants were paid approximately CA$ 30 for their time, converted to their currency of choice.

6.4.2 Apparatus

Owing to the restrictions on in-person user studies due to the COVID-19 pandemic, the study sessions were conducted remotely, using participants' own VR setups. As a result, participants used a variety of VR devices, including Oculus Rift and Rift S, HTC Vive, Vive Pro, and Vive Pro Eye. During the study session, an experimenter was available via video-conferencing for answering participant questions.

6.4.3 Procedure

Participants first used the three systems to draw a computer mouse. These untimed sessions were meant to be interactive tutorials, intended to let the participants familiarize themselves with the interface. Then, the participants used the three systems for timed sessions, in the same order as the tutorial. The timed sessions which were later utilized for qualitative and quantitative analyses. For each system, participants were given 5 minutes to design a desk lamp followed by another 5 minutes to create a running shoe. To reduce the influence of learning effects, the order of the systems was counterbalanced between participants using a Latin square design. It is important to note that participants were not constrained to draw the same shape with all the three systems. Instead, we asked them to work with the capabilities and constraints of each system to design the optimal object.

Each participant was sent a set of instructions a day before their study slot, informing them of the nature of the research, the objects they were expected to draw, and the expected drawing style of a sparse curve network. We hoped that this preparatory material helped participants think ahead and plan their designs in advance, thus focusing on concept modelling in addition to pure ideation. However, participants were not required to do so.

The study session started with participants signing an informed consent form, followed by them watching a detailed instructions video (11 min), and filling out a demographics questionnaire. After evaluating each system, participants took a break and filled a short questionnaire for that system. Finally, participants filled a more detailed questionnaire comparing the three systems.

6.4.4 Results

Figure 6.11 shows the 3D sketches and models created by the six participants who declared themselves to be professional artists. Results for the six amateur participants are shown in Figure 6.12.

Most participants created shapes of similar complexity with all three systems (65 strokes on average for *freehand*, 44 for *armature*, 42 for *patch*, with no statistically significant difference between the three), although some participants took advantage of the *freehand* system to achieve a more sketchy style with significant over-sketching (P5). Under close inspection, most *freehand* sketches exhibit over-shooting and gaps at stroke ends, which are mostly absent in the drawings created with the *armature* and *patch* systems. Finally, several of the models created with the *patch* system contain spurious holes in the 3D surface, which are either due to disconnected curves in the sketched network, or to failure of the automatic cycle detection algorithm.

Figure 6.13 shows two drawings created in the *armature* system next to the raw strokes before snapping, which illustrates the approximate user inputs and the net effect of stroke neatening, structuring, and snapping.

We next discuss participants' feedback about their experience with the three systems, and then perform quantitative analysis of the curve networks they created.

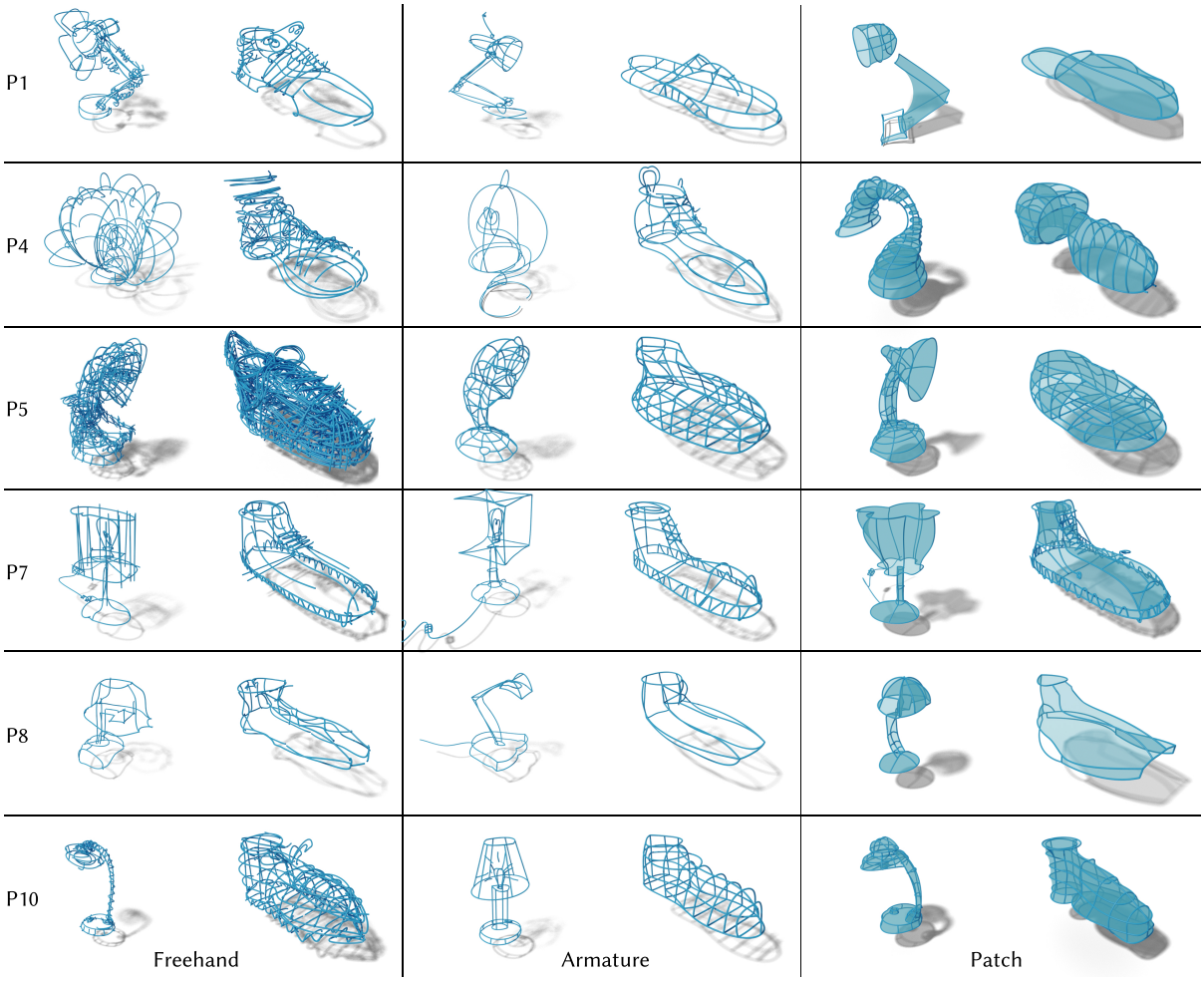


Figure 6.11: Designs created by the six professional participants in the study using all three systems.

User Feedback

Users evaluated the degree of creative support provided by each system via the Creative Support Index (CSI) questionnaire [46]. We skipped the collaboration part of the questionnaire, as we did not build any collaborative tools in our systems. The summary results are shown in Table 6.1. The three systems fare almost evenly, most differences being statistically insignificant ($p > .05$ on a repeated-measures ANOVA), except for *Exploration* where the *freehand* system outperforms the two others.

Participants commented that each system offered support for a slightly different portion of the *ideation–concept design continuum*: "Each has its own uses. Freehand is good for quickly sketching up an idea as you are not restricted by the system. Armature is good for refining a design..." *(P11)* , "Armature seems [to be] a way to express volumes for more technical goals, but sketching [freehand] expresses an idea." *(P1)*

Some desired the ability to quickly switch back-and-forth between the systems, a feature we implemented after the study: "[*Armature*] handles noise coming from the input motion and signal and creates a much nicer looking stroke. It would be nice to be able to turn off snapping however, to allow strokes to be drawn closer together." *(P3)* In particular, automatic curve neatening reaches its limits on detailed sketches: "...predictive neatening was useful to help me quickly block out the initial shape,

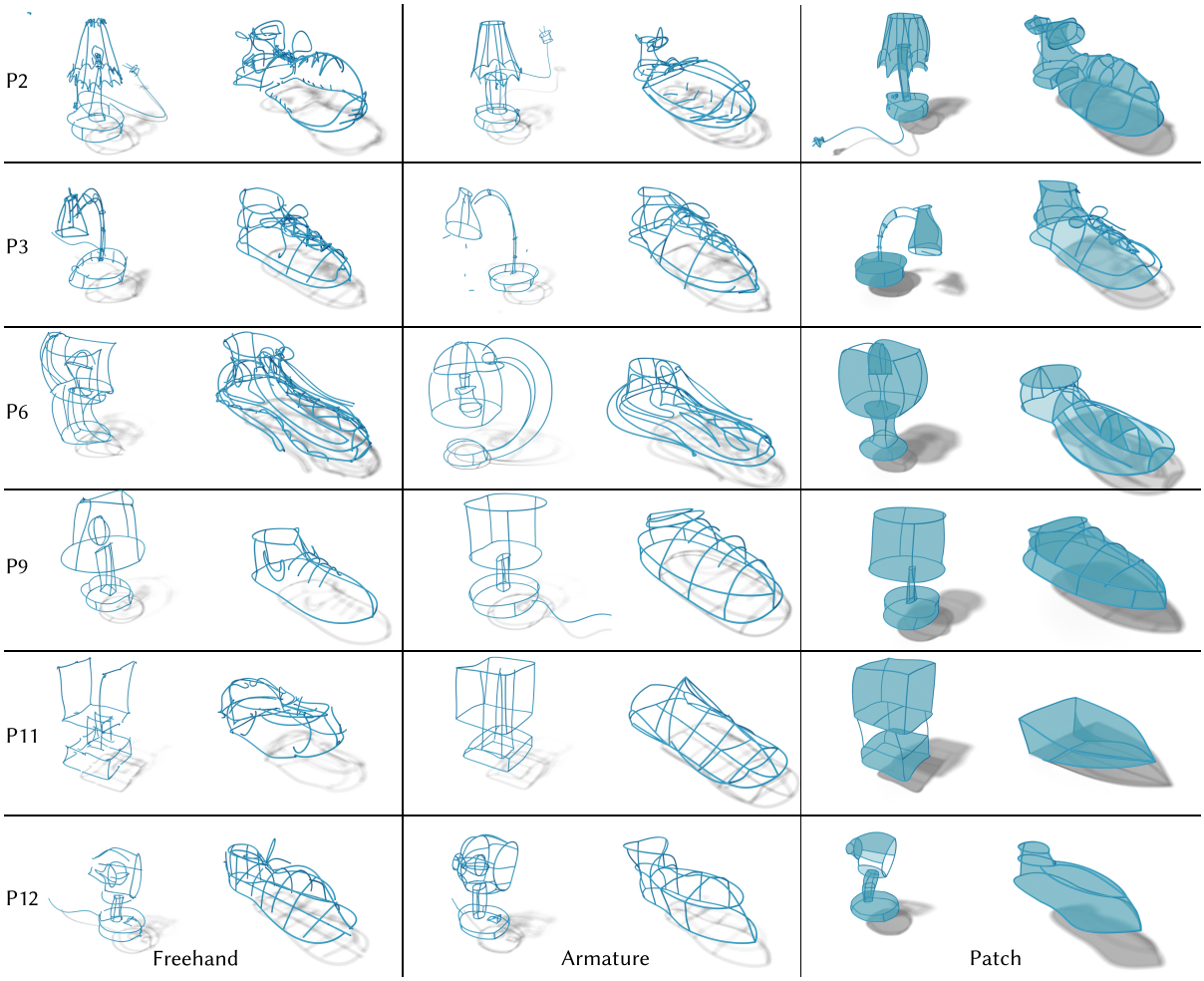


Figure 6.12: Designs created by the six other participants in the study using all three systems.

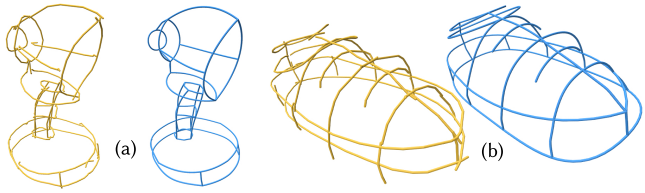


Figure 6.13: Input strokes (yellow) and beautified result (blue) for sketches by P12 (a) and P9 (b). Notice how our beautification creates well-connected curve networks while deviating minimally from the original user strokes. Images rendered using Polyscope [215].

and snapping to form continuous lines was helpful since I was able to break up a complex stroke into smaller ones. But when the sketch became more complex and I had to draw finer/smaller strokes, it often "over-neatened" my strokes." *(P7)*

Still, to compare the three systems, participants were asked to rate the value of the *armature* features compared to *freehand*, and the value of the *patch* system compared to *armature* on 5-point Likert scales (5 is highest). Most participants agreed that *armature* features added to the *freehand* system (6×5, 6×3, **median 4**) and the *patch* features were a valuable addition over *armature* (5×5, 4×4, 2×3, 1×2,

Table 6.1: Creative Support Index (CSI) [46] scores (out of 20, higher is better) for each of the 3 systems (mean and standard deviation). Statistically significant differences are marked with asterisks (*). These scores put *armature* and *patch* on par with the *freehand* system, showing that the additional mental load imposed by these two systems did not cause a noticeable drop in users' creative potential.

	Freehand	**Armature**	**Patch**
Enjoyment	16.8 (2.7)	15.5 (2.4)	16.2 (2.7)
Exploration	15.0* (4.3)	13.1* (4.4)	13.5 (4.3)
Expressiveness	16.8 (2.9)	14.6 (3.2)	14.6 (3.6)
Immersion	16.6 (2.4)	15.5 (2.3)	15.2 (3.5)
Results Worth Effort	15.1 (3.5)	16.1 (1.9)	15.6 (3.0)
CSI	16.1	15.0	15.0

median 4). Participants felt that the curve network *armatures* encouraged them to think more about the object's use in downstream processes: "...with armature you can see the beautified sketch closer to a final draft." *(P8)* , "In the freehand method, I find I'm more inclined to do a messy sketch model rather than a more thought out one when using the armature system." *(P4)*

The surface *patches* further bolstered this idea, motivating the creation of solid, usable, real-world shapes: "The patch system let [sic] you understand the geometry to create solid object necessary for many projects in an organic way. As for armature you can't see this right away taking an extra step in the process." *(P8)* , "The patch system visually assisted my understanding of the model, [it] seemed to give me a better sense of its volume." *(P9)* Similarly, P2 commented that *patch* "allows users to define [a] surface without many lines which can clutter a design and possibly de-emphasize important features that need line defininition [sic]."

However, the incentive to create well-connected curve networks in the *patch* system was also experienced as a constraint by some: "I found that using the patch system actually distracted me with trying to get the patching correct over the act of just sketching out what I was thinking about." *(P4)* , "I felt that I had to change the way I worked to accommodate the system. For example, I would have to add more strokes to get the patches to show up, which meant that I deviate from my initial intent, or create a messier/busier looking drawing than I would have liked." *(P7)*

Nevertheless, all the participants felt that they could derive value by incorporating the *armature* and *patch* functionalities in their typical workflow. For instance, P10 remarked that the systems were ideal for conceptualization: "Could be a great way to define rough shapes before bringing them into a desktop app to refine."

Quantitative Analysis

Our goal when designing CASSIE was to support users in creating well-connected curve networks that can be surfaced for downstream applications. While visual comparison between *freehand* sketches and *armatures* show that the latter are much more connected, we hypothesize that the *patch* system provides additional incentive to create connected networks by rewarding participants when they do so. We evaluate this hypothesis by computing, for both *armature* and *patch* curve networks, the ratio of the number of dangling (degree one) nodes to all the nodes. We found that the curve networks created with *patch* contain fewer endpoint nodes ($\mu = 9.8\%,\ \sigma = 6.8\%$) compared to armature ($\mu = 14.9\%,\ \sigma = 11.4\%$), meaning that they are better connected. The difference is statistically significant ($t = 2.50,\ p =$

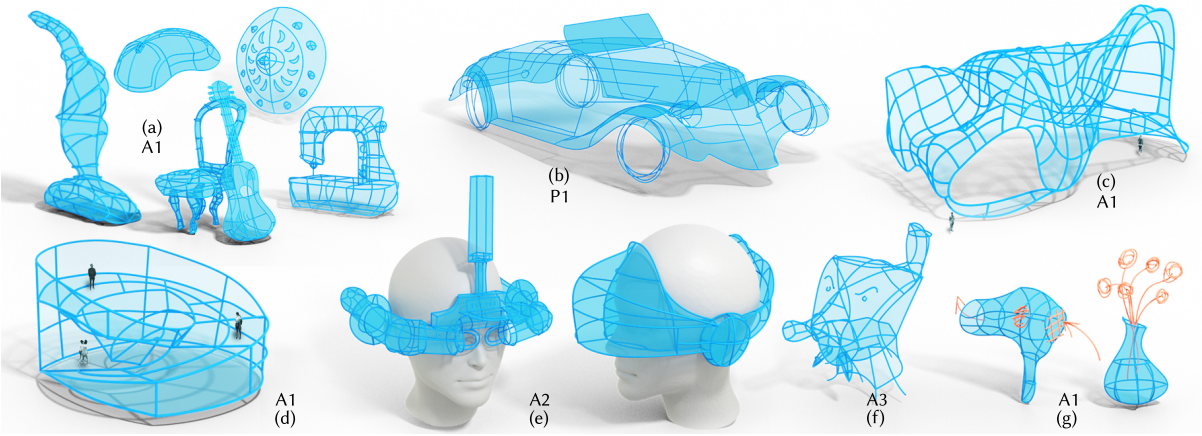


Figure 6.14: A gallery of 3D sketches created with CASSIE, labelled with the creator (A: author, P: participant). While most designs made use of the *armature* and *patch* features, some also included freehand strokes (**orange**) to suggest less definite parts (h).

.020 on a paired t-test). The variance of this metric is also lower with the *patch* system, suggesting that more participants tended to sketch well-connected networks with *patch*.

While the *armature* and *patch* systems attempt to detect curve intersections automatically, they might sometimes miss intended intersections, or create unintended ones. When this is the case, users typically delete the stroke and redraw a new version that better reflects their goal. In contrast, users of the *freehand* system might correct for erroneous strokes simply by over-sketching duplicate strokes. We quantify these different strategies by counting the ratio of deleted strokes for each system: *freehand* ($\mu = 27.2\%$, $\sigma = 14.1\%$), *armature* ($\mu = 36.3\%$, $\sigma = 12.5\%$), *patch* ($\mu = 39.5\%$, $\sigma = 13.3\%$). A repeated-measures ANOVA reveals that users deleted significantly less strokes in the *freehand* system ($F_{2,22} = 6.93, p = .005$).

6.5 Results

Figure 6.14 provides a gallery of models created with CASSIE by several trained users, including a professional study participant (P1) and 3 different authors of the paper that resulted from this work [271]. The sketches cover diverse application domains such as product design (a,b,e,g), architecture (c,d), and character design (f). Note how VR sketching facilitates design in context, as illustrated with the VR headsets drawn using a 3D head as an underlay to achieve accurate proportions and contacts. Most sketches were done in short sessions of 5–20 minutes, while the 1:1 scale car design (3.2m in length) took an hour and 40 minutes.

6.5.1 Comparisons to Commercial VR Modelling Software and 2D Sketch-based Modelling

A focused user study comparing new research prototypes like CASSIE, to commercial software like Gravity Sketch [91], or CAD tools (Sketchup, Blender) is difficult for multiple reasons: CAD modelling is fundamentally different from CASSIE in workflow and interface; Gravity Sketch also uses disparate workflows for curve ideation and concept surface creation; mature commercial systems have many addi-

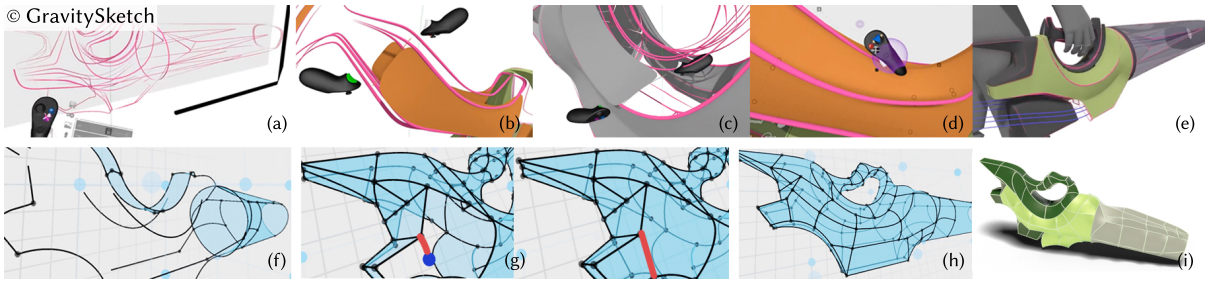


Figure 6.15: Comparing design workflows in Gravity Sketch [91] (top) and CASSIE (bottom). In Gravity Sketch, the user starts with a loose sketch composed of strokes with no explicit connectivity (a). Adding surfaces comes at a later stage, either by bridging nearby curves (b) or by roughly aligning the surface to the strokes (c). The user can then refine the surfaces via control points. With CASSIE, the user simultaneously sketches well-connected curve networks (f) and surfaces automatically appear as a cycle is closed (g) by a new stroke (red). Surfaces can be refined by adding strokes that directly control the shape of the surface (h). The final result is a leaf blower (e,i) coloured and rendered in Blender.

tional controls like stroke type, colour, and polished renders that can confound participants' perception of creative support; and several of our participants already had extensive experience with commercial systems, while they were all entirely unfamiliar with CASSIE. As Cherry and Latulipe, the creators of the Creative Support Index [46], point out, "the key is to ensure that only one of the factors (tool, task, and expertise of participant group) varies at a time; otherwise, comparison of the CSI score between two different treatments will be difficult to decipher."

