Master's Thesis

VR Space Drawing with Adjustable and Directional Brush

Yude Lu

Supervisor Haoran Xie

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Knowledge Science)

March, 2026

Abstract

Virtual reality (VR) has emerged as a promising platform for 3D shape modeling, offering immersive environments where users can directly manipulate objects in three-dimensional space. Compared to traditional desktop modeling that relies on 2D screen interfaces with mouse and keyboard, VR enables more natural and intuitive 3D modeling through immersive six-degrees-of-freedom (6DOF) interaction using VR controllers. However, existing VR brushes often use simple line-segment cross-sections and require demanding wrist twisting to control surface orientation, making continuous rolling direction control difficult to achieve for users.

To address these challenges, this thesis presents a VR spatial drawing system using the generalized cylinder paradigm, allowing users to customize the design of cross-sections and apply real-time morphing. The generalized cylinder paradigm generates volumetric geometry by sweeping 2D profiles along spatial paths, a natural approach that, when cross-section shapes and trajectories are appropriately defined, holds significant potential for application in virtual reality environments. The proposed system represents cross-sections as graph structures and trajectories as C^1-continuous Catmull-Rom splines. This thesis utilizes linear blend shapes for smooth interpolation between cross-section configurations during the drawing process. The proposed framework provides three interfaces: a drawing interface for spatial input and rolling control, where users place control points and adjust cross-section orientation; a base shape editor for creating custom 2D cross-section geometries; and a morph editor for defining shape deformation targets.

This thesis conducted two user studies to evaluate the system's effectiveness. Experiment 1 assessed system usability and creativity support with 12 participants completing creative modeling tasks, while Experiment 2 was a comparative preference study where 4 participants compared the proposed system with baseline ribbon brushes. Experiment 1 (N=12) achieved a Creativity Support Index (CSI) of 77.1/100 and a System Usability Scale (SUS) score of 67.29/100, indicating strong support for creative expression with acceptable system usability. In Experiment 2 (N=4), participants consistently preferred the proposed approach over baseline ribbon brushes (mean rating 4.25/5 vs. 2.13/5, 5-point Likert scale). It is verified that superior volumetric control and expressiveness are enabled by custom cross-sections.

These results demonstrate that the proposed framework provides considerable creativity support through flexible cross-section and morphing control

with acceptable usability. This work contributes a unified VR spatial drawing paradigm that integrates trajectory creation, shape editing, and real-time deformation for creative modeling and prototyping.

Keywords: Virtual Reality, Spatial Drawing, Generalized Cylinder Paradigm, Generalized Cylinder

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the background and motivation for this research, identifies the research questions addressed by this thesis, outlines the proposed approach and contributions, and provides an overview of the thesis organization.

1.1 Background and Motivation

Virtual reality (VR) has emerged as a promising platform for 3D shape modeling. Compared to desktop modeling that relies on mouse and keyboard with 2D screen interfaces, VR modeling leverages 6DOF controllers and immersive 3D environments, enabling more natural and intuitive spatial interactions. As discussed in a recent systematic review [5], VR-based modeling is generally easier to learn and more engaging than desktop approaches, with users showing higher preference for immersive 3D interaction.

The appeal of VR for creative 3D work stems from its ability to provide direct manipulation in three-dimensional space. Users can reach out and "touch" their creations, walk around them, and make modifications from any angle. This stands in contrast to traditional desktop 3D modeling, where artists must mentally translate 2D mouse movements into 3D object transformations, often switching between multiple viewports to achieve the desired result.

VR spatial drawing has attracted some research attention. Some research focuses on curve creation, for example Dynamic Dragging [6] introduces an adaptive drag line that adjusts based on curvature and drawing speed, helping users create more controlled curves in VR environments. Research systems such as SurfaceBrush [7] and AdaptiBrush [1] utilize ribbon-alike strokes to construct manifold 3D surfaces, while CASSIE [3] leverages free-hand

sketching with optimization to create connected curve networks. The potential of VR as a creative medium has also been demonstrated by commercial applications such as Tilt Brush [8], Gravity Sketch [9] and VRChat [2], which are used in workflows for concept design and rapid prototyping.

1.2 Problem Statement

Despite the advantages of VR for 3D modeling, existing VR spatial drawing tools face significant limitations. Line-based approaches, while expressive for artistic mark-making, produce simple stroke representations that lack volumetric structure. Curve network approaches require post-processing to generate complete surfaces and may not be easily to achieve the artist's intended surface details. These limitations motivate the use of ribbon-alike strokes, which offer a promising middle ground: they can directly generate surface geometry while the user draws in free forms, providing immediate visual feedback of the resulting 3D form.

However, from a geometric perspective, ribbon-alike strokes can be viewed as a special case of generalized cylinder generation, where a line segment (the cross-section) is swept along a trajectory. This connection to the generalized cylinder paradigm suggests the potential for extending ribbon-alike approaches to support richer cross-section shapes (Figure 1.1).

Indeed, research on generalized cylinder modeling has demonstrated the versatility of sweeping arbitrary cross-sections along paths. For instance, 3-Sweep [4] extracts generalized cylinders from single images, while desktop 3D modeling software such as Blender [10] provides comprehensive sweep functionality through curve modifiers. Some sweep-to-curve workflows in Blender rely on third-party add-ons (e.g. Curves-to-Mesh [11]) to simplify or automate the profile-to-path conversion. In Blender, users typically create a path curve defining the trajectory, then create a separate profile curve for the cross-section, apply a bevel or sweep modifier to extrude the profile along the path, and finally adjust parameters such as scale, twist, and taper to refine the result. This workflow offers precise control over the final geometry; however, the multi-step nature of the process, requiring users to switch between different editing modes and manage separate curve objects, stands in contrast to the more direct, immersive manipulation that VR environments can afford.