We did, however, qualitatively compare our system to Gravity Sketch [91], a commercial software that allows the creation of 3D curves and surfaces in VR. A major difference between this system and ours is that Gravity Sketch provides completely independent facilities for curve and surface creation that are switched between using a modal interface. As a result, users typically first lay down the salient feature curves of the model before draping these curves with surface patches. Since the curves are drawn without explicit connectivity, users need to explicitly define every single surface patch to be created, as illustrated in Figure 6.15 that shows a sequence of modelling steps extracted from an online tutorial (https://youtu.be/ymCe5C_IlF4). In contrast, users of CASSIE can quickly create a similar model by seamlessly drawing the feature curves over the inferred surface patches.

We qualitatively extend this comparison to desktop-based CAD modelling tools by looking at the perceived advantages of CASSIE from participants with varied experience with those tools. P1 particularly liked how CASSIE's natural interface contrasted with their prior experience with Gravity Sketch and desktop CAD tools, remarking verbally: "I need the most direct tool possible in order to maintain a state of creative flow. The tool shouldn't obstruct that, which is what I find to be an issue in desktop CAD tools and Gravity Sketch where the interface presents many different modes and tools to me. In contrast, [CASSIE's] sketching interface helps me express this creative momentum." P3, who rated themselves as a beginner with CAD modelling (2/5 on prior CAD experience), appreciated the fact that CASSIE allows for "quick generation of sketches in VR without cumbersome tooling and commands." Similarly, participants with extensive CAD experience remarked that CASSIE is well suited to "quickly express a 3D concept to colleagues" *(P4)* or for "fast modelling prototyping." *(P8)* P5, who experimented with a mixed workflow with ZBrush (Figure 6.16c), appreciated the use of CASSIE to build a rough shape as it enabled them to "step into the model and create the curves and shapes I was thinking." Still, Gravity Sketch and other mature modelling tools provide a wide range of precise editing features

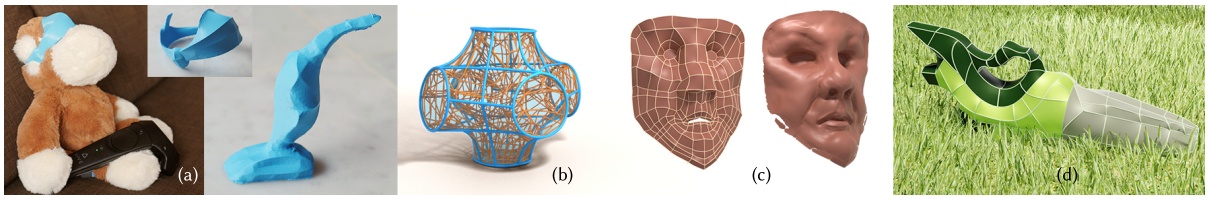


Figure 6.16: Applications: (a) 3D printed HMD from Figure 6.14f and vacuum cleaner from Figure 6.14a, (b) structural analysis and truss generation using Arora et al. [15], (c) detailed face model sculpted by P5 by sketching in CASSIE (left) and then refining patches in ZBrush [187] (right), (d) coloured and shaded render to use as communication material.

which were missed in CASSIE by some participants (P1, P10), thus pointing out an interesting avenue of future work in combining both manual and assisted creation paradigms.

Another comparison can be drawn to sketch-based modelling tools that *lift* 2D sketches into 3D. While CASSIE adopts a natural interface, such tools typically require designers to either follow a strict creation style [209] or provide additional manual annotations [267]. Moreover, layered or geometrically complex 3D objects—such as the hat (Figure 6.1c), buildings (Figure 6.14c–d), or HMDs (Figure 6.14e)—that do not have a single descriptive 2D viewpoint [65] can be extremely cumbersome or impossible to design using purely 2D sketch-based modelling tools.

6.5.2 Downstream Processing

3D models created in CASSIE are readily usable in downstream software, as illustrated in Figure 6.16. Using remeshing tools [48, 104], CASSIE patches can be merged to form a manifold surface, which can then be 3D printed (Figure 6.16a). Our surfaced models can also be used for engineering applications; for instance, for creating volumetric trusses (Figure 6.16b). Finally, CASSIE models can be refined in commercial sculpting applications (Figure 6.16c) or coloured and shaded for presentation (Figure 6.16d).

6.6 Conclusion

Ideation and concept modelling are the foundation of 3D design. In current practice, 2D sketches dominate ideation, and disparate CAD-like tools produce 3D concept models. Our system CASSIE showcases the ability of VR/AR to effectively support both ideation and concept modelling in a shared immersive space. CASSIE combines the fluidity of freehand sketching with geometric and aesthetic constraints of 3D modelling, to predictively neaten, structure, and surface user strokes into 3D models of sufficient quality for downstream applications. Importantly, our study reveals that such progressive structuring and surfacing is not detrimental to user agency, is seen by some users as an aid in creating and perceiving a solid 3D shape, and acts as an incentive to draw well-connected 3D concept models.

6.6.1 Limitations

While automatic neatening greatly eases the creation of connected curve networks, our current implementation based on a single proximity threshold was judged too intrusive by some participants. Several strategies could be explored to provide finer control on neatening, such as adapting the threshold to stroke speed or pressure, or learning a user-specific threshold by analyzing a few annotated freehand drawings

of that user. Additionally, exploring interactive solutions to edit and manually tune the neatening in ambiguous cases, as well as combining manual deformation and edition of strokes [22, 91, 171, 250] with automatic neatening are promising avenues of research to enhance user agency in the neatening process. Surface generation could also be improved, for instance by better reproducing the curvature depicted by the armature [176], or by proposing surface editing interactions based on the sketched strokes [142, 220]. In order to bootstrap these efforts, we openly release the interaction data from the 72 sketches in our study as well as the 16 other sketches depicted in the paper, including raw user strokes as well as the neatened curves at https://gitlab.inria.fr/D3/cassie-data. We have also released the CASSIE source code for academic research at https://gitlab.inria.fr/D3/cassie.

6.6.2 Future Work

While our current prototype allows users to switch between creative exploration and more precise modelling simply by enabling neatening and surfacing, we see several avenues to achieve a continuum between these two forms of sketching. On one hand, freehand strokes could be used as underlays to trace more definite armatures, and as extra guidance in our optimization to best position the neatened curves and surface patches. On the other hand, sparse armatures could be used as structured spatial deformers to warp denser freehand strokes, allowing designers to explore design variations while retaining the original look of their ideation sketches. The VR/AR setting could also be leveraged for sketching in situ, as suggested with the headsets sketched over a head in Figure 6.14. In this usage scenario, the contextual 3D models could be used as extra cues for stroke beautification and surface generation.

Another line of work concerns focus on the effectiveness of ideation and sketch-based communication. According to Buxton [36], an important aspect of ideation sketches is that they do not come across as finished, thereby inviting suggestions from reviewers and collaborators. It would be interesting to explore shape representations and rendering techniques which allow a designer to explore physically-realizable 3D shapes, while also communicating a loose intent and eliciting comments from their peers. This is a challenging but important problem, and we expect significant research into this problem in the near future.

We believe that mid-air drawing in immersive environments has the potential to significantly disrupt the interactive 3D design process, and CASSIE is a positive step in that direction. Moving beyond static geometry, the next chapter investigates the use of mid-air interactions for immersive animation.

Chapter 7

Hand Gestures for Animation Authoring

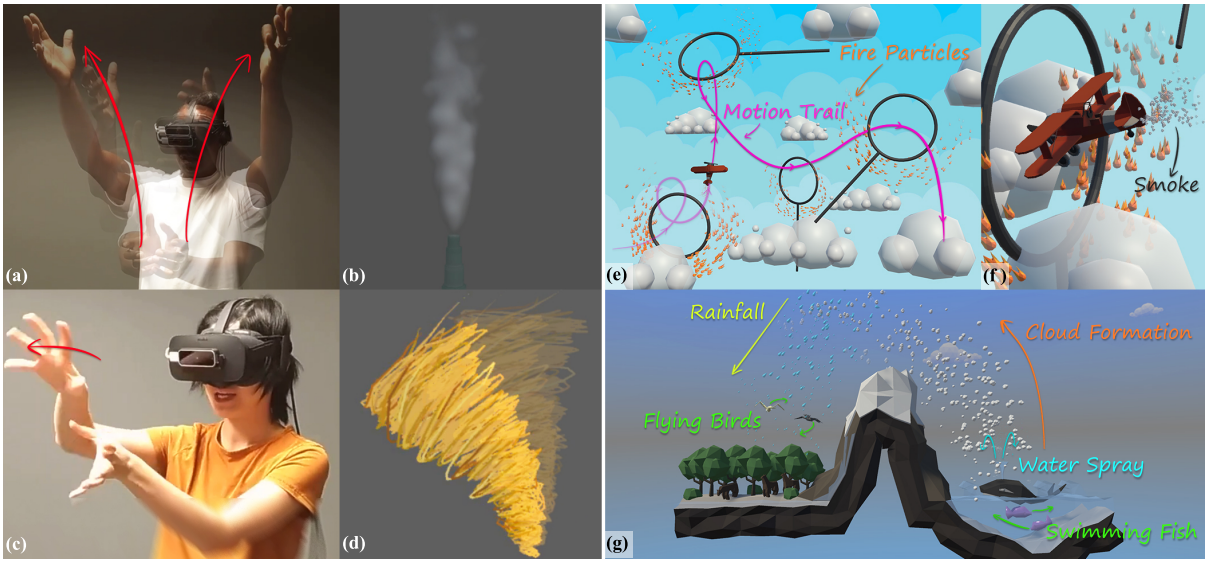


Figure 7.1: I study the use of mid-air gestures for animation authoring in VR, utilizing high-level creation tasks to understand the basic operations utilized by animators, and the features of gestures undertaken to effectuate the same. Here, I show two example gestures from the study: a high-bandwidth gesture manipulating the direction, spread, and randomness of smoke emission (a–b) and a gesture directly bending an object to describe a *follow-through* behaviour (c–d). I then built an animation system—MagicalHands—based on the insights gained from the study. The system supports 3D manipulation and particle systems, which were used to create an airplane stunt scene (e–f) and a water cycle visualization (g). Annotations added to indicate animated phenomena. Please see the accompanying video for the effects in action. Water cycle model © Hermes Alvarado; used with permission.

The previous chapters looked into the use of immersive environments and mid-air interactions for various geometric modelling tasks. In this chapter, we will move beyond static geometry and investigate the use of Virtual Reality for immersive animation. We will also go beyond tracked external devices and explore the use of hand gestures as the medium of interaction.

Hand gestures are a ubiquitous tool for human-to-human communication, often employed in con-

junction with or as an alternative to verbal interaction. They are the physical expression of mental concepts [237], thus augmenting our communication capabilities beyond speech [158]. The expressive power of gestures has inspired a large body of human-computer interaction research on gesture-based interfaces, from generic 2D manipulation techniques [190, 254, 258] to specialized tools for 3D modelling [136, 173, 204, 270], 3D model retrieval [101, 268], and visualization [9]. Mid-air gestures allow users to form hand shapes they would naturally use when interacting with real-world objects [228, 268], thus exploiting users' knowledge and experience with physical devices.

This chapter investigates the application of gestures to the emerging domain of VR animation [71, 167, 168, 246], a compelling new medium whose immersive nature provides a natural setting for gesture-based authoring. Current VR animation tools are primarily dominated by controller-based direct manipulation techniques, and do not fully leverage the breadth of interaction techniques and expressiveness afforded by hand gestures. Here, we will explore how gestural interactions can be used to specify and control various spatial and temporal properties of dynamic, physical phenomena in VR.

A key contribution of this work is an empirical study that explores user preferences of mid-air gestures for animation. The main challenge for such a study is that animation is a vast discipline with distinct and diverse authoring strategies [180]. Moreover, even animating a simple scene can be a complex task with multiple workflows and many individual operations. Thus, unlike previous gesture elicitation studies [199, 261], we cannot easily define and collect gestures for a pre-defined set of atomic target operations.

To address this challenge, I restricted the scope of this investigation to physics-based animation. The physical plausibility of effects like collisions, deformations, and particle systems can strongly enhance immersion and presence in VR, which makes this an important class of phenomena to study. In addition, unlike prior work on performance-based animation that leverages the relatively direct mapping between humans and articulated digital characters [183, 264], the mapping between hand gestures and physics-based effects is much more ambiguous and requires further investigation.

In terms of the study design, I developed a methodology that supports the unstructured nature of animation tasks. I created several simple but realistic animated scenes in VR to act as the target stimuli. I then recruited 12 professional animators and asked them to create each animation (which involved multiple individual steps) using gestures, while following a think aloud protocol. Based on direct visual observation and their spoken explanations, I segmented the data into individual interactions that were then analyzed along several dimensions. This approach enabled comparison between the use of gestures across participants and identify common usage patterns.

The study elicited many high-bandwidth gestures assuming simultaneous control of a number of parameters (Figure 7.1). However, I also found that most gestures did not encode information in the nuances of hand shape, and thus, even coarse hand-pose recognition should suffice for building gestural VR animation interfaces. From these findings, I propose several design guidelines for future, gesture-based, VR animation authoring tools. Finally, as a proof of concept, I implemented a system, MagicalHands, consisting of 11 of the most commonly observed interaction techniques in the study. The implementation leverages hand pose information to perform direct manipulation and abstract demonstrations for animating in VR.

To summarize, the main contributions of this chapter are

— a gesture elicitation study on creating dynamic phenomena in VR,

— a taxonomy of mid-air gestures for VR animation,

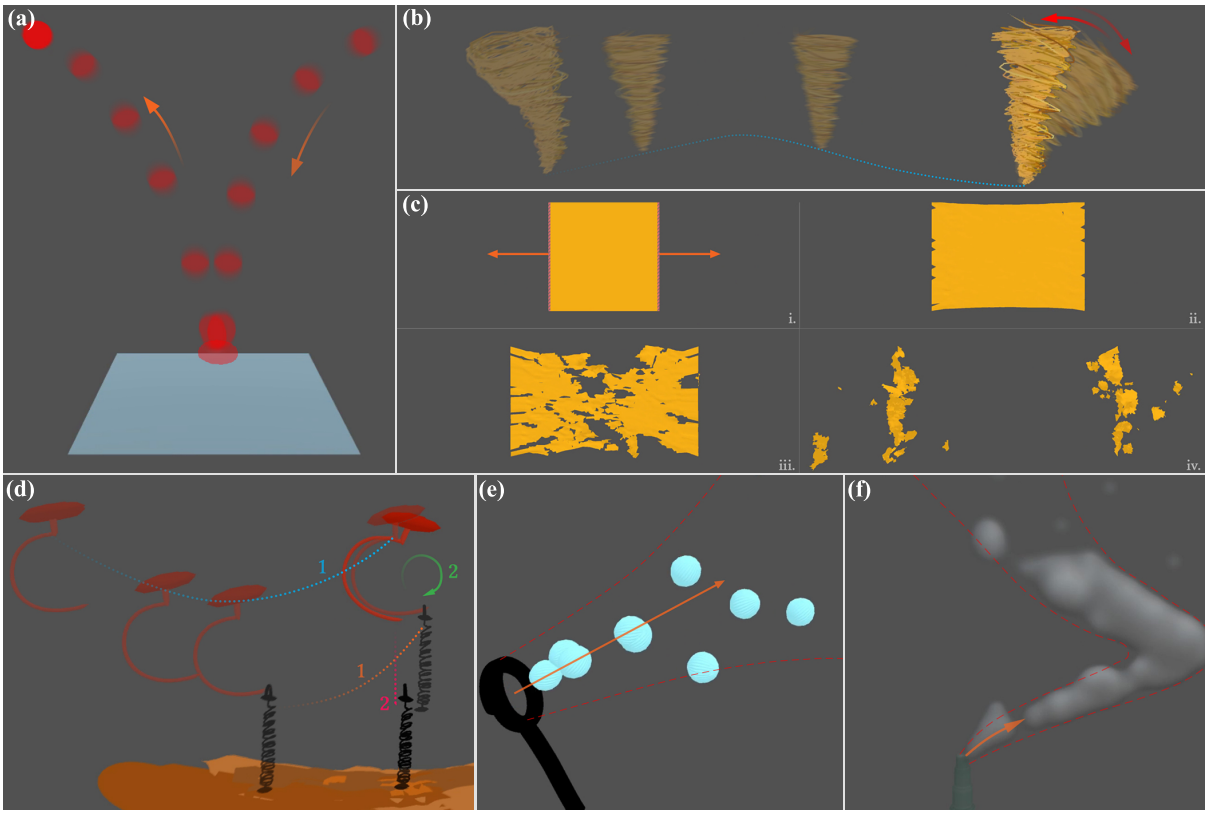


Figure 7.2: The six phenomena studied in our experiment—showing one clip from each phenomenon here. Bouncing ball with squash and stretch (a), tornado motion with follow through/bending (b), topological changes in shattering sheet animation (c), physical coupling and decoupling (d), particle system making a bubble blowing animation (e), and smoke simulation (f). Please see the online video https://youtu.be/YOJVYCQeXFs for the animated clips.

— a set of commonly observed gestures for VR animation,

— a set of design guidelines for VR animation tools, and

— a prototype animation system and informal study to assess the efficacy of gestural interactions for VR animation.

7.1 Hand Gesture Usage Study

My overall goal was to elicit a broad range of gestures for creating VR animations in order to derive design guidelines for a gestural animation system. A challenge of this setting is that animation involves many individual operations to select and modify various spatial and temporal properties. Furthermore, animators can choose to specify such properties at the individual entity level or control high-level physical (e.g., material stiffness) or abstract (e.g., amount of noise) parameters. Thus, a key objective for the study was to elicit both a diverse set of atomic operations and various gestures for executing those tasks. To this end, I created a range of target scenes covering common dynamic effects and then asked participants to describe operations and gestures to create each animation.

7.1.1 Target Animated Scenes

Animated stories include a diverse range of motions and dynamic effects. As noted earlier, previous research on gesture-based animation largely focuses on the problem of animating humanoid characters through facial or full-body performance where the degrees of freedom of the performer and the target character are often very similar.

My study focuses on common classes of physical systems like particles, fluids, and multi-body interactions where the gesture-to-motion mapping is less obvious than for characters.

Based on an analysis of animation literature [84, 180, 265] and physical phenomena featured in commercial animation software [21, 218, 235], I created six scenes covering a range of physical effects (Figure 7.2). For each scene, I generated 3–4 target animations to cover typical stylistic variations and presented these variants to participants in increasing order of visual complexity.

Bouncing Ball. This scene shows a ball bouncing once on the ground. It includes a collision and a response, typical for physically-based object interactions. The first animation shows the ball moving at a constant speed; the second exhibits easing around the collision event; and in the third, the ball squashes and stretches. This scene thus helps study rigid-body interaction as well as elastic deformation.

Tornado. This scene involves layering multiple simultaneous motions on an object. The first animation shows a rotating tornado; the second adds a motion path; and the third includes a follow-through effect, bending at the end of the path.

Bubbles. This scene includes variations of a simple particle system where bubbles emanate from a wand. In the first animation, the bubbles appear at regular intervals in groups of 10; the second adds high-frequency vibrations to each bubble; and the third staggers the timing so that bubbles appear one at a time. This scene was included to understand how animators gesturally interact with a particle system, an entity typically controlled using multitudes of abstract parameters.

Hook and Spring. This scene investigates physical coupling and decoupling of objects, which occurs in many multi-body interactions. Animators often use such interactions to emphasize the physical attributes of objects and add secondary motion to a scene.

The initial animation involves a hook picking up a spring object as it moves along a path; the second adds vertical vibrations to the spring after it gets picked up; and in the third, the hook rotates to drop the spring at the end.

Sheet. This scene includes shape and topology changes to a sheet that gets stretched from opposite edges. The first animation shows the sheet stretching in an area-preserving manner; in the second, the sheet tears apart down the middle; the third variant modifies the shape of the tear to be a complex curve; and the last shows the sheet disintegrating into small pieces, which fly away as it tears. These animations explore the well-studied cloth simulation scenario as well as the topological changes introduced by ripping and shattering.

Smoke. The last scene involves several variations of a smoke simulation. The first variant shows nearly-laminar flow straight up from a chimney; in the second, the smoke follows a curved, non-planar

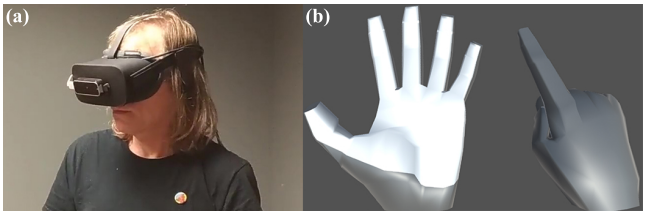


Figure 7.3: Experimental setup: Leap Motion mounted on an Oculus HMD (a), enabling participants to see their hands while gesturing (b).

path; and finally, the last clip adds turbulent noise and dispersion to the flow. This scene was utilized to understand how gestural control of a continuum of fluid can be achieved.

Of course, these examples do not cover all possible physical phenomena. However, the target animations (19 in total) span a wide range of dynamic effects and modifications that are common across real-world animations.

7.1.2 Participants

Twelve animators (9M, 3F) aged 19 to 58 (median 42) were recruited for the study. All participants had a minimum of two years of experience creating animations, and five had over a decade of experience. Participants had diverse backgrounds in 2D animation, 3D character animation, motion graphics, and procedural animation. Participants had experience with numerous software packages including After Effects, Maya, Blender, Character Animator, Apple Motion, Toon Boom Harmony and Unity. While most had tried VR, none had authored VR animations.

For four animators (P1–P4), an initial session with only the first three target scenes was conducted. P1 and P2 completed the remaining target scenes in a follow-up session, but P3 and P4 were unavailable. The other participants (P5–P12) went through the study in one session with a break in the middle. The study took between 75 and 120 minutes.

7.1.3 Apparatus and Implementation

Participants wore an Oculus Rift HMD with a Leap Motion hand tracker mounted on it, allowing them to see non-photorealistic renderings of their hands while gesturing (Figure 7.3). Note that allowing participants to see their hands was the sole purpose of the Leap Motion device, and the tracking data was not utilized for analysis.

The first four target scenes were created using Oculus Quill's frame-by-frame animation tools. *Sheet* animations were simulated in Maya [21] and then imported into Unity [247], while the *smoke* was simulated in real-time using the smoothed-particle hydrodynamics (SPH) [133] implementation in Fluvio [239].

7.1.4 Procedure

In each session, target animations were presented to the participant in the order listed above. For each example, I started by showing the static scene objects and then playing the target animation. Participants were asked to consider the static objects as the input for the task and author the animation with gestures. Since the aim was to elicit a broad range of behaviours, participants were encouraged to

demonstrate additional gestures after their initial attempt for each task. Moreover, the initial interaction was expected to be biased by the dominant interface elements in participants' preferred animation software, and eliciting additional gestures could help alleviate this bias.

An experimenter was always present to answer questions, provide clarifications, and update the state of the virtual scene (e.g., participants could ask to see the static objects, replay the animation, or pause at a frame). Sessions were audio and video recorded for later analysis.

During each task, participants were asked to think-aloud as they worked. Given the complexity of the target tasks, it was critical for participants to verbalize the intent of each gesture and describe how they expected the system to respond. For example, in some cases, participants demonstrated high bandwidth gestures and explained how these actions were meant to accomplish multiple atomic operations. In other situations, participants proposed the use of traditional UI components like menus, buttons, and sliders to modify specific attributes of the animation. Here, the choice of professional animators as participants proved to be useful: lacking direct visual feedback from the scene, novices may have missed controlling some of the more subtle aspects of the complex and diverse animation tasks. In this respect, the experiment differs considerably from more structured elicitation studies such as Wobbrock's [261], where each participant's gestures have a clear one-to-one mapping with the intended effects.

7.2 Results

The recorded user sessions were analyzed to identify the most commonly observed gestures for our VR animation tasks (Figure 7.4). The gestural interactions were then taxonomized along geometric and semantic dimensions and common trends in how participants expressed various spatio-temporal operations were analyzed.

7.2.1 Analysis Methodology

For each participant, we manually segmented the recording into separate disjoint interactions. Most interactions (493 total) were gestural, and these were categorized based on existing gesture taxonomies. Since participants also had the option of describing interactions with traditional widgets, 133 non-gestural interactions with imaginary menus, buttons, sliders, etc. were also observed. Figure 7.4 presents commonly observed gestures during the study sessions for transformations, deformations, coupling, and interacting with particle systems. The resulting gestures demonstrate the breadth and richness of mid-air interaction techniques to control dynamic phenomena.

7.2.2 Taxonomies of Interactions

The gestural interactions were categorized along four separate dimensions: two that characterize geometric features, and two that describe the semantics of all interactions. For each gesture, the effect participants wanted to execute with the interaction was also categorized. After forming the list of codes, my coauthor on the associated publication [18] and I independently coded each interaction for a subset of data (approx. 10%) into the taxonomy of actions and effects (Table 7.2) and as either gestural or non-gestural, with gestural interactions further coded into a taxonomy of gestures (Table 7.1). Since the Cohen's kappa measure showed excellent agreement between the two authors on all four dimensions ($\kappa \in [0.81, 0.97]$), I then proceeded to code the rest of the data all by myself.

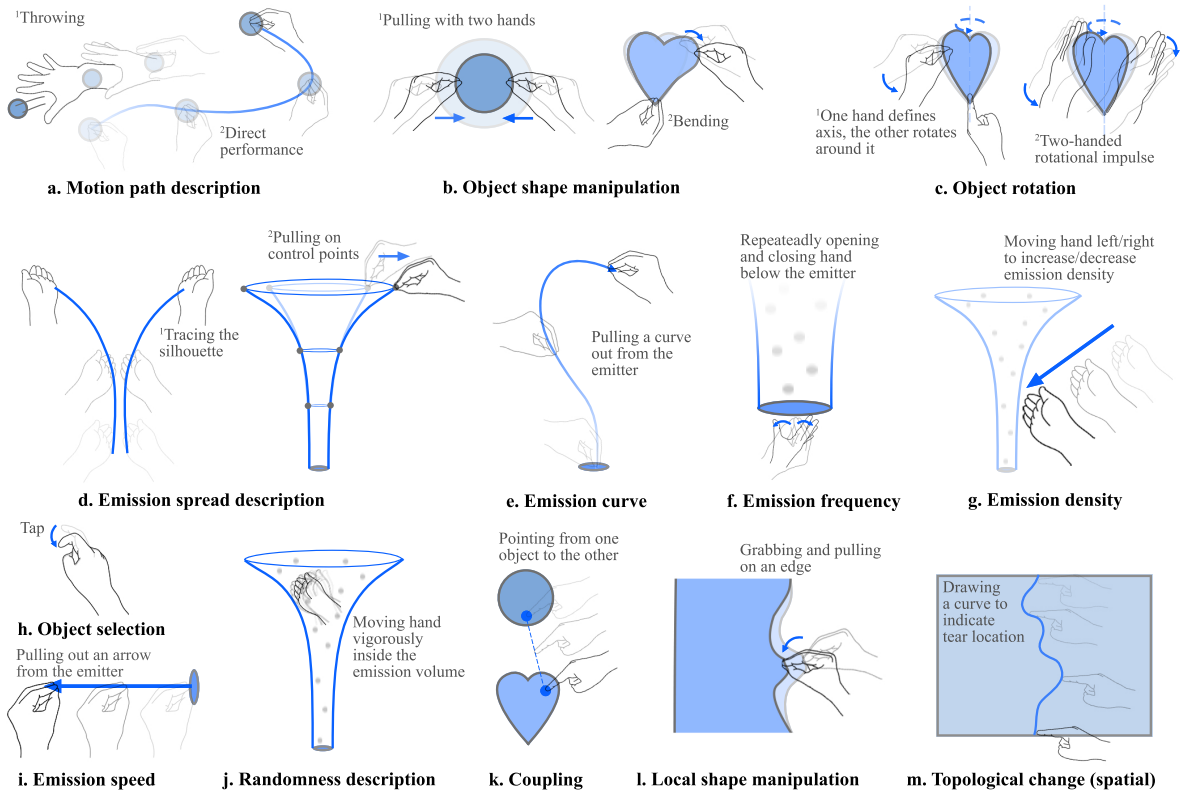


Figure 7.4: Most commonly observed gestures (dominant gestures for commands which had at least 10 occurrences in the study). These gestures include direct manipulation (a–c) and abstract demonstrations (d, f, g, i, j). Many involve interactions with physics (a, c), simultaneously specifying multiple parameters, such as speed and emission cone (d), and coordinating multiple parallel tasks using bi-manual interactions, where the non-dominant hand specifies the rotation axis, and the dominant hand records the motion (d).

Geometric Taxonomy of Gestures

Gestures were classified along taxonomies based on hand usage [35] and form [261] (Table 7.1).

Hand usage distinguishes between *unimanual* and *bimanual* gestures, with bimanual interactions further classified as *static* where one hand stays still, *symmetric* if both hands move about a point or plane of symmetry, or *other* if the two hands move independently. Typical examples include deforming an object by using one hand as a static constraint (Figure 7.1c–d: *static*), defining an emission cone by tracing its silhouette (Figure 7.4d: *symmetric*), and moving in time with one hand while setting a parameter with the second (*other*).

Form categorizes if and how the pose and position of hands vary within a gesture. Table 7.1 briefly describes and Figure 7.5 illustrates the four categories. See Wobbrock et al. [261] for details. Examples include using extended hands to pause: *static pose*; tracing out a path (Figure 7.4a): *static pose and path*; tapping to select (Figure 7.4h): *dynamic pose*; and throwing objects (Figure 7.4a): *dynamic pose and path*. A bimanual gesture is considered to have a dynamic pose and/or path if either of the hands satisfies the respective criteria.

Table 7.1: Taxonomy of mid-air gestures for animation.

Hand Usage	*unimanual*	Only one hand is actively utilized.
	bimanual-static	One hand moves, the other is fixed.
	bimanual-symmetric	Both hands move symmetrically.
	bimanual-other	Both hands move independently.
Form	*static pose*	Hand pose is held at one location.
	dynamic pose	Hand pose changes at one location.
	static pose and path	Hand pose is held while the hand moves.
	dynamic pose and path	Hand pose changes as the hand moves.

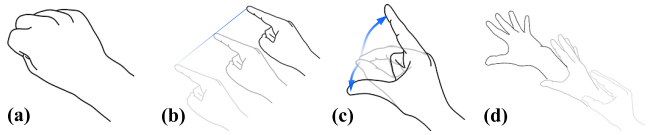


Figure 7.5: Gesture *form* classification: *static pose* (a), *static pose and path* (b), *dynamic pose* (c), and *dynamic pose and path*.

Semantic Taxonomy of Interactions

For all gestural and non-gestural interactions, I also considered what desired action the interaction conveys (i.e., the *nature of action*) and the intended effect on the animation (i.e., the *nature of effect*).

Table 7.2: Taxonomy of interactions based on the nature of the user's action and the intended effect for VR-based animation authoring.

NATURE OF ACTION	
direct manipulation	Hand(s) interact with objects in the scene directly.
demonstrative	Features of hand pose and/or path emulate the intended effect.
semaphoric	Abstract or learned relation between gesture and effect.
widget use	Manipulation via a traditional widget, such as a push button, toggle, or slider.
NATURE OF EFFECT	
spatial	Addition or editing of a purely spatial property.
temporal	Manipulation of the timing of an animated effect.
spatiotemporal	Altering both spatial and temporal properties.
abstract	Manipulation at a higher abstraction level.
interface	Navigational tools that do not modify the animation.

To classify the *nature of actions*, I took cues from existing gesture taxonomies [7, 121, 261], but customized the categories to focus on animation-specific features (Table 7.2). *Direct manipulation* gestures directly influence an object in the scene, including direct performances of trajectories, sculpting to alter object shape, and throwing or flicking gestures that induce motion via an implied underlying physics engine. In contrast, *demonstrative* gestures do not manipulate objects directly, but some spatiotemporal properties of the action mimic the desired effect. Examples include mimicking the shape change of an object using the hand's pose, manipulation of assumed spatial interfaces such as emission cones or motion trails, and *lasso*-ing to group objects.

Semaphoric gestures—such as extending both hands forward to indicate pause or stop—rely on accepted interpretations of specific gestures or poses to indicate a desired effect. Finally, users wanted

to perform some operations using abstract controls such as buttons and sliders. Tasks where such *widget uses* dominate may not be well-suited for gestural control.

I also considered the *nature of effects* that each interaction intended to produce. These categories are specific to the animation authoring domain. Some interactions only modified *spatial* properties of the scene, such as scaling or positioning an object at a given instant to set a keyframe. Conversely, some interactions only modified *temporal* properties, such as stretching a motion trail like an elastic band to locally "expand time", or non-gestural interactions to set speed variables. Other interactions modified *spatiotemporal* attributes. For example, direct demonstration of how a spatial property changes over time, such as tracing an object's rotation or pulling away from an emitter to describe both the speed and trajectory of particles. Throwing and flicking manipulations also fall into this category since they induce spatiotemporal behaviour via implied physics. *Abstract* interactions such as duplication, coupling objects together, and setting physical parameters (e.g., mass, elasticity) manipulate the animation at a more abstract level, beyond explicit spatial or temporal scene properties. Finally, *interface* interactions were used to navigate through space and/or time (e.g., to view a specific part of the clip) and did not explicitly modify the animation.

Taxonomic Distributions and Relationships

Figure 7.6 shows how the elicited interactions were distributed across the classification dimensions. Here, interesting second-order effects between the classification dimensions were also observed.

A large number (35%) of interactions were *direct manipulations* (Figure 7.4a-c), which can perhaps be explained by the enhanced feeling of presence attributed by participants (P3, 4, 7, 10–12) to the immersive space and hand tracking. Several of these gestures included participants "performing" the intended motion, illustrated by two-thirds of the *spatiotemporal* interactions being *direct manipulations*. Among purely *spatial* interactions, while 35% were effectuated by *direct manipulations*, over 54% utilized the indirect *demonstrative* actions such as manipulating motion trails and emission cones.

While non-gestural *widget use* was requested for nearly 45% of the *temporal* interactions, participants still used *demonstrative* gestures such as stretching timelines for 40% of those. For *abstract* interactions, *widget uses* jumped to 50%, while 25% of the actions were *semaphoric*.

Gesture *forms* with a *static pose* were generally utilized for *interface*-related tasks only (10/11 gestures). *Static pose and path* gestures were utilized for diverse tasks ranging from *direct manipulations* to pose and perform, to *demonstrative* interactions specifying the rate of smoke emission, and to *semaphoric* actions for duplication. A *dynamic pose and path* was used for physics-affecting *direct manipulations* such as throwing and shattering, for high-bandwidth gestures defining multiple *abstract* properties with a single interaction, and for dragging and dropping *interface* elements.

Bimanual gestures tended to involve a *static pose and path*, (over 90%). Among gestures with *symmetric* hand motion—which formed 64% of all *bimanual* gestures—*spatial* effects such as scaling were the most common (57%), followed by *spatiotemporal* ones (25%) such as performing a rotation (Figure 7.4c).

A unique aspect of the experiment compared with existing gesture studies is the lack of predefined operations that the gestures are meant to execute. To help identify relationships between the taxonomies defined above and key tasks in animation authoring processes, I identified 39 unique *atomic operations* from the elicited interactions. Table 7.3 describes the distribution of various taxonomic classes across these operations. I now describe the process used to arrive at these atomic operations.

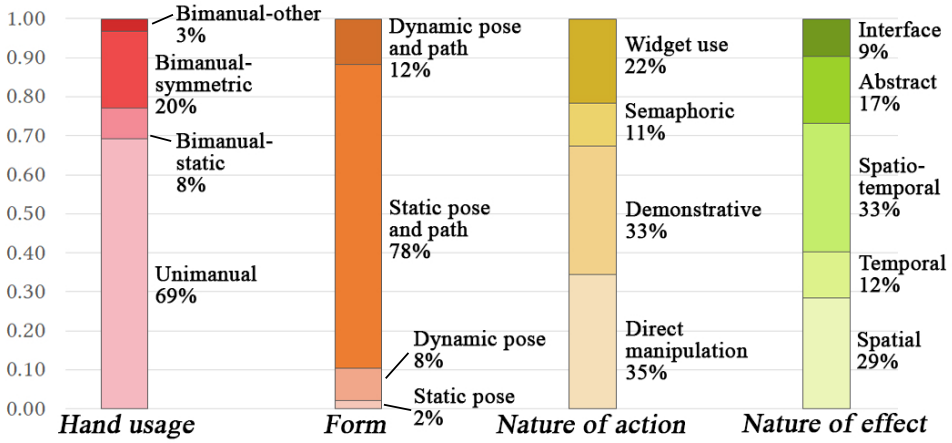


Figure 7.6: Distribution of gestures and all interactions in the taxonomies.

7.2.3 Observed Effects to Atomic Operations

Each gestural or non-gestural interaction observed in the study comes in as an *action-effect pair*—the participant executes an action which is meant to create an effect in the animation system. Given the variety of tasks in animation, as many as 70 unique effects were observed in our study. As noted earlier, many of the observed actions sought to affect numerous objects and parameters in the scene. However, the desired effects could still be categorized into either *simple* or *compound*. Concretely, if $\mathbf{E}$ is the set of unique effects, an effect $e \in \mathbf{E}$ is called *compound* if and only if it can be expressed as a logical union of other effects in the set. That is, if

$$e = \bigcup_{f \in \mathbf{E} \setminus \{e\}} f. \tag{7.1}$$

All other effects are called as *simple*. For example, if setting the emission speed of a particle system and setting the emission path (or curve) of that system are both simple effects, then an action implying both the effects is considered to be *compound*. After limiting ourselves to the set of *simple* effects, it was noticed that some of the effects were similar in spirit, and could be grouped together. For example, **bending description** used to bend the tornado and **squashed/stretched shape description** used to deform the bouncing ball could be grouped as **object shape manipulation**. After finishing this process, I arrived at a set of 39 distinct *atomic* operations. Note that the atomic operations themselves may still be high-dimensional, but were the most basic operations observed in our study. For completeness, the list of *simple* effects is included in Table 7.4. Further, all the *simple* effects observed in a single *compound* effect have been marked with a shared superscript. The next section defines all 39 atomic operations, while Table 7.3 reports the observed frequency of the taxonomic classes for each.

7.2.4 Description of Atomic Operations

I now briefly describe the atomic operations observed in the study. In general, I will use the term *manipulation* to refer to the editing of an already existing entity, *specification* to the definition of a numeric parameter, and *description* to the definition of a parameter which cannot generally be expressed as a single number.

Table 7.3: Observed atomic operations with classification of the interactions utilized for each. The numbers in the cells represent the observed frequencies of each taxonomic class for the actions performed by the participants. U–*unimanual*, B.St–*bimanual-static*, B.Sy–*bimanual-symmetric*, B.O–*bimanual-other*, N/A–*no gesture*, S–*static pose*, S.P–*static pose and path*, D–*dynamic pose*, D.P–*dynamic pose and path*, DM–*direct manipulation*, Dt–*demonstrative*, Sm–*semaphoric*, W–*widget interaction*, Sp–*spatial*, T–*temporal*, S.T–*spatiotemporal*, A–*abstract*, I–*interface*.

	Hand Usage					**Form**					**Nature of Action**				**Nature of Effect**				
	U	B.St	B.Sy	B.O	N/A	S	S.P	D	D.P	N/A	DM	Dt	Sm	W	Sp	T	S.T	A	I
Abstract parameter manipulation	4	0	0	0	0	0	4	0	0	0	1	1	2	0	0	0	1	3	0
Attaching particle to emitter	3	1	0	1	1	0	4	0	1	1	5	0	0	1	0	0	1	5	0
Bringing up a menu	1	0	1	0	1	1	1	0	0	1	0	0	2	1	0	0	0	0	3
Characterizing object as emitter	4	0	0	0	1	0	1	2	1	1	0	1	3	1	0	0	0	5	0
Coupling	7	1	1	1	4	0	2	5	3	4	0	6	3	5	1	0	1	11	1
Decoupling	4	0	1	0	5	0	5	0	0	5	1	0	4	5	1	0	0	9	0
Duplication	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0
Emission curve description	19	4	5	0	0	0	22	1	5	0	6	22	0	0	17	0	10	1	0
Emission density description	11	1	1	1	12	1	11	1	1	12	4	5	5	12	2	0	2	20	2
Emission frequency description	8	4	2	2	7	0	10	2	4	7	2	10	4	7	1	17	0	4	1
Emission lifetime specification	5	2	0	0	7	0	6	0	1	7	0	7	0	7	0	0	2	12	0
Emission speed description	15	2	5	0	8	0	17	0	5	8	3	17	2	8	0	19	10	1	0
Emission spread description	22	0	20	0	2	0	36	1	5	2	4	37	1	2	36	0	6	2	0
Group path description	6	0	3	0	3	0	6	1	2	3	2	7	0	3	3	0	8	1	0
Group shape manipulation	2	0	0	0	3	0	1	1	0	3	1	1	0	3	1	0	4	0	0
Grouping	4	0	0	3	0	0	7	0	0	0	0	7	0	0	0	0	0	1	6
Local region selection	4	0	0	2	0	0	4	0	2	0	3	2	1	0	1	0	2	1	2
Local shape manipulation	7	0	2	2	1	0	10	0	1	1	10	1	0	1	0	0	10	2	0
Motion path description	73	0	3	1	11	0	65	3	9	11	60	11	6	11	18	5	54	8	3
Motion path manipulation	6	0	1	0	0	0	6	1	0	0	3	4	0	0	5	0	2	0	0
Moving a scaling anchor	1	1	0	0	0	0	2	0	0	0	1	1	0	0	1	0	1	0	0
Object rotation	43	15	13	0	7	0	53	0	18	7	61	8	2	7	10	2	64	2	0
Object scaling	0	0	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0
Object selection	17	0	0	0	0	7	2	7	1	0	3	2	12	0	0	0	1	0	16
Object shape manipulation	22	13	30	1	8	0	59	5	2	8	49	13	4	8	40	0	23	7	4
Pause/Play	5	0	1	0	0	1	3	2	0	0	0	0	6	0	0	0	0	0	6
Posing (move w/o record)	0	0	2	0	0	0	2	0	0	0	2	0	0	0	2	0	0	0	0
Randomness description	26	1	1	2	20	0	18	3	9	20	7	21	1	21	5	0	35	7	3
Record ON/OFF toggling	6	0	0	0	5	1	4	0	1	5	3	1	2	5	3	0	0	2	6
Scrubbing the timeline	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1
Secondary emission description	5	0	0	0	5	0	5	0	0	5	4	1	0	5	1	0	7	2	0
Timing manipulation	11	1	3	0	3	0	15	0	0	3	3	11	1	3	0	13	5	0	0
Topological change spatial description	28	1	4	0	2	0	30	0	3	2	7	25	1	2	28	0	4	2	1
Topological change timing description	4	0	3	0	16	0	5	0	2	16	3	3	1	16	2	12	5	4	0
UI panel manipulation	3	0	0	0	0	0	0	1	2	0	0	3	0	0	0	0	0	0	3
Vibrational amplitude specification	9	2	3	0	2	0	10	3	1	2	10	2	2	2	7	2	6	1	0
Vibrational frequency specification	6	2	0	1	6	0	7	2	0	6	7	0	2	6	1	5	8	1	0
World rotation	0	0	2	0	0	0	2	0	0	0	1	1	0	0	0	0	1	0	1
World scaling	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1

Abstract parameter manipulation. Manipulate an abstract parameter which does not have a direct spatial or temporal meaning, or a physical analogue in the scene. For example, using a diegetic slider to change the emission lifetime in the smoke scene.