Existing VR systems for drawing ribbon-like strokes also face interaction challenges. In these approaches, users must simultaneously control two aspects while drawing: the spatial trajectory (through controller translation) and the cross-section orientation (through wrist rotation). This is similar to

a 2D freehand pencil tool, which requires continuous, uninterrupted motor control where any momentary lapse affects the entire stroke. In VR, the demands are even greater: users must maintain smooth, continuous control over both position and orientation in three-dimensional space, where small tremors or perceptual misjudgments become visually amplified. Due to anatomical constraints, the free-form shapes human postures can create are limited; for example, it is difficult to draw continuously twisted ribbon shapes (Figure 1.2), and the distance between the drawing point and the user's wrist amplifies any tremor during rotation. These factors make it extremely difficult to achieve continuous, smooth, and precise control over cross-section orientation during the drawing process.

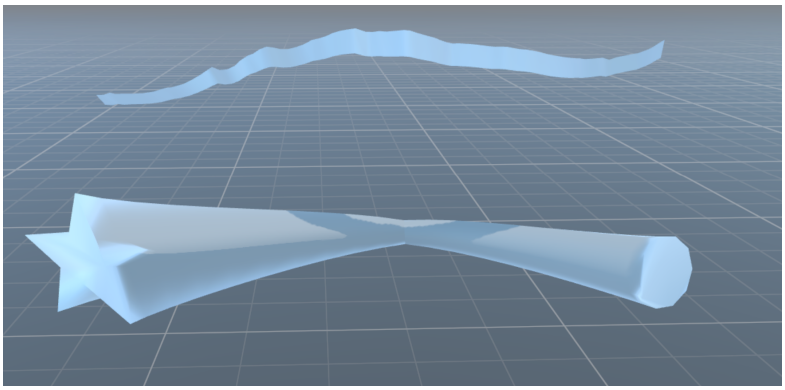


Figure 1.1: Ribbon shape vs. sweeping curves. The cross-section of ribbon brushes is essentially a fixed line segment, while allowing arbitrary cross-section shapes increases the diversity of drawable forms.

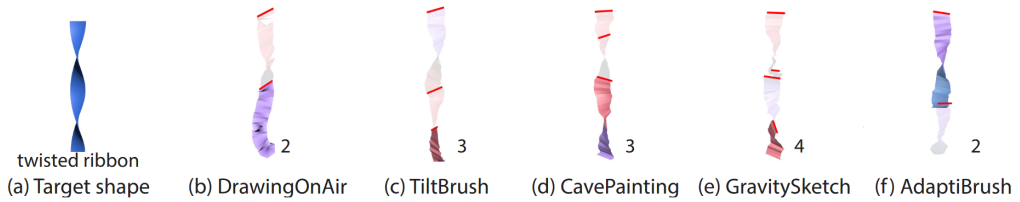


Figure 1.2: The shapes drawn by free-form brushes are limited by user posture, making it difficult to draw such twisted ribbon shapes. This result figure is referenced from AdaptiBrush [1].

1.3 Research Questions

From one perspective, existing ribbon-producing VR tools typically represent cross-sections as fixed line segments, limiting the diversity of creatable forms; furthermore, free-form brushes result in constrained drawn shapes. From the perspective of generalized cylinder modeling, there is no research that adequately applies generalized cylinder modeling methods in VR to manipulate both cross-sections and curve trajectories. Therefore, this thesis proposes three questions from two dimensions:

The first dimension of this research focuses on extending this representation to support arbitrary cross-section shapes:

1. **Cross-Section Flexibility:** Ribbon-alike strokes used in certain VR research are essentially a special case of generalized cylinders with line-segment cross-sections. Can we extend the cross-section to arbitrary shapes and allow real-time deformation during the drawing process?

2. **Cross-Section Control:** Current ribbon-alike approaches require users to simultaneously control trajectory and orientation through continuous wrist movement, making it difficult to achieve smooth, precise, and sustained control. How to provide continuous cross-section positioning and orientation control while maintaining usability, avoiding the need for users to simultaneously control trajectory and orientation in real-time with their wrist?

The second dimension addresses how to bring generalized cylinder modeling capabilities into the VR environment effectively:

3. **Intuitive VR Interaction for Generalized Cylinder Modeling:** How can we design sufficiently intuitive interaction methods to enable users to apply generalized cylinder modeling principles in VR, allowing direct manipulation of cross-sections and trajectories in an immersive 3D environment?

1.4 Proposed Approach

To address these challenges, this thesis presents a VR spatial drawing system that applies the generalized cylinder paradigm in virtual reality. Compared to the ribbon stroke methods used in some existing VR research, the proposed system extends the representation and interaction capabilities to support user-definable cross-sections and real-time shape morphing.

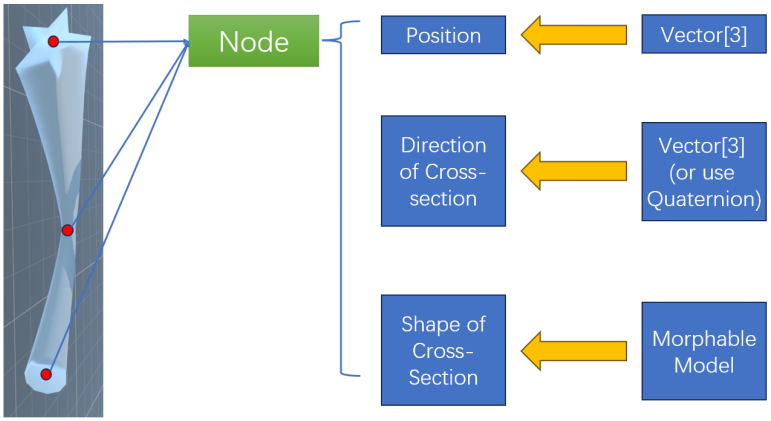


Figure 1.3: Proposed Approach. A scheme for representing generalized cylinder control points as a data representation.

The generalized cylinder method generates 3D surfaces by sweeping a 2D profile curve along a spatial path. In the proposed system, the generalized cylinder is realized through specific **representation and computation methods**, as in Figure 1.3. The components are categorized into shape definition (cross-section representation and morphing capability) and path definition (trajectory representation with orientation control).

- **Shape Definition (Cross-section, Morphing):** The profile curve utilizes a graph structure to describe arbitrary shapes. Linear blend shapes are applied to interpolate between target shapes for deformation.
- **Path Definition (Trajectory, Orientation):** The spatial path is defined by a sequence of control points interpolated using Catmull-Rom splines to form a smooth C^1-continuous curve. At each control point, the cross-section orientation is specified through rotation angles (particularly rolling), enabling precise control over how the cross-section twists along the trajectory.

The proposed system provides three user interfaces to support this paradigm. First, a drawing interface is provided where users place control points by pressing the trigger, specify the cross-section normal direction by dragging, and control rolling angles by rotating the controller (achievable through circular wrist motion or controller twisting). Additionally, a base shape editor is provided for users to create custom cross-section shapes and a morph editor

to define deformation targets. This approach directly addresses the research questions posed in Section 1.3 by enabling cross-section flexibility through the graph-based representation, providing continuous and precise orientation control through separated interaction phases, and offering intuitive VR interaction for generalized cylinder modeling through direct manipulation in an immersive 3D environment.

1.5 Contributions

The primary contributions of this thesis are listed as follows:

1. **A VR-based Generalized Cylinder Framework:** This thesis presents a novel framework for spatial drawing that integrates generalized cylinder modeling principles into a virtual reality environment. This framework enables users to create complex 3D volumes through intuitive sweep-based interactions, bridging the gap between precise CAD-like modeling and immersive freehand drawing.

2. **Enhanced Geometric Freedom through Cross-section and Orientation Control:** This thesis develops a graph-based representation for arbitrary cross-sections and implements separated orientation control with real-time morphing capabilities. This design significantly expands the diversity of 3D forms that users can create, moving beyond simple lines or fixed ribbon-alike shapes to complex, evolving volumetric geometries.

3. **Validation of Creativity Support in VR:** This thesis provides empirical evidence of the system's effectiveness through comprehensive user studies. The evaluation demonstrates that the system offers strong support for creative expression (CSI: 77.10/100) and is preferred by users over traditional ribbon-alike approaches due to its volumetric modeling capabilities and precise control over shape evolution.

1.6 Thesis Organization

The remainder of this thesis is organized as follows:

- **Chapter 2: Related Works** reviews existing approaches in VR spatial drawing and generalized cylinder modeling, identifying gaps that this work addresses.

- **Chapter 3: Preliminary Knowledge** introduces the mathematical and technical foundations including spline curves, generalized cylinders, blend shapes, and VR input.
- **Chapter 4: System Foundations** presents the core data structures and algorithms that support the user interface functionality described in Chapter 5, building upon the theoretical foundations from Chapter 3 to provide the implementation basis for real-time mesh generation and shape manipulation.
- **Chapter 5: User Interfaces** describes the three interaction interfaces: drawing interface, base shape editor, and morph editor.
- **Chapter 6: Implementation** details the technical implementation in Unity with OpenXR.
- **Chapter 7: Evaluation** presents the user study methodology and results, including two experiments assessing system usability (SUS: 67.29/100), creativity support (CSI: 4.083/5, 5-point Likert scale), and comparative user preferences.
- **Chapter 8: Conclusions** summarizes the contributions of this thesis and discusses future work directions.

Chapter 2

Related Works

This chapter reviews existing work in two main areas: VR spatial drawing systems and generalized cylinder modeling. The limitations of current approaches that motivate this research are identified.

2.1 VR Spatial Drawing

VR spatial drawing has attracted significant research attention in recent years. For example, Dynamic Dragging [6] focuses on curve creation, solving the problem of creating controlled curves in VR environments. If research and applications are organized based on geometric representation of created shapes, they can be categorized into the following classes:

2.1.1 Line-based Approaches

Some research focuses on curve creation, for example Dynamic Dragging [6] introduces an adaptive drag line that adjusts based on curvature and drawing speed, helping users create more controlled curves in VR environments. This technique helps users create more controlled curves by automatically adjusting the input sensitivity based on the complexity of the intended shape.

Commercial applications such as Tilt Brush [8] and Gravity Sketch [9] enable users to paint and sketch in 3D space with various brush styles. These systems provide multiple brush types. Although these brushes provide various visual styles, some of them are fundamentally simple strokes generated by sampling controller positions along a trajectory, belonging to line representation visually. Some of those applications also include ribbon-style brushes among their brush options, which we discuss in Section 2.1.3. VRChat also

provides line-based brushes [2] (Figure 2.1). As a popular virtual reality social application, VRChat provides line-based brushes in many scenarios to allow users to draw freely in space.