Attaching particle to emitter. For a particle system, specify which object serves as the emitted particle geometry. In the study, this typically meant attaching the spherical bubble geometry (*particle*)

Table 7.4: All the observed *simple* effects along with the atomic operations they were grouped into.

Index	*Simple* effect	Atomic operation (after grouping)
1	Abstract parameter manipulation[f]	Abstract parameter manipulation
2	Attaching particle to emitter	Attaching particle to emitter
3	Bringing up a menu	Bringing up a menu
4	Bringing up a control panel	
5	Characterizing object as emitter	Characterizing object as emitter
6	Coupling[b]	Coupling
7	Decoupling	Decoupling
8	Duplication	Duplication
9	Emission curve description[fghijl]	Emission curve description
10	Emission density description	Emission density description
11	Emission frequency description	Emission frequency description
12	Emission lifetime specification[i]	Emission lifetime specification
13	Smoke speed description[hij]	Emission speed description
14	Particle speed description[kl]	
15	Emission spread description[fghjkn]	Emission spread description
16	Group path description[d]	Group path description
17	Group shape manipulation	Group shape manipulation
18	Grouping	Grouping
19	Local region selection[t]	Local region selection
20	Local shape manipulation	Local shape manipulation
21	Motion path description[bopq]	Motion path description
22	Motion path manipulation[rs]	Motion path manipulation
23	Moving a scaling anchor	Moving a scaling anchor
24	Object rotation	Object rotation
25	World scaling	World scaling
26	Object selection	Object selection
27	Bending description	Object shape manipulation
28	Boundary description[a]	
29	Shape manipulation	
30	Squashed/stretched shape description[a]	
31	Pause/Play	Pause/Play
32	Posing (moving without recording)[bo]	Posing (moving without recording)
33	(Degree of) randomness specification[m]	Randomness description
34	Random path description[m]	
35	Smoke disturbance description[h]	
36	Record ON/OFF toggling	Record ON/OFF toggling
37	Scrubbing the timeline	Scrubbing the timeline
38	Secondary emission description[nr]	Secondary emission description
39	Timing manipulation[ps]	Timing manipulation
40	Crack location description[cd]	Topological change spatial description
41	Tear location specification[e]	
42	Cracking time/flow description[cd]	Topological change timing description
43	Tearing time/flow description[e]	
44	Manipulating a UI panel	UI panel manipulation
45	Repositioning a UI panel	
46	Vibrational amplitude specification[qtu]	Vibrational amplitude specification
47	Vibrational frequency specification[qtu]	Vibrational frequency specification
48	World rotation	World rotation
49	World scaling	World scaling

to the wand (*emitter*) in the bubbles scene.

Bringing up a menu. Participants wanted to bring up diegetic or egocentric menus similar to desktop UIs for complex manipulations; e.g., selecting the emitter in the smoke scene might show a UI panel with smoke simulation properties.

Characterizing object as emitter. Used in the smoke and bubbles scenes to specify that the chimney and the wand were to be used as smoke and particle emitters, respectively.

Coupling. Specify that the timelines of two objects are coupled. Typically used to specify the physical coupling in the hook and spring scene.

Decoupling. Specify that the timelines of two objects which were previously coupled, should now be decoupled.

Duplication. Duplicate an object or group of objects. An example utilization was duplication of the bubble geometry in the bouncing ball scene by participants who approached the task from a keyframing perspective.

Emission curve description. Define a 1-D curve which describes the general path that particles emanating from an emitter should take. Note that this does not mean that the particles have to exactly follow this curve, as the **emission spread** described below can change how the particles stray away from the emission curve.

Emission density description. This refers to how many particles are emitted from a particle emitter in some unit of time. The emission density may be a numeric parameter (first clip of bubbles), or may be described more elaborately (waning density of smoke in the smoke scene).

Emission frequency description. Typically used in the bubbles scene, this refers to how often a particle emitter spews out particles. Similar to the **emission density**, this may or may not be a single number.

Emission lifetime specification. Particle systems specify a lifetime on emitted particles. This was either defined as the amount of time after which an emitted particle disappears, or as a spatial location on the emission curve where particles vanish.

Emission speed description. Speed of the emitted particles as they move along the **emission curve**. Similar to the **density** and **frequency**, the emission speed may either be constant, or a spatiotemporal variation scheme may be defined.

Emission spread description. This describes how far particles can stray away from the **emission curve**. One can think of the emission spread and the emission curve together as a "particle cone." Similar terminology is employed in commercial software such as Unity [247] and Unreal Engine [69].

Group path description. Description of the path taken by a group of objects. Participants typically used this operation to illustrate the motion of the cracked pieces in the fourth clip of the sheet scene.

Group shape manipulation. Changing the shape of a group of objects. One common use was the crumpling of the cracked pieces in the final clip of the sheet scene.

Grouping. This action was used to mark a set of geometric objects as a logical group. For example, grouping bubbles in the bubbles scene to allow future operations to impact the set as a whole.

Local region selection. Selecting a region of an object. For example, in the second and third clips of the sheet scene, participants would select the regions in the middle of the sheet to allow for a future **local shape manipulation** operations to depict the vibrations. Please observe the relevant clips in the online video at https://youtu.be/YOJVYCQeXFs.

Local shape manipulation. Manipulating the shape of a portion of an object, while the rest of it is constrained to maintain its shape. See **local region selection** above for an example.

Motion path description. Defining the path of an object in spacetime. For example, the paths taken by the ball and tornado in the bouncing ball and tornado scenes, respectively.

Motion path manipulation. Editing a motion path, such as that of the tornado in the tornado scene.

Moving a scaling anchor. While scaling an object, the choice of the origin of the coordinate system in which it is scaled is important. Participants mentioned the use of an "anchor" to indicate the position of the origin for scaling operations. A typical use was in the second and third clips of the hook and spring scenes since the spring first scales about its top point, and then about its base.

Object rotation. Rotating an object, such as the tornado in the tornado scene.

Object scaling. Uniformly scaling an object.

Object selection. Selecting an object so that further operations can be carried out on it.

Object shape manipulation. Changing the shape of a given object. Participants utilized this operation to accomplish many shape deformation tasks. For example, to squash or stretch the ball geometry in the bouncing ball scene, to bend the tornado object in the tornado scene, to non-uniformly scale the spring geometry in the hook and spring scene, and to change the shape of the sheet in the first clip of the sheet scene. Remarkably, many of these gestures implicitly simulated the three-dimensional widgets designed by Conner et al. [53].

Pause/Play. Pause or play the animation clip. While useful for the animator, note that this is purely navigational and does not change the resulting animation.

Posing (moving without recording). Participants tried keyframing object positions, most notably in the bouncing ball and the tornado scenes. Given the pervasiveness of both keyframed and performed motion paths in our study, we separate these two categories (the other is **motion path description**).

Randomness description. Adding randomness to a motion path. Typical uses included randomness in the paths of the cracked pieces in the final sheet clip, bubble vibrations in the last two bubbles clips (when participants interpreted this vibration as random motion), and the turbulence-like behaviour in the smoke clips.

Record ON/OFF toggling. In performance-based animation, it is crucial to specify when an animator is posing an object vs. when they want to use their hand motion to actually define a motion path. Many a time, participants utilized an explicit gesture to indicate whether they wanted their actions to pose (record OFF) or to actually move an object (record ON). This operation was typically utilized in the bouncing ball, tornado, and hook and spring scenes.

Scrubbing the timeline. This refers to moving along the animation timeline. Note that this operation in itself does not modify the animated scene.

Secondary emission description. While working with particle emitters, some participants had the notion of a carefully crafted "primary" emission, along with "secondary" emissions which were treated more as random events. This was usually employed for the last two clips of the smoke scene to define small wisps of smoke emanating from the main particle cone.

Timing manipulation. This refers to operations—such as locally stretching or compressing an object's timeline—which impact only the temporal properties of an animated effect.

Topological change spatial description. All but the first clip of the sheet scene involved topological changes in the animated object. Participants used this operation to define the shape of the tears and shatters.

Topological change timing description. Continuing from above, participants also wanted to indicate when the topological change happens. This involved either setting up different "glue strengths" along the edges where the sheet tore to indicate when each portion of the sheet breaks, or directly indicating when each portion breaks apart by scrubbing the timeline and indicating the portions which break at each instant of time.

UI panel manipulation. This operation refers to moving a UI panel (see **bringing up a menu**) in space in order to focus on a particular panel, relocating a diegetic panel w.r.t its parent object, or to hide a panel to allow for a better view of the animated clip.

Vibrational amplitude specification. Many participants looked at the behaviour of the spring in the last two hook and spring clips, of the bubbles in the last two bubbles clips, and the middle portion of the cloth in the second and third sheet clips as repetitive vibrations. They then proceeded to define the amplitude of the vibration either by example or by performance.

Vibrational frequency specification. Continuing from above, participants would indicate the frequency of vibration either in the same gesture as the amplitude (performance-based) or indicate the frequency on a timeline (keyframe-based).

World rotation. This operation rotates the whole scene to look at it from other viewpoints.

World scaling. Similar to world rotation, participants wanted to scale the world to get a better viewpoint, typically to "zoom into" small details or to "zoom out" and get a bird's eye view of the scene.

7.2.5 Salient Observations and Discussion

The freedom afforded to participants and the think-aloud protocol allowed informal observation of high-level trends.

Expressiveness of gestures. Significant variation was observed between the number of DoFs controlled by a single gesture. While many gestures specified a single parameter such as particle emission speed, high-bandwidth gestures manipulating many spatiotemporal parameters simultaneously were also noticed. For example, describing the emission curve, speed, spread and randomness using a single gesture (Figure 7.1a–b). In between the two extremes, gestures described motion paths in space ($\mathbb{R}^3$) or space-time ($\mathbb{R}^4$), and 3D rotations (4 DoFs: axis and speed).

Expectation and control of physics. Almost all participants had an expectation of interactive physics simulations. In particular, users expected gravity, deformation and contact modelling, and they presumed their ability to directly apply linear and rotational impulses to objects. Participants also attempted to set physical and simulation parameters that would then modify the resulting simulated behaviours. For example, by changing the physical properties, e.g., mass or stiffness of the bouncing ball, or by changing simulation parameters, e.g., "glue strength" of the tearing edges in the sheet animations. Some gestures demonstrated the desired physical behaviour by both direct action and by providing key examples. For instance, P12 described the motion of the pieces in the shattering sheet animation by using their hands as claws to break the surface, followed by moving them along the desired path of the pieces to provide example motion. Finally, another common class of physics-based gestures directly demonstrated the desired motion by providing examples; for instance by moving one bubble in space and then expecting this to induce the demonstrated degree of randomness to all the bubbles (Figure 7.4j).

Use of traditional widgets. Animators frequently wanted to use traditional widgets to manipulate abstract and temporal parameters. While this can partly be attributed to habit or the difficulty of coming up with gestures for abstract features, animators also note that sliders and numeric inputs allow finer parameter control. P7, P9 also expressed the need for standard terminology for communicating with peers and interoperability with desktop software.

Diegetic interface elements. In immersive interfaces, the term *diegetic* describes an interface which exists "where the action is" [202]. A number of *demonstrative* interactions involved imaginary high-dimensional diegetic interface elements that animators manipulated gesturally. Typical examples include motion trails, onion skins, emitter-attached UI, force fields, and standard anchors for 3D manipulation.

Similar high-DoF diegetic widgets have been described, for example, by Conner et al. [53] for manipulating 3D objects and by Sheng et al. [216] for a sculpting interface utilizing physical props. Further, animators frequently desired traditional widgets to be *diegetically positioned*. For example, showing a *properties* menu directly above a selected object.

Gesture overlap. Some overlaps were observed between gestures elicited from different participants, as well as between different operations performed by the same participant. That is, participants would specify the same (or similar) gestures for different effects. Some of these ambiguities could potentially be resolved with carefully designed diegetic UIs. For example, P3 used a common "point at emitter and move finger up/down" gesture to specify emission density as well as frequency. Other overlaps could be reduced by contextual mode-switching, such as when P1 and P6 used a "double-tap" gesture for selection and bringing up a diegetic panel, respectively.

Comparison with existing studies on gestures. There are revealing similarities between these findings for mid-air gestures and those of earlier studies on touch and motion gestures. Wobbrock et al. [261] reported that users rarely cared about the number of fingers used for touch gestures, instead relying on the *form* of the gesture. My findings for mid-air gestures reinforce theirs. For instance, when directly manipulating an object, participants' hand pose either mimicked grabbing the object physically or was a canonical pose such as pointing with the index finger. A similar effect was noticed for some gestures with dynamic hand poses—participants would, for example, flutter their fingers to add noise to smoke, but only the quantity of fluttering was deemed important and not the exact spatial relationship between the fingers. My findings on gesture expressiveness follow Brouet et al.'s [35]. While participants did use gestures rich in spatiotemporal information, the number of DoFs in the intended effect was never close to the theoretical limit of 27 (54 for bimanual). The elicited bimanual gestures also fit neatly into the *bimanual-static* or *bimanual-symmetric* categories Brouet et al. defined for multi-touch. Another useful finding was that semaphoric and demonstrative gestures for logically opposite tasks tended to be geometrically mirrored—move hand left/right to scrub forward/back in time, move hand up/down to increase/decrease the value of a numeric parameter, bring the thumb and index finger together/move away to couple/decouple objects, etc. Similar findings have been reported for motion gestures [199] and touch gestures [261].

Comparisons can also be drawn with studies on mid-air gestures targeting other applications. Aigner et al. [7] elicited a number of pointing gestures for human-to-human interaction. In contrast, participants in my study generally chose to directly grab objects. Vatavu [249] suggests augmenting mid-air gestures for TV control with other modalities such as on-screen widgets or a traditional remote control. My findings also suggest that a *gesture-dominated* interface prudently utilizing traditional widgets and spatial UI can be more useful than a purely gestural interface. Lastly, owing to the ubiquity of touch devices, some gestures seemed to be inspired by touch gestures. For example, participants air-tapped for selection and used "pinch to zoom" to scaling. A similar observation was made by Troiano et al. [243] for gestures manipulating elastic displays.

7.3 Design Guidelines for Gestural Animation

I now consolidate my formal and informal observations into a set of design guidelines for VR animation interfaces that employ mid-air gestures. I hope these guidelines serve as a useful starting point for future experiments and the design of new immersive animation systems.

Breakdown of Interactions for Various Effects

The study suggests direct manipulation as the primary interaction mode for spatiotemporal operations (Figure 7.4a,d). Such interactions allow users to directly perform complex motions in spacetime. For operations that are either spatial or temporal (but not both), a combination of widget use, direct manipulation, and abstract demonstrative gestures may be appropriate. In addition, editable diegetic representations of spatiotemporal properties (e.g., editable motion trails) can be effective for visualizing and refining previously-authored motions.

Contextual Interaction Bandwidth

Most animators in the study adopted a coarse-to-fine workflow where they quickly specified a rough scene before delving into the details. To support this process, VR animation systems should provide high-bandwidth, direct manipulation gestures to quickly specify the overall spatiotemporal properties of an animation, and then allow layering of other, often finer, details with gestures that are scoped to control just the relevant, local, spatial and/or temporal properties. Moreover, context-dependent gestures can help reduce friction in such workflows. For example, beginning with a point-and-move gesture to perform the rough spatiotemporal path of an object, the same gesture can then be re-used for fine-grained control of the spatial positioning of the motion trail or small adjustments to the object's motion specified in its local coordinate frame. In each context, the system should infer which attributes of the gesture to use and ignore; e.g., when editing the motion trail, the execution speed of the gesture is not relevant.

Natural Interactions with Physics

Physical systems in VR animation utilize a wide range of physical parameters to obtain expressive simulations. However, my study indicates that animators generally do not want to consciously manipulate all such physical variables. Instead we see animators focusing individually on only particular, generally high-level, aspects of physical simulation output, with the expectation that the underlying simulation system will automatically make reasonable choices by "taking care" of the rest of the parameters. Moreover, many different types of interactions related to physics were observed, from direct (e.g., throwing an object as in Figure 7.4a) to indirect (e.g., changing noise, changing emission frequency as in Figure 7.4f). This suggests that VR animation systems should support real-world physical behaviours with reasonable default settings and that users should be able to interact with simulations in various ways while seeing real-time results. At the same time, it is important to provide expressive controls that allow animators to art-direct and exaggerate physics-based effects beyond the boundaries of realistic defaults when desired.

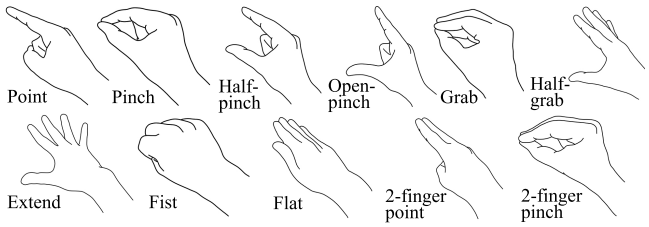


Figure 7.7: Typical hand poses observed in the study.

Hand Pose Granularity and Gesture Recognition

A practical observation from the study is that most of the gestures did not encode information in the nuances of hand shape. Participants were largely ambivalent towards the hand poses they used for a large number of gestures. Further, most of the observed gestures utilized only a few important poses (Figure 7.7), that can be easily distinguished. Thus, we observe that users are both forgiving of high-frequency noise in pose recognition and thus possibly cheaper, unsophisticated, hardware and coarse recognition can potentially serve as a good starting point for gestural interfaces.

Also note that participants were willing to learn a few key poses to disambiguate between commonly used commands such as manipulating pose vs. performance. In turn this suggests flexibility in the choice of poses assigned to operations, which can be exploited to maximize recognition accuracy.

Controller-based Interfaces

Finally, while the experiment was conducted to understand bare-handed gestures, some findings can potentially benefit controller-based interaction as well. Prior research has shown that grip can be leveraged as an implicit dimension conveying some user intention as they interact with physical devices [221, 233, 260]. More recently, multi-purpose haptic rendering [47] controllers demonstrate how hand poses can be utilized to define gestures similar to bare-handed gestures. Beyond static postures, future controller designs can also potentially leverage dynamic poses and high-bandwidth gestures to simulate the expressiveness of bare-handed input, while retaining the benefits of haptic controllers. For example, participants defined the amount of smoke dispersion by interpolating between the "pinch" and "extend" poses (Figure 7.7). Controllers detecting a continuous pose change via a proxy measure can enjoy an input effectively as rich as bare-hands for such interactions. Given current limitations in hand-gesture tracking technologies [273], I explore this guideline further in the next section.

7.4 MagicalHands: Design and Implementation

To investigate the above design guidelines, I developed a prototype VR animation system—MagicalHands. Similar to the SymbiosisSketch system (Chapter 4) which was built to assess the design guidelines developed in Chapter 3, I use MagicalHands to explore concrete instantiations of the guidelines and to assess the feasibility and usability of a subset of the novel gestures observed in the elicitation study. In this proof-of-concept tool, I focus on the creation and manipulation of particle systems. Based on the formative study, I implemented eleven interactions, including eight gestures (from direct manipulation to abstract demonstrations) and three abstract operations utilizing 3D UI.

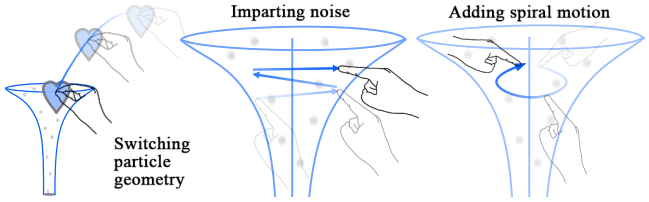


Figure 7.8: Implemented gestures for particle system manipulation.

Figure 7.9: Touch-sensitive buttons and triggers on the Oculus Touch controller can be mapped to hand poses. For example, the absence of touch on the front trigger (a, red arrow) is interpreted as the index finger being extended (b), while its presence is interpreted as the finger being bent (c).

7.4.1 Interaction Design Concepts

I first describe the UI concepts utilized by the system.

Visual Entities. Suitable for an immersive setting, I built several types of visual entities, including scene objects (e.g., bubbles), animation primitives (e.g., particle emitter), and UI elements (e.g., playback control buttons). As suggested by the design guideline on breakdown of interactions, explicit diegetic realizations of hand gestures and performances could also be used as visual entities for editing, though these were not included in the prototype. Access to objects for animation is provided via an *object shelf* positioned at a fixed location in the immersive world. The shelf contains animation-ready 3D meshes imported into the system and special *emitter* objects which can be used to spawn particle systems.