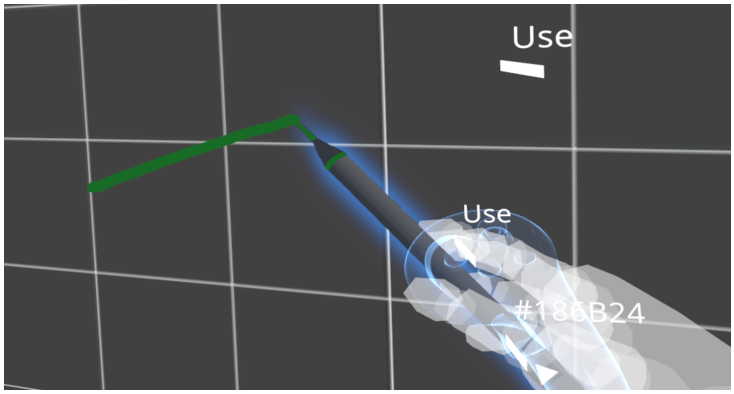


Figure 2.1: VRChat allows users to draw in space using pen-like tools; visually this appears as a free curve (screenshot). [2]

While stroke-based approaches excel at expressive mark-making and artistic exploration, they generally do not produce watertight 3D meshes suitable for further geometric processing or fabrication.

2.1.2 Curve Network Approaches

CASSIE [3] leverages free-hand mid-air sketching with a 3D optimization framework to create connected curve networks. The system infers surface patches from the user's curve network, providing a bridge between freehand sketching and structured surface modeling (Figure 2.2).

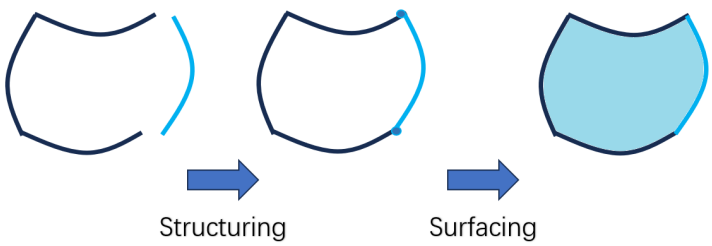


Figure 2.2: Overview of the method of CASSIE, adapted from CASSIE [3].

Inspired by FiberMesh [12], RodMesh [13] uses two-handed bending and stretching of virtual rods, allowing users to define outline shapes that are

subsequently inflated into manifold mesh surfaces. This approach leverages bimanual interaction for more intuitive control over curve shapes.

Curve network approaches provide more structured output than pure stroke-based methods, but they typically require post-processing to generate complete surfaces and may not preserve the artist's intended cross-sectional details.

2.1.3 Ribbon-structure based Approaches

CavePainting [14] (2001) pioneered ribbon-based 3D drawing in immersive CAVE environments, using physical props and gestures for natural interaction. SurfaceBrush [7] and AdaptiBrush [1] are two representative systems that utilize ribbon-alike strokes with controllable ruling directions as a medium for constructing manifold 3D surfaces (Figure 2.3). Compared to ribbon brushes where position and rotation are completely bound to the controller, SurfaceBrush [7] proposed the concept of using ribbon-alike strokes to build manifold surfaces that are automatically stitched together to create watertight meshes. AdaptiBrush [1] decouples the orientation control part, employing predictive algorithms based on stroke trajectory and interaction history to predict the user's ruling direction, expanding the space of shapes users can draw, thereby reducing physical fatigue during drawing. The proposed system reproduces this ribbon-alike brush functionality and extends it with additional capabilities.

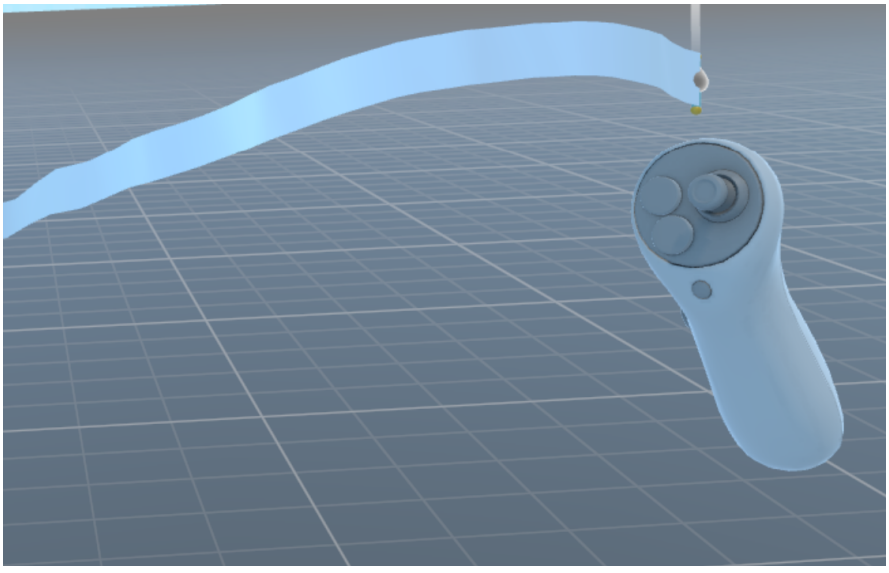


Figure 2.3: An example of brushes drawing ribbon-alike shapes.

In these systems that produce ribbon-alike shapes, the orientation of the cross-section is controlled through both the translational movement and rotational posture of the controller. However, since users must control the wrist rotation while simultaneously moving to draw the trajectory, and given that the wrist cannot rotate continuously beyond certain angles and that distance from the wrist amplifies tremor, achieving continuous and precise orientation control remains challenging. This interaction model resembles freehand pencil drawing rather than deliberate anchor-point placement: once the trigger is pressed, users must maintain uninterrupted motor control, where any momentary lapse in attention or slight hand tremor affects the smoothness of the entire stroke. In VR's three-dimensional space, such perceptual and motor errors become visually amplified, making it even more difficult to achieve the smooth, precise orientations needed for creating intentionally twisted shapes or maintaining consistent surface normals. Additionally, these ribbon-alike strokes are restricted to line-segment cross-sections, limiting the diversity of shapes users can create.

The proposed system addresses both limitations by supporting arbitrary cross-section shapes and providing continuous, smooth, and precise orientation control through a separated interaction paradigm where orientation is adjusted after position placement.

2.1.4 Multi-Viewpoint Approaches

Beyond brush-based approaches, some research addresses the fundamental perceptual challenges of VR spatial creation. MultiBrush [15] takes a different approach by providing multiple synchronized viewpoints during painting. Rather than modifying the brush or stroke representation, MultiBrush enables users to see their work from multiple angles simultaneously, helping them better perceive spatial relationships and achieve greater precision. This addresses the depth perception problem that plagues single-viewpoint VR drawing: users often misjudge distances, and switching viewpoints frequently interrupts the creative flow. While MultiBrush focuses on perceptual enhancement rather than geometric representation, it highlights the importance of addressing spatial perception challenges in VR creative tools, a consideration that informs the design of clear visual feedback during the drawing process.

2.2 Generalized Cylinder Modeling

Beyond the VR spatial drawing research discussed above, a substantial body of work has explored generalized cylinder modeling methods for 3D shape creation. Generalized cylinder modeling, a concept introduced by Binford for visual perception and object recognition in computer vision [16], creates 3D surfaces by sweeping a 2D profile along a spatial path. This approach, also known as sweep-based surface generation, has been widely adopted in CAD systems and sketch-based modeling [17], proving versatile across domains from early computer vision applications to interactive modeling interfaces.

The geometric foundation, sweeping diverse cross-section shapes along trajectories, demonstrates the potential for expressive shape creation. However, existing implementations of generalized cylinder modeling operate primarily in 2D desktop interfaces, where users perform operations through mouse and keyboard, making it difficult to achieve the intuitive, direct manipulation that VR's immersive 3D environment can provide.

2.2.1 Image-based Extraction

3-Sweep [4] introduced an innovative method for extracting editable 3D objects from a single photograph. Users simply drag three times on an image to specify the geometry and main directions of an object's cross-section, and the system generates a 3D model by sweeping the inferred profile, as illustrated in Figure 2.4.

While powerful for object extraction, this approach has two key limitations: it requires an image as reference input, limiting creative freedom; and the post-extraction editability is limited, as users can only adjust scale, rotation, and position of the extracted objects, but cannot modify detailed shape geometry.

2.2.2 Desktop CAD Approaches

Desktop 3D modeling software such as Blender [10] provides comprehensive sweep functionality through curve modifiers (Figure 2.5). The typical workflow involves creating a path curve that defines the trajectory, then creating a separate profile curve for the cross-section geometry. Users then apply a bevel or sweep modifier to extrude the profile along the path, and finally adjust parameters such as scale, twist, and taper to achieve the desired result. This workflow offers precise parametric control over the final geometry and supports arbitrary cross-section shapes. However, the multi-step nature

Figure 2.4: Schematic of the 3-Sweep method. The method reconstructs models from images using three drags; image generated by Gemini, method adapted from 3-Sweep [4].

of the process, requiring users to switch between different editing modes, manage separate curve objects, and adjust numerical parameters, represents a fundamentally different interaction paradigm from the direct, immersive manipulation that VR environments can afford.

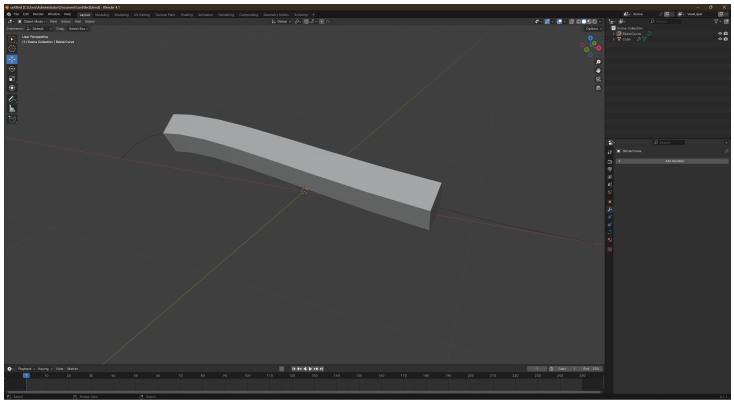


Figure 2.5: Shapes created in Blender using sweep modelling.

2.2.3 VR Sweep Modeling

From the perspective of generalized cylinder modeling, the VR research approaches that employ ribbon-alike strokes (such as SurfaceBrush [7] and AdaptiBrush [1], discussed in Section 2.1.3) can be viewed as a special case where the cross-section is fixed to a line segment. These systems focus on controlling the ruling direction of this line segment during stroke creation, but do not allow users to define arbitrary cross-section shapes.

Lift-Off [18] is a VR 3D modeling system, representing an early exploration of VR sweep modeling. Users can interactively select a pair of points from imported reference sketches and lift 2D curves into 3D space; users can sweep a surface by moving the two endpoints of a line along two curves. From the perspective of surface generation, it is similar to the CASSIE [3] method, but its surface is generated by dragging rather than direct generation, so it can also be considered as a drag trajectory formed by a changing line segment along the endpoints on the two curves. It relates to the generalized cylinder method, but it lacks a clear drag trajectory and does not have a clear definition of the drag cross-section shape.

Therefore, to the best of the author's knowledge, no existing VR sweep modeling system provides users with the capability to define arbitrary cross-section shapes beyond line segments, apply shape morphing along the trajectory, and exercise continuous and precise cross-section orientation control

through controller rotation simultaneously.