Extrinsic and Intrinsic Attributes. To avoid visual clutter, *extrinsic* entities (e.g., 3D models, shelf, UI controls) have persistent visual affordances, and can be directly accessed using gestures for common animation operations. *Intrinsic* entities (e.g., emission curve, spread, and noise) are made visible and operable only in context, by gesturally selecting the pertinent object first. Direct manipulation of an extrinsic attribute in MagicalHands is naturally invoked by gesturally grabbing its visible affordance. An explicit selection gestures can be used to select objects and enable intrinsic attribute manipulation.

Creation Process Freedom. Animation of object attributes in an interactive setting is typically authored using keyframes (for smooth low frequency changes); human performance (for high frequency detail and timing); and simulation (for high-DoF changes governed by the laws of physics). My implementation supports all three, driven off an animation control widget with familiar UI and gesturally-activated extrinsic play/pause/record/scrub attributes.

Eulerian and Lagrangian Gestures. In simulation, *Lagrangian* specification directly defines an object's attribute in time, while *Eulerian* specification describes the properties of the ambient space that an object exists in. In MagicalHands, the rigid motion of an object is specified in the former manner by grabbing it and performing/keyframing its attributes. Additionally, position and orientation are ani-

mated separately, as cognitive and anatomic limitations make it difficult to perform arbitrarily complex motion trajectories. Gestures impacting *intrinsic* attributes such as emission properties impact particles in an *Eulerian* manner, influencing the space around an emitter rather than directly manipulating the particles.

Figure 7.10: Stills from animations authored using MagicalHands, along with execution time (asset arrangement + animating + experimentation) for each clip. Author creations (a, b, e, i), and participant creations (c, d, f, g, h). Please see the online video at https://youtu.be/YOJVYCQeXFs for the animated scenes. Man typing on computer (a) model ©Chuantao Xu; used with permission.

7.4.2 User Interaction

I now describe how users interact with the above concepts.

Instantiation and Deletion. Dragging an object from the shelf—by pointing at it, pinching it, or grabbing it—and then dropping it into the scene instantiates a copy of that object. Deletion involves performing this gesture in reverse.

Selection. Users can *tap* objects (Figure 7.4h) in the scene to select. For particle systems, this visualizes the current emission curve and spread, and enables emitter-specific interactions.

Direct Manipulation for Natural Interactions. I implemented three direct manipulation performance gestures to record a motion. Based on the design guidelines, direct performance of a motion path is supported (Figure 7.4a). The user pinches onto an object with the dominant hand and then moves it to **translate** the object over time. Pinching with both hands and moving the hands closer together or further apart (Figure 7.4b[1]) **scales** the object uniformly. Finally, using the dominant hand in a pointing pose defines an axis of rotation and the motion of the non-dominant hand around this axis (Figure 7.4c[1]) **rotates** the target object. Recall that the study suggests implicit selection for manipulating such *extrinsic* properties, and thus, an explicit selection gesture is not required for direct manipulation.

High-Bandwidth & Demonstrative Gestures for Particles. Instantiating an emitter object creates a particle system. As indicated by the design guideline on physical simulations, this interaction automatically produces an animated result with default particle geometry, and default parameters for emission curve, speed, and spread. In addition, the simulation provides real-time feedback and allows users to modify it with various interactions. For example, they can move the emitter itself and drag and drop objects from the shelf into the emitter to **attach particle to emitter** (Figure 7.8). Selecting an emitter visualizes the current emission curve and volume, and enables specialized, context-dependent gestures.

— A *high-bandwidth gesture* describes the **emission curve, speed, and spread** intrinsics in a single action. Users position both hands close to an emitter in fist pose and move away to perform the gesture (Figure 7.4d[1]). The curve swept out by the mid-point of the imaginary line joining the hands defines the emission curve and speed, while the distance between them defines the emission spread along the curve.

— Users can perform an abstract demonstration by moving their hand inside the emission volume in a pinch, point, or fist shape to **impart noise** (Figure 7.8). Performing a **spiral motion** makes the particles swirl around the emission curve. Moving the hand along a single dominant axis activates the noise gesture, while moving in a spiral activates the spiral force gesture. This inference is automatic; see Section 7.4.3 for details.

Egocentric Interface for Navigation. A UI panel is positioned close to the non-dominant hand of the user, with buttons for **playing or pausing** the animation clip, and another for **toggling record on or off**. When recording is ON, the direct manipulation gestures are treated as spatiotemporal *performances* and the speed of the gesture is transferred onto the target object. In contrast, when recording is OFF, these gestures serve as *posing* tools—defining a keyframe at the current frame. The panel also contains timeline scrubbing buttons.

7.4.3 Setup and Implementation Details

Implementation of the gestures requires robust, real-time 3D hand recognition capabilities. Fortunately, the design guidelines suggest that most gestures utilize only a few important poses. Thus, mindful of current hand recognition technology limitations, I use the hand-held Oculus Touch controllers. They

provide larger tracking volume (compared to the Leap Motion, for example) and reliable hand-pose information which allows testing the guidelines in this setup—sacrificing some gesture recognition capability for robustness. Note that gestures which require continuous pose tracking and/or unconventional hand poses are rendered infeasible by this setup. This reduces the possible set of gestures that could be implemented in the system.

My implementation utilizes a layer of abstraction over the raw tracking data so that the inference system only uses hand pose information from the controller (Figure 7.9). Practically, this means that the system can be easily adapted to run with hand-pose data generated by bare-handed tracking techniques, motion-captured gloves, or other lightweight sensors.

The position of the fingertips w.r.t the wrist and the distance between them is used to infer hand poses among point, pinch, fist, and extended (neutral) poses (Figure 7.7). 6-DoF hand tracking data combined with this inferred pose is utilized for gesture recognition. For direct manipulation gestures, transformations are applied to the objects in their local space, which allows chaining together different objects by parenting. In the current prototype, parental hierarchy has to be manually defined while importing objects. Animations are played back to the users in a loop, and the duration of the clip is simply determined by the performed gestures: the clip initially consists of a single frame, and performing manipulations whose length goes past the clip duration automatically adds frames to the clip to record the entire gesture. The translation, rotation, and scaling gestures define a dense set of keyframes when used as performance (record ON), and a sparse set when used to pose (record OFF). The system was implemented in C# using Unity.

Particle System Implementation

The particle system implementation builds upon Unity's native implementation [247], and adds direct control to the particles. The emission curve $\mathbf{c}(s) : [0,1] \to \mathbb{R}^3$ is created by fitting a C^1-continuous piecewise cubic Bézier spline to the input points [212]. The particle lifetime $t(s) \in \mathbb{R}^+$ and spread $r(s) \in \mathbb{R}^+$ are stored as piecewise linear functions on the curve. Since, $t(s)$ is monotonic, $\mathbf{c}$ and r can equivalently be thought as functions on t. Particle are generated at a random position on the plane normal to $\mathbf{c}(0)$, given by polar coordinates $(\delta, \theta_0) \in ([0,1] \times (0, 2\pi])$ w.r.t the Frenet frame of the curve at $s = 0$. The emission noise control is realized by modulating the particle's position and orientation with a Perlin noise [182] function, whose scale and frequency are gesturally controlled. Finally, a "spiral velocity" can be applied by rotating the particles in the cross-section plane with a speed supplied by the gesture. For a particle born at time t_0, its position at time $t_0 + t$ is given by

$$\mathbf{p}(t) = \mathbf{c}(t) + \delta r(t) \left(\mathbf{N}(t) \cos\left(\theta(t)\right) + \mathbf{B}(t) \sin\left(\theta(t)\right)\right) + \alpha \mathcal{N}(\nu t). \tag{7.2}$$

Here, $\theta(t) = \theta_0 + \omega t$, where ω is the spiral force parameter, and $\mathcal{N}$ is a 2D Perlin noise function modulated by an amplitude parameter α and a frequency (or scale) parameter ν. The noise function is applied in the **N**-**B** plane. A similar noise function applied to the particle orientation rotates it about **N** and **B**.

Noise and Spiral Motion Gestures

Since the user uses the same hand pose for spiral motion and noise gestures, we need to automatically distinguish between the two. To achieve this, I utilize the time-sampled sequence of 3D hand positions

$\{\mathbf{x}_i\}$ and accelerations $\{\ddot{\mathbf{x}}_i\}$ from the gesture. First, the nearest point $\mathbf{c}(t_i)$ on $\mathbf{c}$ for each of the sampled positions $\mathbf{x}_i$ is computed. Then, the acceleration $\ddot{\mathbf{x}}_i$ is transformed to the Frenet frame at $\mathbf{c}(t_i)$. In the third step, a Singular Value Decomposition (SVD) on the transformed acceleration data sequence is computed. The singular values represent the variation in the data along its principal components. For the noise gesture, the acceleration vectors are expected to be clustered along a single axis, while spiral motion gestures should exhibit significant acceleration in two directions. Thus, if the ratio of the first (largest) and second singular values is larger than a threshold (this was set to 2.0), the data is inferred as a noise gesture. Otherwise, the spiral force command is executed.

For the former, a Fast-Fourier Transform (FFT) gives the dominant frequency and its associated amplitude, which directly map to the parameters α and ν of our procedural noise. For the latter, the ratio of the average speed and position projected to the **N-B** planes is enough to define the angular velocity ω.

7.4.4 Testing and Results

Four artists were invited to try the MagicalHands system in an informal setting. P1 had formal training in animation, P2–3 had amateur experience with desktop-based animation tools, and P4 was a professional industrial designer, with 5 years of experience with VR-based 3D design and modelling. However, none had experience using VR for animation tasks.

Users were given a short tutorial explaining how they can utilize hand gestures for various tasks. They could then freely explore the system and create animations. Users had access to 35 static objects in the shelf to use in their creations, including three particle emitters with different preset emission densities. Figures 7.1 and 7.10 show stills from the clips created by the participants, as well as those created by the author of this work. Depending on scene complexity, creating these clips took between 5 and 40 minutes, excluding planning time.

User Feedback. In general, users really liked the the use of VR and live performance aspects of the system. P2 found that "just placing objects [in VR] is already interesting…being able to animate directly is really cool…" . The ability to directly control particle systems was also appreciated. All the users found gestural manipulation of particle systems useful. P2 found it "extremely fun to work with" , while P1, who had experience with Maya [21], found the "gestures in VR make a lot of sense since drawing 3D splines on a 2D screen is really hard" . More generally, they thought that "drawing the space curve live was useful for timing the particle emission and [for] translations" .

P4 had insights about interacting with higher-level abstractions of gestures: "I need to see my gesture materialize… it should be like a sculpture…" . While other gestures were appreciated, users found the rotation gesture a bit mentally taxing. P3 commented that "being able to define a particular axis [of rotation] is useful" but alternatives such as "being able to use one hand as a pivot point and other to [implicitly] define the axis" could provide more intuitive control. Still, users successfully utilized the gesture, and their creations incorporated both performed (P1) and keyframed (P3) rotations.

7.5 Conclusion and Future Work

This chapter expands the body of HCI research on hand gestures to animation authoring. From the elicitation study, I identified and categorized a broad spectrum of gestural interactions that can be

used to depict complex dynamic effects in VR. In addition to gestures, my observations suggest that traditional widgets and spatial UI elements are useful for fine control over specific parameters. Based on these findings, I derived a set of design guidelines to inform the design of gesture-based VR animation authoring tools, and developed a prototype system to test the guidelines and observed gestures. The user study demonstrates the effectiveness and potential of hand gestures to author animation in VR.

Looking forward, this work points at several interesting directions for future research. While my study describes a range of potential gestures for animation, it would be valuable to conduct additional experimental work on the usability of such gestures. Another interesting research direction is the evaluation of mid-air gesture taxonomies, including the one presented here, to identify gaps and compare different classification strategies, similar to the recent work of Kim et al. [135] on shape-changing interfaces. In parallel, I hope these design guidelines inspire system builders to develop novel gesture-based VR animation tools; and that the MagicalHands system can help in these future efforts. Such efforts would likely require new gesture tracking and recognition algorithms and novel user interfaces that combine mid-air gestures with diegetic and traditional widgets. In this vein, one specific sub-area to investigate is gestural control of physical simulations. Here, a key challenge is co-developing robust and controllable simulation methods that predictably respond to gestures. In order to facilitate further development of such experimental work, I have released our source code at https://github.com/rarora7777/MagicalHands.

Finally, although this work focuses on animating physical phenomena, some of my findings may apply more broadly to other dynamic effects, such as lighting changes and mechanical movements. Moreover, some gestures could translate to non-VR animation tasks provided that users have effective ways to target gestures to desired objects. Validating these ideas is an exciting future work direction that would further expand the scope of gestural interactions for content creation.

Another potential limitation relates to the criteria used for developing gesture sets. In the presented study, the gestures were derived entirely via elicitation, which emphasizes *intuitiveness*, and to some extent, *discoverability*, but ignores other important aspects of a successful gesture set. In particular, physical factors such as the complexity of executing a gesture, or the ergonomics of the gesture, were not fully considered. The study also did not emphasize computation-related factors such as the ease of developing a gesture recognizer and recognition accuracy. I hope that in the future, the large gesture set elicited in this work can form the basis of successful gesture sets that take physical and computational factors fully into account.

Chapter 8

Conclusion

With rapidly advancing tracking and display technologies and constantly improving comfort levels, VR and AR have the potential to fundamentally transform 3D creation. The aim of this work is to help realize that potential. In this thesis, I touched on various problems in immersive sketching, geometric modelling, and animation. However, the progress we've made so far is just the beginning. Important questions about broad topics such as immersive ideation and character animation remain unanswered, which will require further fundamental studies along with algorithmic developments. In addition, immersive creation has transformative potential for other creative domains, such as visual effects, filmmaking, and digital fabrication. My hope is that this work inspires further research in immersive creation—interdisciplinary work involving HCI, graphics, design, and vision. With burgeoning consumer interest as well as unanswered scientific questions, immersive creation promises to be an active field of research for many years to come.

8.1 Identifying and Overcoming the Human Factors Challenges

A consistent theme through this thesis is the advocacy for a combination of graphics and HCI techniques to design immersive creation tools. Conducting surveys, interviews, and quantitative studies gives the community of current and future users a voice in shaping the future of immersion. While the importance of representation itself cannot be overstated, such user studies help identify the challenges faced by the human users of immersive technologies. In the presented work, I identified challenges related to visual perception (Section 3.1), precision (Section 3.2), musculoskeletal limitations (Section 5.1.3), the feeling of creative freedom (Section 6.4), and the difficulties in manipulating complex systems intuitively (Section 7.2). Identifying these challenges helped define concrete design guidelines for immersive creation and devise new interactions and algorithms to overcome the challenges. For instance, identifying the challenge in drawing curves onto surfaces while simultaneously controlling all six degrees of freedom of a VR controller led to my new method which significantly cut down on the user effort. I believe that this is a fruitful strategy for future work on immersive creation. For most artists and designers today, immersive creation represents a paradigm shift in the way they create. With the novelty this paradigm shift provides, we can expect reduced rigidity from users in adopting unfamiliar interaction mechanisms. This novelty can and should be exploited by researchers and application developers to design interactions that improve the user's experience, but may be drastically different from familiar 2D tools. In-depth

user studies provide an avenue for identifying the correct path towards such disruption.

8.2 Learning from the 2D World

"If I have seen further it is by standing on the shoulders of Giants."

—*Isaac Newton*

When designing immersive creation tools, the plethora of knowledge developed for 2D input and output devices is truly the giant whose shoulders we stand on. In the previous section, I encouraged developing disruptive new techniques even if they are completely unfamiliar to the users of desktop tools. But familiarity breeds user confidence and familiar techniques should therefore only be discarded if they truly encumber the user's creative expression. As an example, in Chapter 4, I combined 2D sketching with 3D sketching to improve the user experience as compared to a purely 3D interface. In Section 7.4, I developed an animation tool which combines gestural performance with a familiar keyframe-based system. A lot of the research presented here builds on the incredible work in sketch-based modelling, performance-based animation, and evaluation methodologies for 2D interactions. As we have seen from the various challenges faced by immersive tools, the 3D interaction world requires similar ideas as those utilized for 2D interfaces for creating three-dimensional artifacts. We still need interpretative algorithms to correct user inputs, visualization techniques to aid shape perception, and interaction techniques for interacting with subspaces of the 3D space. I expect a rich exchange of ideas between immersive creation research and desktop-based tools in the future.

8.3 Novel Aesthetics from and for the Immersive Medium

In addition to the impact of 3D interactions on existing design pipelines, immersive creation opens up exciting possibilities for developing novel aesthetics for art and design. In Chapter 4, we saw how a combination of 2D and 3D sketching led to a novel aesthetic for 3D sketching—sketches that include precise details and rich textures as well as organic surfaces that take advantage of the 3D interaction space. In the commercial space, many VR experiences have adopted minimalist rendering and animation styles not just for computational efficiency, but to avoid the uncanny valley effect of almost-realistic immersive virtual worlds [213]. Even the constraints imposed by commercial freehand stroke–based tools such as Tilt Brush [87] and Quill [71] have led to a new design language: surfaces created using a dense set of closely overlaid strokes. Interesting questions about adapting non-photorealistic rendering techniques to the immersive world have also been opened up by the interplay between 3D geometry and a user-controlled camera. For instance, consider a simple toon shader with a flat colour and an outline. Should the outline always stay at the silhouette and depth discontinuities as seen from the current point of view? Or should the contours follow sharp geometric features? Creations meant to be consumed in Augmented Reality throw up even more intriguing questions. How and how much should a virtual work of art adapt to the surrounding environment? Going beyond the visual look and feel, adapting shapes and motion to the viewer's environment also presents exciting open problems.

8.4 The Tool Dictates the Outcome

When developing creativity support tools, our goal is a low barrier to entry, a high ceiling in terms of the complexity of the artifacts enabled by the tool, along with support for diverse creation strategies and outcomes. A perfect tool should simply mediate between the creator's mental idea and the final creation. Given the perfect tool then, the user could externalize the precise idea that exists in their mind. Unfortunately, this Utopian ideal never plays out in practice, neither with digital tools nor physical. Creation is an imperfect process and the resulting creative artifact is impacted by the process itself—creative outcomes are biased by the capabilities of the tool, the skills of the creator, and the creator's familiarity with the tool.

Naturally, the tools described here suffer from the above limitations. All the choices made in the tool design, from the input devices to the geometric representations, impose a constraint on the user's ability to create. For instance, while the choice of a pen in SymbiosisSketch (Chapter 4) makes the interface intuitive due to the familiar form factor, a standard VR controller equipped with multiple buttons could have helped users switch between colours and tools easily, potentially leading to more visually vibrant creations. In the CASSIE project (Chapter 6), the choice of a poly-Bézier spline for curve fitting makes the curve network solver efficient, enabling it to run at an interactive rate. While this likely contributed to the users' feeling of unencumbered creativity, the fitting procedure could lead to the loss of high-frequency detail. The use of hand gestures in MagicalHands (Chapter 7) aids expressiveness and allows an animator to create a complex animation quickly; but at the same time, it cannot provide the precise spatial or temporal control enabled by keyframing motion in traditional software.

In summary, every choice made when building a novel creative tool biases its creative outcomes. Some choices place crystal clear constraints on the achievable results, while others manifest in subtle ways, biasing the creator's process towards certain classes of artifacts. These choices must be weighted carefully when planning, building, and testing creativity support systems.

Bibliography

[1] Fatemeh Abbasinejad, Pushkar Joshi, and Nina Amenta. 2011. Surface Patches from Unorganized Space Curves. *Computer Graphics Forum* 30, 5 (Aug. 2011), 1379–1387. https://doi.org/10.1111/j.1467-8659.2011.02012.x

[2] William Abend, Emilio Bizzi, and Pietro Morasso. 1982. Human Arm Trajectory Formation. *Brain* 105, 2 (1982), 331–348. https://doi.org/10.1093/brain/105.2.331

[3] Adobe. 2021. Character Animator. https://adobe.com/products/character-animator.html.

[4] Adobe. 2021. Medium by Adobe. https://www.adobe.com/products/medium.html.

[5] Adobe. 2021. Substance Painter. https://www.substance3d.com/substance-painter/.

[6] Anne M.R. Agur and Ming J. Lee. 1999. *Grant's Atlas of Anatomy* (10 ed.). Williams and Wilkins.