The proposed system brings generalized cylinder modeling into VR with a more intuitive interaction paradigm, allowing users to directly design cross-sections and control their deformation during drawing.

2.3 Comparison with Related Works

Table 2.1 summarizes key characteristics of related approaches, comparing their interaction environments, cross-section/shape variation flexibility, and orientation control mechanisms.

Table 2.1: Comparison with related works. Systems marked with † indicate our reproduced implementations used as baselines in Experiment 2, not the original commercial software.

System	**Env.**	**Cross-section / Variation**	**Orientation Control**
Tilt Brush [8]†	VR	Line segment / None	Wrist pose
Gravity Sketch [9]†	VR	Line segment / None	Wrist pose
AdaptiBrush [1]	VR	Line segment / None	Wrist + history
Lift-Off [18]	VR	From image	N/A
3-Sweep [4]	Desktop	From image	N/A
Blender [10]	Desktop	Arbitrary curve / Scale/taper	Parameter

Existing VR spatial drawing systems (such as Tilt Brush and Gravity Sketch reproduced in AdaptiBrush) provide predefined stroke/ribbon styles, and do not provide explicit orientation control parameters or arbitrary cross-section customization. Research systems employing ribbon-alike strokes (SurfaceBrush, AdaptiBrush) allow implicit orientation control through wrist posture (and in AdaptiBrush also including predictive behavior based on interaction history), but involve only line-segment cross-sections. Desktop generalized cylinder modeling methods (3-Sweep, Blender) support diverse cross-sections, but require multi-step 2D interface interactions, and are not VR-native.

An ideal VR spatial drawing system for generalized cylinder modeling should operate in VR's immersive environment, support variable cross-sections that can evolve along the trajectory, and provide continuous, smooth, and precise cross-section orientation control, enabling users to accurately create intentionally twisted shapes or maintain consistent surface normals without

the challenges of simultaneous trajectory and orientation control. This research aims to realize this vision by combining the immersive interaction of VR with the flexibility of arbitrary cross-sections and explicit orientation control, bridging these two research directions.

Chapter 3

Preliminary Knowledge

This chapter provides the theoretical background necessary for understanding the proposed system. The mathematical foundations of spline curves, generalized cylinder representation, and blend shapes, which form the core of the approach, are introduced.

3.1 Spline Curves

Spline curves are piecewise polynomial functions that provide smooth interpolation through control points. They are fundamental to computer graphics and geometric modeling.

3.1.1 Parametric Curves

A parametric curve in 3D space is defined as a function $\mathbf{C} : [0, 1] \to \mathbb{R}^3$ that maps a parameter t to a position:

$$\mathbf{C}(t) = (x(t), y(t), z(t)) \tag{3.1}$$

where $x(t)$, $y(t)$, and $z(t)$ are scalar functions representing the coordinates of the curve position in three-dimensional space, and $t \in [0, 1]$ is the curve parameter.

The derivatives of a parametric curve provide important geometric information:

- **Tangent**: $\mathbf{T}(t) = \frac{d\mathbf{C}}{dt}$ indicates the direction of motion along the curve
- **Curvature**: $\kappa(t) = \frac{\|\mathbf{C}' \times \mathbf{C}''\|}{\|\mathbf{C}'\|^3}$ measures how sharply the curve bends, where $\mathbf{C}' = \frac{d\mathbf{C}}{dt}$ is the first derivative (tangent vector) and $\mathbf{C}'' = \frac{d^2\mathbf{C}}{dt^2}$ is the second derivative (curvature vector)

3.1.2 Catmull-Rom Splines

Catmull-Rom splines [19] are a type of cubic interpolating spline that passes through all control points. Given a sequence of control points $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_{n-1}$, the curve between $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$ is defined as:

$$\mathbf{C}(t) = \frac{1}{2} \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{i-1} \\ \mathbf{P}_i \\ \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \end{bmatrix} \tag{3.2}$$

The tangent vector at any point along the curve can be computed by taking the derivative:

$$\mathbf{C}'(t) = \frac{1}{2} \begin{bmatrix} 0 & 1 & 2t & 3t^2 \end{bmatrix} \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{i-1} \\ \mathbf{P}_i \\ \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \end{bmatrix} \tag{3.3}$$

Catmull-Rom splines have several desirable properties:

1. **Interpolation**: The curve passes through all control points
2. C^1 **Continuity**: The tangent is continuous at control points
3. **Local Control**: Moving a control point only affects the four adjacent segments

These properties make Catmull-Rom splines ideal for interactive editing, as users can modify individual control points without affecting distant portions of the curve. In the proposed system, Catmull-Rom interpolation is used to compute both the position and tangent direction at sample points along the trajectory, ensuring smooth and predictable curve behavior.

3.2 Generalized Cylinders

A generalized cylinder is a swept surface generated by moving a 2D cross-section along a 3D trajectory while potentially varying its size, orientation, and shape.

3.2.1 Mathematical Definition

Formally, a generalized cylinder is defined by:

- A **trajectory curve** $\mathbf{C}(t) : [0, 1] \to \mathbb{R}^3$
- A **cross-section function** $\mathbf{S}(t, \phi) : [0, 1] \times [0, 2\pi) \to \mathbb{R}^2$
- A **local coordinate frame** defined at each point along the trajectory

The surface is parameterized as:

$$\mathbf{P}(t, \phi) = \mathbf{C}(t) + \mathbf{R}(t) \cdot \mathbf{S}(t, \phi) \tag{3.4}$$

where $\mathbf{R}(t)$ is a rotation matrix that aligns the cross-section with the local frame.

3.2.2 Frenet-Serret Frame

The classical approach to defining the local frame uses the Frenet-Serret frame:

- **Tangent**: $\mathbf{T} = \frac{\mathbf{C}'}{\|\mathbf{C}'\|}$
- **Normal**: $\mathbf{N} = \frac{\mathbf{T}'}{\|\mathbf{T}'\|}$
- **Binormal**: $\mathbf{B} = \mathbf{T} \times \mathbf{N}$

However, the Frenet-Serret frame is undefined at points of zero curvature and can exhibit sudden flips at inflection points.

3.2.3 Rotation Minimizing Frames

Rotation Minimizing Frames (RMFs) address the limitations of the Frenet-Serret frame by minimizing the rotation of the frame around the tangent as the curve is traversed. While the Frenet-Serret frame can exhibit sudden flips at inflection points, the RMF provides more stable orientation behavior.

In the proposed system, a simplified approach is adopted: instead of computing a full RMF, spherical linear interpolation (Slerp) is used to smoothly blend the user-specified orientation at each control point, and project the resulting up vector onto the plane perpendicular to the tangent to ensure the frame remains properly oriented. This approach provides stable cross-section orientation without the computational cost of full RMF computation.

In the workflow, this frame construction is used when generating the swept surface: for each trajectory sample a tangent direction is obtained, an up direction is interpolated from neighboring control points (derived from controller orientation/rolling input), and then a rotation is constructed (e.g., via LookRotation($\mathbf{t}, \mathbf{up}$)) to orient the cross-section before applying the world-space transformation (Equation 4.4).

3.3 Blend Shapes

Blend shapes (also known as morph targets or shape keys) are a deformation technique widely used in character animation and facial animation. A foundational work in this area is the 3D Morphable Model by Blanz and Vetter [20], which demonstrated the power of linear shape blending for synthesizing 3D faces. While their work operates on 3D facial geometry with hundreds of basis shapes learned from scanned data, the underlying mathematical principle, representing shape variations as weighted linear combinations of basis configurations, is directly applicable to 2D cross-section morphing. In the proposed system, users manually define a small number of morph targets rather than learning them from data, but the interpolation mechanism follows the same linear blending formulation.

3.3.1 Linear Blend Shapes

Given a base shape with vertices $\{\mathbf{b}_j\}_{j=0}^{m-1}$ and k morph targets with displacement vectors $\{\Delta_j^{(i)}\}$, the blended shape is computed as:

$$\mathbf{v}_j = \mathbf{b}_j + \sum_{i=1}^{k} \lambda_i \cdot \Delta_j^{(i)} \tag{3.5}$$

where λ_i are the blend weights. This formulation is additive, meaning multiple morph targets can be combined.

3.3.2 Properties

Linear blend shapes have several useful properties:

- **Linearity**: The result is a linear combination of the base and morph shapes
- **Efficiency**: Computation is fast, requiring only weighted vector addition
- **Intuitiveness**: Blend weights directly control the amount of each deformation
- **Composability**: Multiple morph targets can be active simultaneously

In the proposed system, blend shapes are applied to 2D cross-sections, where each morph target represents a deformation of the base polygon.

3.4 Virtual Reality Input

Virtual reality systems provide rich input modalities that differ significantly from traditional desktop interfaces.

3.4.1 Pose Input

Modern VR controllers provide six degrees of freedom (6DoF) tracking:

- Three translational degrees: position (x, y, z)
- Three rotational degrees: orientation (roll, pitch, yaw)

This enables natural 3D input where controller motion directly maps to virtual object manipulation. The pose information is fundamental to the drawing interface, as it determines the placement and orientation of control points.

3.4.2 Other Controller Inputs

In addition to pose tracking, VR controllers typically provide:

- Trigger button (analog or digital)
- Grip button
- Thumbstick or touchpad
- Face buttons (A, B, X, Y)

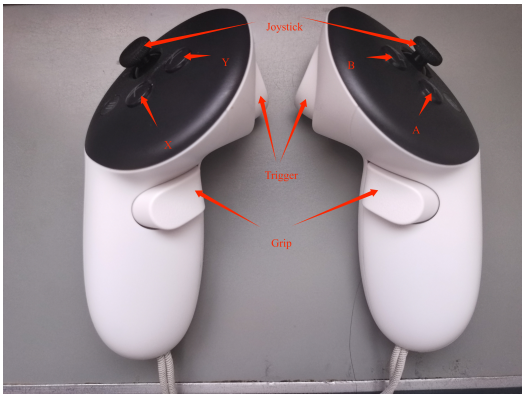


Figure 3.1: Input buttons on controllers of Meta Quest 3s.

Figure 3.1 labels the controller buttons used by the proposed interaction mappings.

The proposed system maps these inputs to drawing actions, menu navigation, and shape editing operations. For example, the trigger initiates control point placement, while thumbstick input adjusts morph target weights.

Chapter 4

System Foundations

This chapter describes the core data structures and algorithms of the proposed system. The focus is on the hierarchical representation of morphable brush shapes and the multi-parameter interpolation framework that enables real-time 3D mesh generation. These foundations support the user interface functionality described in Chapter 5, building upon the theoretical concepts introduced in Chapter 3.

4.1 Brush Shape Representation

A linear blend shapes approach is adopted for morphable cross-sections. This representation provides flexibility for arbitrary topologies (open/closed/branching) while enabling smooth interpolation for morphing effects.

4.1.1 Graph-based Geometry

Each brush shape is represented as a 2D graph $\mathcal{S} = (V, E)$ stored in Cartesian coordinates, where $V = \{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{m-1}\}$ are vertices with 2D coordinate attributes (x, y), and E defines edge connectivity between vertices. This graph-based representation is well-suited for this purpose because vertices in the graph directly correspond to mesh vertices in the generated cross-section, while edges define the mesh connectivity. Each vertex requires 2D coordinates since cross-sections are defined on a plane perpendicular to the trajectory.