[7] Roland Aigner, Daniel Wigdor, Hrvoje Benko, Michael Haller, David Lindbauer, Alexandra Ion, Shengdong Zhao, and Jeffrey Tzu Kwan Valino Koh. 2012. *Understanding Mid-Air Hand Gestures: A Study of Human Preferences in Usage of Gesture Types for HCI.* Technical Report MSR-TR-2012-111. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/GesturesTR-20121107-RoA.pdf

[8] Panu Åkerman, Arto Puikkonen, Pertti Huuskonen, Antti Virolainen, and Jonna Häkkilä. 2010. Sketching with strangers: in the wild study of ad hoc social communication by drawing. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*. 193–202. https://doi.org/10.1145/1864349.1864390

[9] David Akers. 2006. Wizard of Oz for Participatory Design: Inventing a Gestural Interface for 3D Selection of Neural Pathway Estimates. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems* (Montreal, Quebec, Canada) *(CHI EA '06)*. ACM, New York, NY, USA, 454–459. https://doi.org/10.1145/1125451.1125552

[10] Pär-Anders Albinsson and Shumin Zhai. 2003. High precision touch screen interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 105–112. https://doi.org/10.1145/642611.642631

[11] George J. Andersen, Myron L. Braunstein, and Asad Saidpour. 1998. The perception of depth and slant from texture in three-dimensional scenes. *Perception* 27, 9 (1998), 1087–1106. http://pec.sagepub.com/content/27/9/1087.short

[12] Alexis Andre and Suguru Saito. 2011. Single-View Sketch Based Modeling. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling* (Vancouver, British Columbia, Canada) *(SBIM '11)*. Association for Computing Machinery, New York, NY, USA, 133—140. https://doi.org/10.1145/2021164.2021189

[13] Tom M. Apostol and Mamikon A. Mnatsakanian. 2007. Unwrapping curves from cylinders and cones. *Amer. Math. Monthly* 114, 5 (2007), 388–416. http://www.ingentaconnect.com/content/maa/amm/2007/00000114/00000005/art00002

[14] Rahul Arora, Darolia Ishan, Vinay P. Namboodiri, Karan Singh, and Adrien Bousseau. 2017. SketchSoup: Exploratory Ideation using Design Sketches. *Computer Graphics Forum* (2017). https://doi.org/10.1111/cgf.13081

[15] Rahul Arora, Alec Jacobson, Timothy R. Langlois, Yijiang Huang, Caitlin Mueller, Wojciech Matusik, Ariel Shamir, Karan Singh, and David I.W. Levin. 2019. Volumetric Michell Trusses for Parametric Design & Fabrication. In *Proceedings of the 3rd ACM Symposium on Computation Fabrication* (Pittsburgh, PA, USA) *(SCF '19)*. ACM, New York, NY, USA, 13. https://doi.org/10.1145/3328939.3328999

[16] Rahul Arora, Rubaiat Habib Kazi, Fraser Anderson, Tovi Grossman, Karan Singh, and George Fitzmaurice. 2017. Experimental Evaluation of Sketching on Surfaces in VR. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 5643–5654. https://doi.org/10.1145/3025453.3025474

[17] Rahul Arora, Rubaiat Habib Kazi, Tovi Grossman, George Fitzmaurice, and Karan Singh. 2018. SymbiosisSketch: Combining 2D & 3D Sketching for Designing Detailed 3D Objects in Situ. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal, Quebec, Canada) *(CHI '18)*. ACM, New York, NY, USA, 15. https://doi.org/10.1145/3173574.3173759

[18] Rahul Arora, Rubaiat Habib Kazi, Danny Kaufman, Wilmot Li, and Karan Singh. 2019. MagicalHands: Mid-Air Hand Gestures for Animating in VR. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. ACM, New York, NY, USA, 12. https://doi.org/10.1145/3332165.3347942

[19] Rahul Arora and Karan Singh. 2021. Mid-Air Drawing of Curves on 3D Surfaces in Virtual Reality. *ACM Trans. Graph.* 40, 3 (July 2021), 17. https://doi.org/10.1145/1122445.1122456

[20] Atlas V. 2020. Goodbye Mr. Octopus. https://atlasv.io/projects/goodbye-mr-octopus/.

[21] Autodesk. 2021. Maya. https://autodesk.com/products/maya/features.

[22] Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3D curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. ACM, 151–160. https://doi.org/10.1145/1449715.1449740

[23] Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2009. EverybodyLovesSketch: 3D Sketching for a Broader Audience. In *Proceedings of the 22nd Annual ACM Symposium on User*

Interface Software and Technology (UIST '09). ACM, New York, NY, USA, 59–68. https://doi.org/10.1145/1622176.1622189

[24] Ronald Michael Baecker. 1969. *Interactive Computer-mediated Animation*. Technical Report. Massachusetts Institute of Technology Cambridge Project MAC.

[25] Yunfei Bai, Danny M. Kaufman, C. Karen Liu, and Jovan Popović. 2016. Artist-directed Dynamics for 2D Animation. *ACM Trans. Graph.* 35, 4, Article 145 (July 2016), 10 pages. https://doi.org/10.1145/2897824.2925884

[26] Alethea Bair and Donald House. 2007. Grid With a View: Optimal Texturing for Perception of Layered Surface Shape. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (Nov. 2007), 1656–1663. https://doi.org/10.1109/TVCG.2007.70559

[27] Ravin Balakrishnan, George Fitzmaurice, Gordon Kurtenbach, and William Buxton. 1999. Digital tape drawing. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*. ACM, New York, NY, USA, 161–169. https://doi.org/10.1145/320719.322598

[28] Gavin Barill, Neil G. Dickson, Ryan Schmidt, David I. W. Levin, and Alec Jacobson. 2018. Fast Winding Numbers for Soups and Clouds. *ACM Trans. Graph.* 37, 4, Article 43 (July 2018), 12 pages. https://doi.org/10.1145/3197517.3201337

[29] Katie Bassett, Ilya Baran, Johannes Schmid, Markus Gross, and Robert W. Sumner. 2013. Authoring and Animating Painterly Characters. *ACM Trans. Graph.* 32, 5, Article 156 (Oct. 2013), 12 pages. https://doi.org/10.1145/2484238

[30] Anil Ufuk Batmaz, Aunnoy K Mutasim, and Wolfgang Stuerzlinger. 2020. Precision vs. power grip: A comparison of pen grip styles for selection in virtual reality. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, 23–28. https://doi.org/10.1109/VRW50115.2020.00012

[31] Miklós Bergou, Saurabh Mathur, Max Wardetzky, and Eitan Grinspun. 2007. TRACKS: Toward Directable Thin Shells. *ACM Trans. Graph.* 26, 3, Article 50 (July 2007). https://doi.org/10.1145/1276377.1276439

[32] Mikhail Bessmeltsev, Caoyu Wang, Alla Sheffer, and Karan Singh. 2012. Design-driven Quadrangulation of Closed 3D Curves. *ACM Trans. Graph.* 31, 6, Article 178 (Nov. 2012), 11 pages. https://doi.org/10.1145/2366145.2366197

[33] Richard L. Bishop. 1975. There is More than One Way to Frame a Curve. *The American Mathematical Monthly* 82, 3 (March 1975), 246. https://doi.org/10.2307/2319846

[34] Barbara Bradley. 2003. *Drawing People* (1st ed.). North Light Books, Cincinnati, Ohio.

[35] Rémi Brouet, Renaud Blanch, and Marie-Paule Cani. 2013. Understanding Hand Degrees of Freedom and Natural Gestures for 3D Interaction on Tabletop. In *Human-Computer Interaction – INTERACT 2013*. Springer Berlin Heidelberg, Berlin, Heidelberg. https://link.springer.com/chapter/10.1007/978-3-642-40483-2_20

[36] Bill Buxton. 2010. *Sketching user experiences: getting the design right and the right design*. Morgan kaufmann.

[37] Brian Cabral and Leith Casey Leedom. 1993. Imaging Vector Fields Using Line Integral Convolution. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (Anaheim, CA) *(SIGGRAPH '93)*. Association for Computing Machinery, New York, NY, USA, 263–270. https://doi.org/10.1145/166117.166151

[38] Xiang Cao and Shumin Zhai. 2007. Modeling Human Performance of Pen Stroke Gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 1495–1504. https://doi.org/10.1145/1240624.1240850

[39] Jiawen Chen, Shahram Izadi, and Andrew Fitzgibbon. 2012. KinÊtre: Animating the World with the Human Body. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. ACM, New York, NY, USA, 435–444. https://doi.org/10.1145/2380116.2380171

[40] Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu, and Daniel Cohen-Or. 2013. 3-Sweep: Extracting Editable Objects from a Single Photo. *ACM Trans. Graph.* 32, 6, Article 195 (Nov. 2013), 10 pages. https://doi.org/10.1145/2508363.2508378

[41] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. 2009. A Benchmark for 3D Mesh Segmentation. *ACM Trans. Graph.* 28, 3, Article 73 (July 2009), 12 pages. https://doi.org/10.1145/1531326.1531379

[42] Xuejin Chen, Sing Bing Kang, Ying-Qing Xu, Julie Dorsey, and Heung-Yeung Shum. 2008. Sketching Reality: Realistic Interpretation of Architectural Designs. *ACM Trans. Graph.* 27, 2 (May 2008), 11:1–11:15. https://doi.org/10.1145/1356682.1356684

[43] Xiang Anthony Chen, Jeeeun Kim, Jennifer Mankoff, Tovi Grossman, Stelian Coros, and Scott E. Hudson. 2016. Reprise: A Design Tool for Specifying, Generating, and Customizing 3D Printable Adaptations on Everyday Objects. ACM Press, 29–39. https://doi.org/10.1145/2984511.2984512

[44] Kelvin Cheng and Masahiro Takatsuk. 2009. Interaction Paradigms for Bare-Hand Interaction with Large Displays at a Distance. In *Human-Computer Interaction*, Inaki Maurtua (Ed.). InTech, 195–222.

[45] Siu-Wing Cheng, Tamal Krishna Dey, and Jonathan Richard Shewchuk. 2013. *Algorithms for constructing Delaunay triangulations*. CRC Press / Taylor & Francis, Boca Raton, Fla. OCLC: 844619099.

[46] Erin Cherry and Celine Latulipe. 2014. Quantifying the Creativity Support of Digital Tools through the Creativity Support Index. *ACM Trans. Comput.-Hum. Interact.* 21, 4, Article 21 (June 2014), 25 pages. https://doi.org/10.1145/2617588

[47] Inrak Choi, Eyal Ofek, Hrvoje Benko, Mike Sinclair, and Christian Holz. 2018. CLAW: A Multifunctional Handheld Haptic Controller for Grasping, Touching, and Triggering in Virtual Reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 654. https://doi.org/10.1145/3173574.3174228

[48] Paolo Cignoni, Massimiliano Corsini, and Guido Ranzuglia. 2008. MeshLab: an Open-Source 3D Mesh Processing System. *ERCIM News* 2008, 73 (2008), 47–48. http://dblp.uni-trier.de/db/journals/ercim/ercim2008.html#CignoniCR08

[49] Zeynep Cipiloglu, Abdullah Bulbul, and Tolga Capin. 2010. A framework for enhancing depth perception in computer graphics. ACM Press, 141. https://doi.org/10.1145/1836248.1836276

[50] Viviane Clay, Peter König, and Sabine Koenig. 2019. Eye tracking in virtual reality. *Journal of Eye Movement Research* 12, 1 (2019). https://doi.org/10.16910/jemr.12.1.3

[51] Dale J. Cohen and Susan Bennett. 1997. Why can't most people draw what they see? *Journal of Experimental Psychology. Human Perception and Performance* 23, 3 (June 1997), 609–621. https://doi.org/10.1037/0096-1523.23.3.609

[52] Patrick Coleman and Karan Singh. 2006. Cords: Geometric Curve Primitives for Modeling Contact. *IEEE Computer Graphics and Applications* 26, 3 (2006), 72–79. https://doi.org/10.1109/MCG.2006.54

[53] Brookshire D. Conner, Scott S. Snibbe, Kenneth P. Herndon, Daniel C. Robbins, Robert C. Zeleznik, and Andries van Dam. 1992. Three-dimensional Widgets. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics* (Cambridge, Massachusetts, USA) *(I3D '92)*. ACM, New York, NY, USA. https://doi.org/10.1145/147156.147199

[54] Intel Corporation. 2009. *Intel Math Kernel Library. Reference Manual*. Intel Corporation.

[55] Michael AA Cox and Trevor F Cox. 2008. Multidimensional scaling. In *Handbook of data visualization*. Springer, New York, NY, USA, 315–347. https://doi.org/10.1007/978-3-540-33037-0_14

[56] Richard C Davis, Brien Colwell, and James A Landay. 2008. K-sketch: a 'kinetic' sketch pad for novice animators. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 413–422. https://doi.org/10.1145/1357054.1357122

[57] Chris De Paoli and Karan Singh. 2015. SecondSkin: Sketch-Based Construction of Layered 3D Models. *ACM Trans. Graph.* 34, 4, Article 126 (July 2015), 10 pages. https://doi.org/10.1145/2766948

[58] Michael F. Deering. 1995. HoloSketch: A Virtual Reality Sketching/Animation Tool. *ACM Trans. Comput.-Hum. Interact.* 2, 3 (Sept. 1995), 220–238. https://doi.org/10.1145/210079.210087

[59] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, 317–324. https://doi.org/10.1145/311535.311576

[60] Ciro Donalek, S George Djorgovski, Alex Cioc, Anwell Wang, Jerry Zhang, Elizabeth Lawler, Stacy Yeh, Ashish Mahabal, Matthew Graham, Andrew Drake, et al. 2014. Immersive and collaborative data visualization using virtual reality platforms. In *2014 IEEE International Conference on Big Data (Big Data)*. 609–614. https://doi.org/10.1109/BigData.2014.7004282

[61] Gianluca Donato and Serge Belongie. 2002. Approximate Thin Plate Spline Mappings. In *Computer Vision — ECCV 2002*, Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen (Eds.). Vol. 2352. Springer Berlin Heidelberg, Berlin, Heidelberg, 21–31. http://link.springer.com/10.1007/3-540-47977-5_2 DOI: 10.1007/3-540-47977-5_2.

[62] Mira Dontcheva, Gary Yngve, and Zoran Popović. 2003. Layered Acting for Character Animation. *ACM Trans. Graph.* 22, 3 (July 2003), 409–416. https://doi.org/10.1145/882262.882285

[63] Julie Dorsey, Songhua Xu, Gabe Smedresman, Holly Rushmeier, and Leonard McMillan. 2007. The Mental Canvas: A Tool for Conceptual Architectural Design and Analysis. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications (PG '07)*. IEEE Computer Society, Washington, DC, USA, 201–210. https://doi.org/10.1109/PG.2007.62

[64] David Eberly. 2000. Least squares fitting of data. (2000). http://ncorr.com/download/publications/eberlyleastsquares.pdf

[65] Koos Eissen and Roselien Steur. 2010. *Sketching: drawing techniques for product designers* (8th pr ed.). BIS Publ, Amsterdam. OCLC: 845762757.

[66] Koos Eissen and Roselien Steur. 2011. *Sketching: the basics ; the prequel to Sketching: drawing techniques for product designers*. BIS, Amsterdam. OCLC: 756275344.

[67] Philip Ekströmer, Renee Wever, Patrik Andersson, and Johan Jönsson. 2019. Shedding light on game engines and virtual reality for design ideation. In *Proceedings of the Design Society: International Conference on Engineering Design*, Vol. 1. Cambridge University Press, 2003–2010. https://doi.org/10.1017/dsi.2019.206

[68] Hesham Elsayed, Mayra Donaji Barrera Machuca, Christian Schaarschmidt, Karola Marky, Florian Müller, Jan Riemann, Andrii Matviienko, Martin Schmitz, Martin Weigel, and Max Mühlhäuser. 2020. VRSketchPen: Unconstrained Haptic Assistance for Sketching in Virtual 3D Environments. In *26th ACM Symposium on Virtual Reality Software and Technology*. 1–11. https://doi.org/10.1145/3385956.3418953

[69] Epic Games. 2021. Unreal Engine. https://www.unrealengine.com/.

[70] Facebook. 2016. Oculus Rift S. https://www.oculus.com/rift-s/.

[71] Facebook. 2021. Quill. https://quill.fb.com.

[72] Lubin Fan, Ruimin Wang, Linlin Xu, Jiansong Deng, and Ligang Liu. 2013. Modeling by Drawing with Shadow Guidance. *Computer Graphics Forum* 32, 7 (2013), 157–166. https://doi.org/10.1111/cgf.12223

[73] Zhe Fan, Ma Chi, Arie Kaufman, and Manuel M. Oliveira. 2004. A Sketch-Based Interface for Collaborative Design .. In *Sketch Based Interfaces and Modeling*. The Eurographics Association, Geneve, Switzerland, 143–150. https://doi.org/10.2312/SBM/SBM04/143-150

[74] Gerald Farin and Nickolas Sapidis. 1989. Curvature and the Fairness of Curves and Surfaces. *IEEE Comput. Graph. Appl.* 9, 2 (March 1989), 52–57. https://doi.org/10.1109/38.19051

[75] Michele Fiorentino, Raffaele de Amicis, Giuseppe Monno, and Andre Stork. 2002. Spacedesign: A mixed reality workspace for aesthetic industrial design. In *Proceedings. International Symposium on Mixed and Augmented Reality*. IEEE, New York, NY, USA, 86–318. https://doi.org/10.1109/ISMAR.2002.1115077

[76] Michele Fiorentino, Giuseppe Monno, Pietro Alexander Renzulli, and Antonio E. Uva. 2003. 3D Sketch Stroke Segmentation and Fitting in Virtual Reality. In *International Conference Graphicon*. https://www.graphicon.ru/html/2003/Proceedings/Technical/paper435.pdf

[77] Jakub Fišer, Paul Asente, and Daniel Sỳkora. 2015. ShipShape: a drawing beautification assistant. In *Proceedings of the workshop on Sketch-Based Interfaces and Modeling*. Eurographics Association, Geneve, Switzerland, 49–57. https://dl.acm.org/doi/abs/10.5555/2810210.2810215

[78] Matthew Fisher, Peter Schröder, Mathieu Desbrun, and Hugues Hoppe. 2007. Design of Tangent Vector Fields. *ACM Trans. Graph.* 26, 3 (July 2007), 56–es. https://doi.org/10.1145/1276377.1276447

[79] George W. Fitzmaurice, Ravin Balakrishnan, Gordon Kurtenbach, and Bill Buxton. 1999. An Exploration into Supporting Artwork Orientation in the User Interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 167–174. https://doi.org/10.1145/302979.303033

[80] Hongbo Fu, Yichen Wei, Chiew-Lan Tai, and Long Quan. 2007. Sketching Hairstyles. In *Proceedings of the 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling* (Riverside, California) *(SBIM '07)*. Association for Computing Machinery, New York, NY, USA, 31–36. https://doi.org/10.1145/1384429.1384439

[81] Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. 2009. iWIRES: An Analyze-and-Edit Approach to Shape Manipulation. *ACM Transactions on Graphics (Siggraph)* 28, 3 (2009), #33, 1–10. https://doi.org/10.1145/1531326.1531339

[82] Anthony G. Gallagher, E. Matt Ritter, Howard Champion, Gerald Higgins, Marvin P. Fried, Gerald Moses, C. Daniel Smith, and Richard M. Satava. 2005. Virtual Reality Simulation for the Operating Room: Proficiency-Based Training as a Paradigm Shift in Surgical Skills Training. *Annals of Surgery* 241, 2 (2005), 364–372. http://journals.lww.com/annalsofsurgery/Fulltext/2005/02000/Virtual_Reality_Simulation_for_the_Operating_Room_.24.aspx

[83] Anne Gehre, Michael Bronstein, Leif Kobbelt, and Justin Solomon. 2018. Interactive Curve Constrained Functional Maps. *Computer Graphics Forum* 37, 5 (2018), 1–12. https://doi.org/10.1111/cgf.13486

[84] Joseph Gilland. 2009. *Elemental Magic, Volume I: The Art of Special Effects Animation*. Routledge.

[85] Bruce Gooch and Amy Gooch. 2001. *Non-Photorealistic Rendering*. A. K. Peters, USA.

[86] Google. 2021. Blocks. https://arvr.google.com/blocks/.

[87] Google. 2021. Tilt Brush. https://www.tiltbrush.com/.

[88] Google. 2021. Virtual Art Sessions. https://virtualart.chromeexperiments.com/.

[89] Google Spotlight Stories. 2017. Sonaria. https://store.steampowered.com/app/713320/Google_Spotlight_Stories_Sonaria/.

[90] Giorgio Gori, Alla Sheffer, Nicholas Vining, Enrique Rosales, Nathan Carr, and Tao Ju. 2017. FlowRep: Descriptive curve networks for free-form design shapes. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–14. https://doi.org/10.1145/3072959.3073639

[91] Gravity Sketch. 2021. Gravity Sketch. https://www.gravitysketch.com/.

[92] Tovi Grossman, Ravin Balakrishnan, Gordon Kurtenbach, George Fitzmaurice, Azam Khan, and Bill Buxton. 2002. Creating Principal 3D Curves with Digital Tape Drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Minneapolis, Minnesota, USA) *(CHI '02)*. ACM, New York, NY, USA, 121–128. https://doi.org/10.1145/503376.503398

[93] Tovi Grossman, Ravin Balakrishnan, and Karan Singh. 2003. An Interface for Creating and Manipulating Curves Using a High Degree-of-freedom Curve Input Device. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Ft. Lauderdale, Florida, USA) *(CHI '03)*. ACM, New York, NY, USA, 185–192. https://doi.org/10.1145/642611.642645

[94] Andrew J Hanson and Hui Ma. 1995. Parallel transport approach to curve framing. *Indiana University, Techreports-TR425* 11 (1995), 3–7. ftp://html.soic.indiana.edu/pub/techreports/TR425.pdf

[95] Changyu He, Peter Kazanzides, Hasan Tutkun Sen, Sungmin Kim, and Yue Liu. 2015. An inertial and optical sensor fusion approach for six degree-of-freedom pose estimation. *Sensors* 15, 7 (2015), 16448–16465. https://doi.org/10.3390/s150716448

[96] Frank Heckel, Jan H. Moltz, Christian Tietjen, and Horst K. Hahn. 2013. Sketch-Based Editing Tools for Tumour Segmentation in 3D Medical Images. *Computer Graphics Forum* 32, 8 (2013), 144–157. https://doi.org/10.1111/cgf.12193

[97] Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala. 2012. 3D Puppetry: A Kinect-based Interface for 3D Animation. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. ACM, New York, NY, USA. https://doi.org/10.1145/2380116.2380170

[98] Laura M. Herman and Stefanie Hutka. 2019. Virtual Artistry: Virtual Reality Translations of Two-Dimensional Creativity. In *Proceedings of the 2019 on Creativity and Cognition* (San Diego, CA, USA) *(C&C '19)*. Association for Computing Machinery, New York, NY, USA, 612–618. https://doi.org/10.1145/3325480.3326579

[99] Aaron Hertzmann and Denis Zorin. 2000. Illustrating Smooth Surfaces. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., USA, 517–526. https://doi.org/10.1145/344779.345074

[100] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. 1994. A Survey of Design Issues in Spatial Input. In *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology* (Marina del Rey, California, USA) *(UIST '94)*. ACM, New York, NY, USA, 213–222. https://doi.org/10.1145/192426.192501

[101] Christian Holz and Andrew Wilson. 2011. Data miming: inferring spatial object descriptions from human gesture. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 811–820. https://doi.org/10.1145/1978942.1979060

[102] Ian P. Howard and Brian J. Rogers. 2008. Depth from monocular cues and vergence. In *Seeing in Depth*. Oxford University Press, 355–410. https://doi.org/10.1093/acprof:oso/9780195367607.001.0001

[103] HTC. 2016. Vive. https://www.vive.com.

[104] Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 60 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201353

[105] Emmanuel Iarussi, David Bommes, and Adrien Bousseau. 2015. BendFields: Regularized Curvature Fields from Rough Concept Sketches. *ACM Trans. Graph.* 34, 3, Article 24 (May 2015), 16 pages. https://doi.org/10.1145/2710026

[106] Takeo Igarashi, Satoshi Matsuoka, Sachiko Kawachiya, and Hidehiko Tanaka. 2007. Interactive beautification: a technique for rapid geometric design. In *ACM SIGGRAPH 2007 courses*. ACM, New York, NY, USA, 18–es. https://doi.org/10.1145/1281500.1281529

[107] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. 1999. Teddy: A Sketching Interface for 3D Freeform Design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 409–416. https://doi.org/10.1145/311535.311602

[108] Takeo Igarashi, Tomer Moscovich, and John F Hughes. 2005. As-rigid-as-possible shape manipulation. *ACM transactions on Graphics (TOG)* 24, 3 (2005). https://doi.org/10.1145/1186822.1073323

[109] Poika Isokoski. 2001. Model for Unistroke Writing Time. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '01)*. ACM, New York, NY, USA, 357–364. https://doi.org/10.1145/365024.365299

[110] Johann Habakuk Israel, Eva Wiese, Magdalena Mateescu, Christian Zöllner, and Rainer Stark. 2009. Investigating three-dimensional sketching for early conceptual design—Results from expert discussions and user studies. *Computers & Graphics* 33, 4 (Aug. 2009), 462–473. https://doi.org/10.1016/j.cag.2009.05.005

[111] Bret Jackson and Daniel F. Keefe. 2011. Sketching Over Props: Understanding and Interpreting 3D Sketch Input Relative to Rapid Prototype Props. In *IUI 2011 Sketch Recognition Workshop*. http://ivlab.cs.umn.edu/papers/Jackson-2011-SketchRecognition.pdf

[112] Brett Jackson and Daniel F. Keefe. 2016. Lift-Off: Using Reference Imagery and Freehand Sketching to Create 3D Models in VR. *IEEE Transactions on Visualization and Computer Graphics* 22, 4 (April 2016), 1442–1451. https://doi.org/10.1109/TVCG.2016.2518099

[113] Alec Jacobson, Daniele Panozzo, et al. 2021. libigl: A simple C++ geometry processing library. https://libigl.github.io/.

[114] Michal Jelínek. 2020. Design ideation in virtual reality. *ALFA* (2020). https://alfa.stuba.sk/design-ideation-in-virtual-reality/

[115] Søren Qvist Jensen, Andreas Fender, and Jörg Müller. 2018. Inpher: Inferring Physical Properties of Virtual Objects from Mid-Air Interaction. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA. https://doi.org/10.1145/3173574.3174104

[116] Thomas Jung, Mark D. Gross, and Ellen Yi-Luen Do. 2002. Annotating and Sketching on 3D Web Models. In *Proceedings of the 7th International Conference on Intelligent User Interfaces* (San Francisco, California, USA) *(IUI '02)*. Association for Computing Machinery, New York, NY, USA, 95–102. https://doi.org/10.1145/502716.502733

[117] Kiia Kallio. 2005. 3D6B Editor: Projective 3D Sketching with Line-Based Rendering. The Eurographics Association. https://doi.org/10.2312/SBM/SBM05/073-079

[118] Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, and Adam Finkelstein. 2002. WYSIWYG NPR: Drawing Strokes Directly on 3D Models. *ACM Trans. Graph.* 21, 3 (July 2002), 755—762. https://doi.org/10.1145/566654.566648

[119] Sho Kamuro, Kouta Minamizawa, and Susumu Tachi. 2011. 3D Haptic Modeling System using Ungrounded Pen-shaped Kinesthetic Display. In *2011 IEEE Virtual Reality Conference*. IEEE, New York, NY, USA, 217–218.

[120] Levent Burak Kara and Kenji Shimada. 2007. Sketch-Based 3D-Shape Creation for Industrial Styling Design. *IEEE Comput. Graph. Appl.* 27, 1 (Jan. 2007), 60–71. https://doi.org/10.1109/MCG.2007.18

[121] Maria Karam and m.c. schraefel. 2005. *A taxonomy of gestures in human computer interactions.* Technical Report ECSTR-IAM05-009. https://eprints.soton.ac.uk/261149/

[122] Peter Karasev, Ivan Kolesov, Karl Fritscher, Patricio Vela, Phillip Mitchell, and Allen Tannenbaum. 2013. Interactive medical image segmentation using PDE control of active contours. *IEEE transactions on medical imaging* 32, 11 (2013), 2127–2139. https://doi.org/10.1109/TMI.2013.2274734

[123] Raghavendra S. Kattinakere, Tovi Grossman, and Sriram Subramanian. 2007. Modeling steering within above-the-surface interaction layers. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 317–326. https://doi.org/10.1145/1240624.1240678

[124] Michael Kazhdan, Jake Solomon, and Mirela Ben-Chen. 2012. Can mean-curvature flow be modified to be non-singular? *Computer Graphics Forum* 31, 5 (2012), 1745–1754. https://doi.org/10.1111/j.1467-8659.2012.03179.x

[125] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, and George Fitzmaurice. 2014. Kitty: Sketching Dynamic and Interactive Illustrations. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. ACM, New York, NY, USA. https://doi.org/10.1145/2642918.2647375

[126] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, Shengdong Zhao, and George Fitzmaurice. 2014. Draco: Bringing Life to Illustrations with Kinetic Textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. ACM, New York, NY, USA, 351–360. https://doi.org/10.1145/2556288.2556987

[127] Rubaiat Habib Kazi, Kien Chuan Chua, Shengdong Zhao, Richard Davis, and Kok-Lim Low. 2011. SandCanvas: a multi-touch art medium inspired by sand animation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. https://doi.org/10.1145/1978942.1979133

[128] Rubaiat Habib Kazi, Tovi Grossman, Hyunmin Cheong, Ali Hashemi, and George Fitzmaurice. 2017. DreamSketch: Early Stage 3D Design Explorations with Sketching and Generative Design. ACM, Quebec City, QC, Canada. https://doi.org/10.1145/3126594.3126662

[129] Ismail Khalid Kazmi, Lihua You, and Jian Jun Zhang. 2014. A Survey of Sketch Based Modeling Systems. In *2014 11th International Conference on Computer Graphics, Imaging and Visualization*. 27–36. https://doi.org/10.1109/CGiV.2014.27

[130] Daniel Keefe, Robert Zeleznik, and David Laidlaw. 2007. Drawing on air: Input techniques for controlled 3D line illustration. *IEEE Transactions on Visualization and Computer Graphics* 13, 5 (2007), 1067–1081. https://doi.org/10.1109/TVCG.2007.1060

[131] Daniel F. Keefe, Daniel Acevedo Feliz, Tomer Moscovich, David H. Laidlaw, and Joseph J. LaViola. 2001. CavePainting: A Fully Immersive 3D Artistic Medium and Interactive Experience. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics (I3D '01)*. Association for Computing Machinery, New York, NY, USA, 85–93. https://doi.org/10.1145/364338.364370

[132] Daniel F. Keefe and David H. Laidlaw. 2007. Analysis of Performance in Precise 3D Curve Input Tasks in Virtual Reality. *IEEE Visualization (Best Posters Session)* (2007). http://ivlab.cs.umn.edu/papers/Keefe-2007-CurveInputPoster.pdf

[133] Micky Kelager. 2006. *Lagrangian fluid dynamics using smoothed particle hydrodynamics*. Master's thesis. University of Copenhagen: Dept. of Computer Science. http://image.diku.dk/projects/media/kelager.06.pdf

[134] Todd R Kelley. 2017. Design sketching: A lost skill: It is not enough to have students know how to create design sketches but to also know the purpose of sketching in design. *Technology and engineering teacher* 76, 8 (2017), 8. https://www.iteea.org/Publications/Journals/TET/TETMayJune2017.aspx

[135] Hyunyoung Kim, Celine Coutrix, and Anne Roudaut. 2018. Morphees+: Studying Everyday Reconfigurable Objects for the Design and Taxonomy of Reconfigurable UIs. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 619, 14 pages. https://doi.org/10.1145/3173574.3174193

[136] Yongkwan Kim, Sang-Gyun An, Joon Hyub Lee, and Seok-Hyung Bae. 2018. Agile 3D sketching with air scaffolding. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/3170427.3186522

[137] Yongkwan Kim and Seok-Hyung Bae. 2016. SketchingWithHands: 3D Sketching Handheld Products with First-Person Hand Posture. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 797–808. https://doi.org/10.1145/2984511.2984567

[138] Venkat Krishnamurthy and Marc Levoy. 1996. Fitting Smooth Surfaces to Dense Polygon Meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. Association for Computing Machinery, New York, NY, USA, 313–324. https://doi.org/10.1145/237170.237270

[139] Vojtěch Krs, Ersin Yumer, Nathan Carr, Bedrich Benes, and Radomír Měch. 2017. Skippy: Single View 3D Curve Interactive Modeling. *ACM Trans. Graph.* 36, 4, Article 128 (July 2017), 12 pages. https://doi.org/10.1145/3072959.3073603

[140] Tom Kühnert, Stephan Rusdorf, and Guido Brunnett. 2011. Virtual Prototyping of Shoes. *IEEE Computer Graphics and Applications* 31, 5 (Sept. 2011), 30–42. https://doi.org/10.1109/MCG.2010.81

[141] Kin Chung Kwan and Hongbo Fu. 2019. Mobi3DSketch: 3D Sketching in Mobile AR. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland, UK) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3290605.3300406

[142] Jung-hoon Kwon, Han-wool Choi, Jeong-in Lee, and Young-Ho Chai. 2005. Free-Hand Stroke Based NURBS Surface for Sketching and Deforming 3D Contents. In *Advances in Multimedia Information Processing - PCM 2005*, Yo-Sung Ho and Hyoung Joong Kim (Eds.). Number 3767 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 315–326. https://doi.org/10.1007/11581772_28

[143] Paul Lapides, Ehud Sharlin, Mario Costa Sousa, and Lisa Streit. 2006. The 3D Tractus: A Three-Dimensional Drawing Board. IEEE, 169–176. https://doi.org/10.1109/TABLETOP.2006.33

[144] Sylvain Lefebvre and Hugues Hoppe. 2006. Appearance-Space Texture Synthesis. *ACM Trans. Graph.* 25, 3 (July 2006), 541–548. https://doi.org/10.1145/1141911.1141921

[145] Hoo Yong Leng, Noris Mohd Norowi, and Azrul Hazri Jantan. 2017. A User-Defined Gesture Set for Music Interaction in Immersive Virtual Environment. In *Proceedings of the 3rd International Conference on Human-Computer Interaction and User Experience in Indonesia* (Jakarta, Indonesia) *(CHIuXiD '17)*. ACM, New York, NY, USA. https://doi.org/10.1145/3077343.3077348

[146] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least Squares Conformal Maps for Automatic Texture Atlas Generation. *ACM Trans. Graph.* 21, 3 (July 2002), 362–371. https://doi.org/10.1145/566654.566590

[147] Yuwei Li, Xi Luo, Youyi Zheng, Pengfei Xu, and Hangbo Fu. 2017. SweepCanvas: Sketch-based 3D Prototyping on an RGB-D Image. ACM, Quebec City, QC, Canada.

[148] Chenxi Liu, Enrique Rosales, and Alla Sheffer. 2018. Strokeaggregator: Consolidating raw sketches into artist-intended curve drawings. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–15. https://doi.org/10.1145/3197517.3201314

[149] Noah Lockwood and Karan Singh. 2012. Finger Walking: Motion Editing with Contact-based Hand Performance. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Lausanne, Switzerland) *(SCA '12)*. Eurographics Association, Germany. http://dl.acm.org/citation.cfm?id=2422364

[150] Logitech. 2021. VR Ink Pilot Edition. https://www.logitech.com/en-roeu/promo/vr-ink.html.

[151] Mayra D. Barrera Machuca, Paul Asente, Wolfgang Stuerzlinger, Jingwan Lu, and Byungmoon Kim. 2018. Multiplanes: Assisted Freehand VR Sketching. In *Proceedings of the Symposium on Spatial User Interaction* (Berlin, Germany) *(SUI '18)*. Association for Computing Machinery, New York, NY, USA, 36–47. https://doi.org/10.1145/3267782.3267786

[152] Mayra Donaji Barrera Machuca, Wolfgang Stuerzlinger, and Paul Asente. 2019. The Effect of Spatial Ability on Immersive 3D Drawing. In *Proceedings of the ACM Conference on Creativity & Cognition (C&C'19)*. ACM, New York, NY, USA, 173–186. https://doi.org/10.1145/3325480.3325489

[153] Joel R. Martin, Vladimir M. Zatsiorsky, and Mark L. Latash. 2011. Multi-finger interaction during involuntary and voluntary single finger force changes. *Experimental brain research* 208, 3 (2011), 423–435. https://doi.org/10.1007/s00221-010-2492-z

[154] Masterpiece. 2021. Masterpiece Studio. https://masterpiecestudio.com.

[155] James McCrae and Karan Singh. 2009. Sketching piecewise clothoid curves. *Computers & Graphics* 33, 4 (2009), 452–461. https://doi.org/10.1016/j.cag.2009.05.006

[156] James McCrae, Karan Singh, and Niloy J. Mitra. 2011. Slices: A Shape-Proxy Based on Planar Sections. *ACM Trans. Graph.* 30, 6 (Dec. 2011), 1–12. https://doi.org/10.1145/2070781.2024202

[157] James McCrae, Nobuyuki Umetani, and Karan Singh. 2014. FlatFitFab: Interactive Modeling with Planar Sections. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 13–22. https://doi.org/10.1145/2642918.2647388

[158] David McNeill. 1992. *Hand and mind: What gestures reveal about thought*. University of Chicago press.

[159] Min Meng, Lubin Fan, and Ligang Liu. 2011. iCutter: A Direct Cut-out Tool for 3D Shapes. *Computer Animation and Virtual Worlds* 22, 4 (2011), 335–342. https://doi.org/10.1002/cav.422

[160] Microsoft. 2018. HoloLens. https://www.microsoft.com/en-us/hololens.

[161] Microsoft. 2018. Surface Book. https://www.microsoft.com/en-us/surface-book.

[162] Microsoft. 2021. MixedRealityToolkit-Unity. https://github.com/Microsoft/MixedRealityToolkit-Unity. original-date: 2016-01-28T18:54:58Z.

[163] Robert D Miller. 1995. Quick and simple bezier curve drawing. In *Graphics Gems V*. Elsevier, 206–209. https://doi.org/10.1016/B978-0-12-543457-7.50036-X

[164] Ali Momeni and Zachary Rispoli. 2016. Dranimate: Paper Becomes Tablet, Drawing Becomes Animation. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI EA '16)*. ACM, New York, NY, USA, 3735–3737. https://doi.org/10.1145/2851581.2890267

[165] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. 2007. FiberMesh: Designing Freeform Surfaces with 3D Curves. *ACM Trans. Graph.* 26, 3 (July 2007), 41–es. https://doi.org/10.1145/1276377.1276429

[166] Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization*. Springer Science & Business Media, Berlin-Heidelberg, Germany.

[167] Nvrmind. 2021. AnimVR. https://nvrmind.io.

[168] Ollie. 2021. Ollie VR. https://ollievr.com.

[169] Luke Olsen, Faramarz F. Samavati, Mario Costa Sousa, and Joaquim A. Jorge. 2009. Sketch-based Modeling: A survey. *Computers and Graphics* 33, 1 (2009), 85–103. https://doi.org/10.1016/j.cag.2008.09.013

[170] OptiTrack. [n.d.]. Motive – Optical Motion Capture Software. https://optitrack.com/software/motive/.

[171] Günay Orbay and Levent Burak Kara. 2012. Sketch-based surface design using malleable curve networks. *Computers & Graphics* 36, 8 (2012), 916–929. https://doi.org/10.1016/j.cag.2012.08.008

[172] Michaël Ortega and Thomas Vincent. 2014. Direct Drawing on 3D Shapes with Automated Camera Control. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 2047–2050. https://doi.org/10.1145/2556288.2557242

[173] Patrick Paczkowski, Julie Dorsey, Holly Rushmeier, and Min H Kim. 2018. PaperCraft3D: Paper-based 3D Modeling and Scene Fabrication. *IEEE Transactions on Visualization and Computer Graphics* 25, 4 (2018), 1717–1731.

[174] Patrick Paczkowski, Min H. Kim, Yann Morvan, Julie Dorsey, Holly Rushmeier, and Carol O'Sullivan. 2011. Insitu: Sketching Architectural Designs in Context. *ACM Trans. Graph.* 30, 6 (Dec. 2011), 1–10. https://doi.org/10.1145/2070781.2024216

[175] PaintLab. 2021. PaintLab VR - Paint and Sculpt in Virtual Reality. http://paintlabvr.com.

[176] Hao Pan, Yang Liu, Alla Sheffer, Nicholas Vining, Chang-Jian Li, and Wenping Wang. 2015. Flow aligned surfacing of curve networks. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10. https://doi.org/10.1145/2766990

[177] Zherong Pan, Jin Huang, Yiying Tong, Changxi Zheng, and Hujun Bao. 2013. Interactive Localized Liquid Motion Editing. *ACM Trans. Graph.* 32, 6 (Nov. 2013). https://doi.org/10.1145/2508363.2508429

[178] Julius Panero and Martin Zelnik. 1979. *Human Dimension and Interior Space: A Source Book of Design Reference Standards* (1st ed.). Watson-Guptill, New York.

[179] Daniele Panozzo, Ilya Baran, Olga Diamanti, and Olga Sorkine-Hornung. 2013. Weighted Averages on Surfaces. *ACM Trans. Graph.* 32, 4, Article 60 (July 2013), 12 pages. https://doi.org/10.1145/2461912.2461935

[180] Rick Parent. 2012. *Computer Animation: Algorithms and Techniques*. Elsevier Science. https://books.google.ca/books?id=DudZtbOD2gMC

[181] Theo Pavlidis and Christopher J Van Wyk. 1985. An automatic beautifier for drawings and illustrations. *ACM SIGGRAPH Computer Graphics* 19, 3 (1985), 225–234. https://doi.org/10.1145/325334.325240

[182] Ken Perlin. 2002. Improving Noise. *ACM Trans. Graph.* 21, 3 (July 2002), 681–682. https://doi.org/10.1145/566654.566636

[183] Frederic Pighin and J.P. Lewis. 2006. Performance-Driven Facial Animation. In *SIGGRAPH Courses*. ACM. https://doi.org/10.1145/1185657.1185856

[184] Alan Pipes and Richard Mason. 2007. *Drawing for Designers*. Laurence King Publishing, London, UK. https://books.google.co.nz/books?id=phgfAQAAIAAJ

[185] Thammathip Piumsomboon, Adrian Clark, Mark Billinghurst, and Andy Cockburn. 2013. User-Defined Gestures for Augmented Reality. In *Human-Computer Interaction – INTERACT 2013*, Paula Kotzé, Gary Marsden, Gitte Lindgaard, Janet Wesson, and Marco Winckler (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 282–299. https://doi.org/10.1007/978-3-642-40480-1_18

[186] Thammathip Piumsomboon, Gun Lee, Robert W Lindeman, and Mark Billinghurst. 2017. Exploring natural eye-gaze-based interaction for immersive virtual reality. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 36–39. https://doi.org/10.1109/3DUI.2017.7893315

[187] Pixologic. 2021. ZBrush - The all-in-one-digital sculpting solution. http://pixologic.com.

[188] Konrad Polthier and Markus Schmies. 2006. Straightest Geodesics on Polyhedral Surfaces. In *ACM SIGGRAPH 2006 Courses* (Boston, Massachusetts) *(SIGGRAPH '06)*. Association for Computing Machinery, New York, NY, USA, 30–38. https://doi.org/10.1145/1185657.1185664

[189] Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 2, Article 16 (April 2017), 16 pages. https://doi.org/10.1145/2983621

[190] Jun Rekimoto. 2002. SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Minneapolis, Minnesota, USA) *(CHI '02)*. ACM, New York, NY, USA, 113–120. https://doi.org/10.1145/503376.503397

[191] Mitchel Resnick, Brad Myers, Kumiyo Nakakoji, Ben Shneiderman, Randy Pausch, Ted Selker, and Mike Eisenberg. 2005. Design Principles for Tools to Support Creative Thinking. (2005).

[192] Robert McNeel & Associates. 2018. Rhinoceros 3D. https://www.rhino3d.com/.

[193] Judy Robertson and Maurits Kaptein. 2016. *Modern statistical methods for HCI*. Springer.

[194] Scott Robertson and Thomas Bertling. 2013. *How to draw: drawing and sketching objects and environments from your imagination*. Designstudio Press, Culver City, Calif. OCLC: 897060023.

[195] Isabel Benavente Rodriguez and Nicolai Marquardt. 2017. Gesture Elicitation Study on How to Opt-in & Opt-out from Interactions with Public Displays. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces* (Brighton, United Kingdom) *(ISS '17)*. ACM, New York, NY, USA, 32–41. https://doi.org/10.1145/3132272.3134118

[196] Hugo Romat, Andreas Fender, Manuel Meier, and Christian Holz. 2021. Flashpen: A High-Fidelity and High-Precision Multi-Surface Pen for Virtual Reality. In *Proceedings IEEE Virtual Reality 2021*. https://siplab.org/papers/vr2021-flashpen.pdf

[197] Enrique Rosales, Jafet Rodriguez, and Alla Sheffer. 2019. SurfaceBrush: From Virtual Reality Drawings to Manifold Surfaces. *ACM Transaction on Graphics* 38, 4, Article 96 (2019), 15 pages. https://doi.org/10.1145/3306346.3322970

[198] Christoph Rüegg et al. 2009. MathNET Numerics. https://numerics.mathdotnet.com/.

[199] Jaime Ruiz, Yang Li, and Edward Lank. 2011. User-defined Motion Gestures for Mobile Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. ACM, New York, NY, USA, 197–206. https://doi.org/10.1145/1978942.1978971

[200] E. Sachs, A. Roberts, and D. Stoops. 1991. 3-Draw: a tool for designing 3D shapes. *IEEE Computer Graphics and Applications* 11, 6 (Nov. 1991), 18–26. https://doi.org/10.1109/38.103389

[201] Bardia Sadri and Karan Singh. 2014. Flow-complex-based Shape Reconstruction from 3D Curves. *ACM Trans. Graph.* 33, 2, Article 20 (April 2014), 15 pages. https://doi.org/10.1145/2560328

[202] Paola Salomoni, Catia Prandi, Marco Roccetti, Lorenzo Casanova, Luca Marchetti, and Gustavo Marfia. 2017. Diegetic user interfaces for virtual environments with HMDs: a user experience study with oculus rift. *Journal on Multimodal User Interfaces* 11, 2 (2017), 173–184. https://doi.org/10.1007/s12193-016-0236-5

[203] Rohan Sawhney and Keenan Crane. 2017. Boundary First Flattening. *ACM Trans. Graph.* 37, 1, Article 5 (Dec. 2017), 14 pages. https://doi.org/10.1145/3132705

[204] Steven Schkolne, Michael Pruett, and Peter Schröder. 2001. Surface Drawing: Creating Organic 3D Shapes with the Hand and Tangible Tools. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Seattle, Washington, USA) *(CHI '01)*. ACM, New York, NY, USA, 261–268. https://doi.org/10.1145/365024.365114

[205] Johannes Schmid, Martin Sebastian Senn, Markus Gross, and Robert W. Sumner. 2011. OverCoat: An Implicit Canvas for 3D Painting. *ACM Trans. Graph.* 30, 4, Article 28 (July 2011), 10 pages. https://doi.org/10.1145/2010324.1964923

[206] Ryan Schmidt. 2021. geometry3sharp: Open-Source (Boost-license) C# Library for Geometric Computing. https://github.com/gradientspace/geometry3Sharp.

[207] Ryan Schmidt, Cindy Grimm, and Brian Wyvill. 2006. Interactive Decal Compositing with Discrete Exponential Maps. *ACM Trans. Graph.* 25, 3 (July 2006), 605–613. https://doi.org/10.1145/1141911.1141930

[208] Ryan Schmidt, Azam Khan, Gord Kurtenbach, and Karan Singh. 2009. On Expert Performance in 3D Curve-drawing Tasks. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling (SBIM '09)*. ACM, New York, NY, USA, 133–140. https://doi.org/10.1145/1572741.1572765

[209] Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. 2009. Analytic drawing of 3D scaffolds. *ACM transactions on graphics (TOG)* 28, 5 (2009), 1–10. https://doi.org/10.1145/1661412.1618495

[210] Ryan Schmidt and Karan Singh. 2008. Sketch-Based Procedural Surface Modeling and Compositing Using Surface Trees. *Computer Graphics Forum* 27, 2 (April 2008), 321–330. https://doi.org/10.1111/j.1467-8659.2008.01129.x

[211] Ryan Schmidt and Karan Singh. 2010. Meshmixer: An Interface for Rapid Mesh Composition. In *ACM SIGGRAPH 2010 Talks* (Los Angeles, California) *(SIGGRAPH '10)*. ACM, New York, NY, USA, Article 6, 1 pages. https://doi.org/10.1145/1837026.1837034

[212] Philip J. Schneider. 1990. Graphics Gems. Academic Press Professional, Inc., San Diego, CA, USA, Chapter An Algorithm for Automatically Fitting Digitized Curves, 612–626. http://dl.acm.org/citation.cfm?id=90767.90941

[213] Valentin Schwind, Pascal Knierim, Cagri Tasci, Patrick Franczak, Nico Haas, and Niels Henze. 2017. "These Are Not My Hands!": Effect of Gender on the Perception of Avatar Hands in Virtual Reality. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1577–1582. https://doi.org/10.1145/3025453.3025602

[214] Cloud Shao, Adrien Bousseau, Alla Sheffer, and Karan Singh. 2012. CrossShade: Shading Concept Sketches Using Cross-section Curves. *ACM Trans. Graph.* 31, 4, Article 45 (July 2012), 11 pages. https://doi.org/10.1145/2185520.2185541

[215] Nicholas Sharp et al. 2019. Polyscope. www.polyscope.run.

[216] Jia Sheng, Ravin Balakrishnan, and Karan Singh. 2006. An Interface for Virtual 3D Sculpting via Physical Proxy. In *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia* (Kuala Lumpur, Malaysia) *(GRAPHITE '06)*. ACM, New York, NY, USA, 213–220. https://doi.org/10.1145/1174429.1174467

[217] Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2, Article 11 (Feb. 2015), 36 pages. https://doi.org/10.1145/2629697

[218] SideFX. 2021. Houdini. https://sidefx.com/products/houdini.

[219] Karan Singh. 2006. Industrial motivation for interactive shape modeling: a case study in conceptual automotive design. ACM Press, 3. https://doi.org/10.1145/1185657.1185671

[220] Karan Singh and Eugene Fiume. 1998. Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, 405–414. https://doi.org/10.1145/280814.280946

[221] Hyunyoung Song, Hrvoje Benko, Francois Guimbretiere, Shahram Izadi, Xiang Cao, and Ken Hinckley. 2011. Grips and gestures on a multi-touch pen. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1323–1332. https://doi.org/10.1145/1978942.1979138

[222] Peng Song, Wooi Boon Goh, William Hutama, Chi-Wing Fu, and Xiaopei Liu. 2012. A Handle Bar Metaphor for Virtual Object Manipulation with Mid-air Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. ACM, New York, NY, USA, 1297–1306. https://doi.org/10.1145/2207676.2208585

[223] Olga Sorkine and Daniel Cohen-Or. 2004. Least-squares Meshes. In *Proceedings of Shape Modeling International* (Genova, Italy). IEEE Computer Society Press, Piscataway, NJ, USA, 191–199. https://doi.org/10.1109/SMI.2004.1314506

[224] Jos Stam. 2003. Flows on Surfaces of Arbitrary Topology. In *ACM SIGGRAPH 2003 Papers* (San Diego, California) *(SIGGRAPH '03)*. Association for Computing Machinery, New York, NY, USA, 724–731. https://doi.org/10.1145/1201775.882338

[225] Lucian Stanculescu, Raphaëlle Chaine, Marie-Paule Cani, and Karan Singh. 2013. Sculpting Multi-dimensional Nested Structures. *Comput. Graph.-UK* 37, 6 (Oct. 2013), 753–763. https://doi.org/10.1016/j.cag.2013.05.010 Special issue: Shape Modeling International (SMI) Conference 2013.

[226] Tibor Stanko, Stefanie Hahmann, Georges-Pierre Bonneau, and Nathalie Saguin-Sprynski. 2016. Smooth interpolation of curve networks with surface normals. In *EG 2016 - Short Papers*. The Eurographics Association, Geneve, Switzerland, 21–24. https://doi.org/10.2312/egsh.20161005

[227] Alex W Stedmon, Harshada Patel, Sarah C Sharples, and John R Wilson. 2011. Developing speech input for virtual reality applications: A reality based interaction approach. *International journal of human-computer studies* 69, 1-2 (2011), 3–8. https://doi.org/10.1016/j.ijhcs.2010.09.002

[228] Christian Steins, Sean Gustafson, Christian Holz, and Patrick Baudisch. 2013. Imaginary devices: gesture-based interaction mimicking traditional input devices. In *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services*. ACM, 123–126. https://doi.org/10.1145/2493190.2493208

[229] Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J. Gortler, and Hugues Hoppe. 2005. Fast Exact and Approximate Geodesics on Meshes. *ACM Trans. Graph.* 24, 3 (July 2005), 553–560. https://doi.org/10.1145/1073204.1073228

[230] Ivan E. Sutherland. 1963. Sketchpad: A Man-machine Graphical Communication System. In *Proceedings of the May 21-23, 1963, Spring Joint Computer Conference (AFIPS '63 (Spring))*. ACM, New York, NY, USA, 329–346. https://doi.org/10.1145/1461551.1461591

[231] Kenshi Takayama, Daniele Panozzo, Alexander Sorkine-Hornung, and Olga Sorkine-Hornung. 2013. Sketch-based Generation and Editing of Quad Meshes. *ACM Trans. Graph.* 32, 4, Article 97 (July 2013), 8 pages. https://doi.org/10.1145/2461912.2461955

[232] Shun' ichi Tano, Shinya Yamamoto, Junko Ichino, Tomonori Hashiyama, and Mitsuru Iwata. 2013. Truly Useful 3D Drawing System for Professional Designer by "Life-Sized and Operable" Feature and New Interaction. In *Human-Computer Interaction–INTERACT 2013*. Number 8117 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 37–55. https://doi.org/10.1007/978-3-642-40483-2_3

[233] Brandon T. Taylor and V. Michael Bove Jr. 2009. Graspables: grasp-recognition as a user interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 917–926. https://doi.org/10.1145/1518701.1518842

[234] Daniel Thalmann. 1993. Using virtual reality techniques in the animation process. In *Virtual Reality Systems*. Elsevier, 143–159. https://doi.org/10.1016/B978-0-12-227748-1.50019-6

[235] The Blender Foundation. 2021. blender. https://blender.org/features/simulation.

[236] The CGAL Project. 2020. *CGAL User and Reference Manual* (5.0.2 ed.). CGAL Editorial Board. https://doc.cgal.org/5.0.2/Manual/packages.html

[237] S Thieffry. 1981. Hand gestures. *The Hand (R. Tubiana, ed.)* 488 (1981).

[238] Yannick Thiel, Karan Singh, and Ravin Balakrishnan. 2011. Elasticurves: Exploiting Stroke Dynamics and Inertia for the Real-Time Neatening of Sketched 2D Curves. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 383–392. https://doi.org/10.1145/2047196.2047246

[239] Thinksquirrel. 2021. Fluvio. https://getfluv.io.

[240] Jean-Philippe Thiran, Ferran Marques, and Hervé Bourlard. 2009. *Multimodal Signal Processing: Theory and applications for human-computer interaction*. Academic Press.

[241] Matthew Thorne, David Burke, and Michiel van de Panne. 2004. Motion doodles: an interface for sketching character motion. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 424–431. https://doi.org/10.1145/1186562.1015740

[242] Yiying Tong, Pierre Alliez, David Cohen-Steiner, and Mathieu Desbrun. 2006. Designing Quadrangulations with Discrete Harmonic Forms. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Cagliari, Sardinia, Italy) *(SGP '06)*. Eurographics Association, Goslar, DEU, 201–210. https://dl.acm.org/doi/abs/10.5555/1281957.1281983

[243] Giovanni Maria Troiano, Esben Warming Pedersen, and Kasper Hornbæk. 2014. User-defined Gestures for Elastic, Deformable Displays. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces* (Como, Italy) *(AVI '14)*. ACM, New York, NY, USA, 1–8. https://doi.org/10.1145/2598153.2598184

[244] Greg Turk. 2001. Texture Synthesis on Surfaces. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 347–354. https://doi.org/10.1145/383259.383297

[245] Emmanuel Turquin, Jamie Wither, Laurence Boissieux, Marie-Paule Cani, and John F. Hughes. 2007. A Sketch-Based Interface for Clothing Virtual Characters. *IEEE Comput. Graph. Appl.* 27, 1 (Jan. 2007), 72–81. https://doi.org/10.1109/MCG.2007.1

[246] Tvori. 2021. Tvori VR. http://tvori.co.

[247] Unity Technologies. 2021. Unity. https://unity.com.

[248] Valve. 2021. SteamVR Unity Plugin. https://github.com/ValveSoftware/steamvr_unity_plugin.

[249] Radu-Daniel Vatavu. 2012. User-defined Gestures for Free-hand TV Control. In *Proceedings of the 10th European Conference on Interactive TV and Video* (Berlin, Germany) *(EuroITV '12)*. ACM, New York, NY, USA, 45–48. https://doi.org/10.1145/2325616.2325626

[250] Floor Verhoeven and Olga Sorkine-Hornung. 2019. RodMesh: Two-handed 3D Surface Modeling in Virtual Reality. In *Proceedings of the Symposium on Vision, Modeling and Visualization (VMV)*. The Eurographics Association, Geneve, Switzerland, 10. https://doi.org/10.2312/vmv.20191312

[251] Ilse M Verstijnen, Cees van Leeuwen, G Goldschmidt, Ronald Hamel, and JM Hennessey. 1998. Sketching and creative discovery. *Design studies* 19, 4 (1998), 519–546. https://doi.org/10.1016/S0142-694X(98)00017-9

[252] Vicon. 2021. Tracker Motion Capture Software. https://www.vicon.com/software/tracker/.

[253] Paolo Viviani and Carlo Ternuolo. 1982. Trajectory Determines Movement Dynamics. *Neuroscience* 7, 2 (1982), 431–7. https://doi.org/10.1016/0306-4522(82)90277-9

[254] Stephen Voida, Mark Podlaseck, Rick Kjeldsen, and Claudio Pinhanez. 2005. A Study on the Manipulation of 2D Objects in a Projector/Camera-based Augmented Reality Environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Portland, Oregon, USA) *(CHI '05)*. ACM, New York, NY, USA, 611–620. https://doi.org/10.1145/1054972.1055056

[255] Gerold Wesche and Hans-Peter Seidel. 2001. FreeDrawer: a free-form sketching system on the responsive workbench. ACM Press, 167. https://doi.org/10.1145/505008.505041

[256] Brian Whited, Eric Daniels, Michael Kaschalk, Patrick Osborne, and Kyle Odermatt. 2012. Computer-assisted Animation of Line and Paint in Disney's Paperman. In *ACM SIGGRAPH 2012 Talks* (Los Angeles, California) *(SIGGRAPH '12)*. ACM, New York, NY, USA, Article 19, 1 pages. https://doi.org/10.1145/2343045.2343071

[257] Eva Wiese, Johann Habakuk Israel, Achim Meyer, and Sara Bongartz. 2010. Investigating the Learnability of Immersive Free-Hand Sketching. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium* (Annecy, France) *(SBIM '10)*. Eurographics Association, Goslar, DEU, 135–142. https://dl.acm.org/doi/abs/10.5555/1923363.1923387

[258] Daniel Wigdor, Hrvoje Benko, John Pella, Jarrod Lombardo, and Sarah Williams. 2011. Rock & Rails: Extending Multi-touch Interactions with Shape Gestures to Enable Precise Spatial Manipulations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. ACM, New York, NY, USA, 1581–1590. https://doi.org/10.1145/1978942.1979173

[259] Rand R Wilcox. 2011. *Introduction to robust estimation and hypothesis testing*. Academic press.

[260] Raphael Wimmer and Sebastian Boring. 2009. HandSense: discriminating different ways of grasping and holding a tangible user interface. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*. ACM, 359–362. https://doi.org/10.1145/1517664.1517736

[261] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined Gestures for Surface Computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. ACM, New York, NY, USA, 1083–1092. https://doi.org/10.1145/1518701.1518866

[262] Erik Wolf, Sara Klüber, Chris Zimmerer, Jean-Luc Lugrin, and Marc Erich Latoschik. 2019. "Paint that object yellow": Multimodal Interaction to Enhance Creativity During Design Tasks in VR. In *2019 International Conference on Multimodal Interaction*. 195–204. https://doi.org/10.1145/3340555.3353724

[263] Mike Wu and Ravin Balakrishnan. 2003. Multi-finger and Whole Hand Gestural Interaction Techniques for Multi-user Tabletop Displays. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology* (Vancouver, Canada) *(UIST '03)*. ACM, New York, NY, USA, 193–202. https://doi.org/10.1145/964696.964718

[264] Shihong Xia, Lin Gao, Yu-Kun Lai, Mingzhe Yuan, and Jinxiang Chai. 2017. A Survey on Human Performance Capture and Animation. *J. Comput. Sci. Technol.* 32, 3 (2017), 536–554. https://doi.org/10.1007/s11390-017-1742-y

[265] Jun Xing, Rubaiat Habib Kazi, Tovi Grossman, Li-Yi Wei, Jos Stam, and George Fitzmaurice. 2016. Energy-Brushes: Interactive Tools for Illustrating Stylized Elemental Dynamics. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. ACM, New York, NY, USA, 755–766. https://doi.org/10.1145/2984511.2984585

[266] Jun Xing, Koki Nagano, Weikai Chen, Haotian Xu, Li-Yi Wei, Yajie Zhao, Jingwan Lu, Byungmoon Kim, and Hao Li. 2019. HairBrush for Immersive Data-Driven Hair Modeling. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 263–279. https://doi.org/10.1145/3332165.3347876

[267] Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. 2014. True2Form: 3D Curve Networks from 2D Sketches via Selective Regularization. *ACM Trans. Graph.* 33, 4, Article 131 (July 2014), 13 pages. https://doi.org/10.1145/2601097.2601128

[268] Yukang Yan, Chun Yu, Xiaojuan Ma, Xin Yi, Ke Sun, and Yuanchun Shi. 2018. VirtualGrasp: Leveraging Experience of Interacting with Physical Objects to Facilitate Digital Object Retrieval. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 78. https://doi.org/10.1145/3173574.3173652

[269] Maria C Yang and Jorge G Cham. 2007. An analysis of sketching skill and its role in early stage engineering design. *Journal of Mechanical Design* 129, 5 (2007), 476–482. https://doi.org/10.1115/1.2712214

[270] Xiao Yi, Shengfeng Qin, and Jinsheng Kang. 2009. Generating 3D architectural models based on hand motion and gesture. *Computers in Industry* 60, 9 (2009), 677 – 685. https://doi.org/10.1016/j.compind.2009.05.001

[271] Emilie Yu, Rahul Arora, Tibor Stanko, J. Andreas Bærentzen, Karan Singh, and Adrien Bousseau. 2021. CASSIE: Curve and Surface Sketching in Immersive Environments. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohoma, Japan) *(CHI '21)*. ACM, New York, NY, USA. https://doi.org/10.1145/3411764.3445158

[272] Run Yu and Doug A Bowman. 2018. Force Push: Exploring Expressive Gesture-to-Force Mappings for Remote Object Manipulation in Virtual Reality. *Frontiers in ICT* 5 (2018), 25. https://doi.org/10.3389/fict.2018.00025

[273] Shanxin Yuan, Guillermo Garcia-Hernando, Björn Stenger, Gyeongsik Moon, Ju Yong Chang, Kyoung Mu Lee, Pavlo Molchanov, Jan Kautz, Sina Honari, Liuhao Ge, Junsong Yuan, Xinghao Chen, Guijin Wang, Fan Yang, Kai Akiyama, Yang Wu, Qingfu Wan, Meysam Madadi, Sergio Escalera, Shile Li, Dongheui Lee, Iason Oikonomidis, Antonis Argyros, and Tae-Kyun Kim. 2018. Depth-Based 3D Hand Pose Estimation: From Current Achievements to Future Goals. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/CVPR.2018.00279

[274] Youyi Zheng, Han Liu, Julie Dorsey, and Niloy J. Mitra. 2016. SmartCanvas: Context-inferred Interpretation of Sketches for Preparatory Design Studies. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 37–48. https://doi.org/10.1111/cgf.12809

[275] Yixin Zhuang, Ming Zou, Nathan Carr, and Tao Ju. 2013. A general and efficient method for finding cycles in 3D curve networks. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–10. https://doi.org/10.1145/2508363.2508423

[276] Ming Zou, Tao Ju, and Nathan Carr. 2013. An algorithm for triangulating multiple 3D polygons. *Computer Graphics Forum* 32, 5 (2013), 157–166. https://doi.org/10.1111/cgf.12182