Unlike standard closed-loop polygon representations, this graph-based structure supports:

- **Open polylines:** Useful for ribbon-like or shell-like structures.

- **Branching structures:** Allowing for complex cross-sections resembling multi-pronged forms.
- **Non-manifold configurations:** Supporting intersecting edges and shared vertices for decorative patterns.

4.1.2 Linear Blend Shapes for Morphing

To enable smooth shape transitions during the drawing process, the linear blend shapes technique introduced in Chapter 3 is applied to the 2D cross-sections. For shape morphing, a base shape $\mathcal{S}_0 \in \mathbb{R}^{m\times 2}$ and n morph targets $\{\mathcal{S}_1, \ldots, \mathcal{S}_n\} \in \mathbb{R}^{m\times 2}$ are stored, where m is the number of vertices in each shape. All shapes must share the same vertex count and graph topology to ensure valid interpolation. The morphed vertex position $\mathbf{p}_j$ (the j-th vertex of the resulting cross-section) is computed as a weighted combination:

$$\mathbf{p}_j = \mathbf{o}_j + \sum_{i=1}^{n} \lambda_i \cdot (\mathbf{m}_{i,j} - \mathbf{o}_j) \tag{4.1}$$

where $\mathbf{o}_j \in \mathbb{R}^2$ is the j-th vertex of the base shape, $\mathbf{m}_{i,j} \in \mathbb{R}^2$ is the j-th vertex in morph target i, and $\lambda_i \in [0, 1]$ is the morph weight controlling the influence of morph target i.

This formula computes the final vertex position by starting from the base shape vertex $\mathbf{o}_j$ and adding a weighted combination of displacement vectors. Each term $\lambda_i \cdot (\mathbf{m}_{i,j} - \mathbf{o}_j)$ represents the displacement from the base vertex to the corresponding morph target vertex, scaled by the morph weight λ_i. When $\lambda_i = 0$, morph target i has no effect; when $\lambda_i = 1$, the full displacement is applied. The summation allows multiple morph targets to be combined simultaneously, enabling complex shape variations by blending different deformation modes.

4.2 Adaptive Spline Framework

The system manages a sequence of control points, each serving as a multi-dimensional data container for trajectory and appearance parameters.

4.2.1 Control Point Data Structure

A control point P_i is defined as a tuple:

$$P_i = \langle \mathbf{x}_i, \mathbf{q}_i, s_i, \boldsymbol{\lambda}_i \rangle \tag{4.2}$$

where:

- $\mathbf{x}_i \in \mathbb{R}^3$: World-space position.
- $\mathbf{q}_i \in \mathbb{H}$: Orientation represented as a unit quaternion.
- $s_i \in \mathbb{R}^+$: Local scale (stroke size).
- $\boldsymbol{\lambda}_i \in [0,1]^n$: Vector of morph weights.

4.2.2 Catmull-Rom Spline Trajectory

The system connects control points using Catmull-Rom splines for smooth trajectory representation. For position interpolation, the standard matrix form is used:

$$\mathbf{C}(t) = \frac{1}{2}\begin{bmatrix}1 & t & t^2 & t^3\end{bmatrix}\begin{bmatrix}0 & 2 & 0 & 0\\ -1 & 0 & 1 & 0\\ 2 & -5 & 4 & -1\\ -1 & 3 & -3 & 1\end{bmatrix}\begin{bmatrix}\mathbf{X}_{i-1}\\ \mathbf{X}_i\\ \mathbf{X}_{i+1}\\ \mathbf{X}_{i+2}\end{bmatrix} \tag{4.3}$$

where $t \in [0,1]$ and $\mathbf{X}_k$ are the world-space positions of the control points $P_{i-1}, P_i, P_{i+1}, P_{i+2}$ respectively.

Catmull-Rom splines were chosen over other spline types for several reasons:

- **Interpolation:** The curve passes directly through all control points, making spatial editing intuitive for the user.
- **Locality:** Each segment is influenced only by four neighboring points, ensuring that modifications do not cause global distortion.
- C^1 **Continuity:** The resulting curves have continuous first derivatives, providing sufficient smoothness for high-quality mesh generation without the computational overhead of higher-order splines.

4.2.3 Boundary Handling with Virtual Nodes

Catmull-Rom splines require four control points for each segment, but at trajectory endpoints, the outer neighbors do not exist. To ensure C^1 continuity at the boundaries, virtual boundary nodes are maintained (referred to as head and tail nodes in the implementation). If not explicitly set (e.g., when a stroke is just beginning), these default to repeating the first/last point position, effectively setting $\mathbf{X}_{\text{head}} = \mathbf{X}_0$ and $\mathbf{X}_{\text{tail}} = \mathbf{X}_{n-1}$. This approach ensures stable interpolation at the edges without requiring extra user input.

4.3 Real-time Mesh Generation

The 3D surface is generated by sweeping interpolated cross-sections along the spline trajectory.

4.3.1 Multi-parameter Interpolation

In the generalized cylinder mesh generation process, synchronized interpolation is performed across all parameters at each sample $t \in [0, 1]$ within a segment $[P_i, P_{i+1}]$. This multi-dimensional interpolation transforms discrete control nodes into a continuous 3D surface:

- **Position $\mathbf{X}(t)$:** Computed using the Catmull-Rom spline formula (Equation 4.3).
- **Orientation $\mathbf{Q}(t)$:** Interpolated via Spherical Linear Interpolation (Slerp) between $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ to preserve rotation behavior.
- **Appearance $\langle s(t), \boldsymbol{\lambda}(t) \rangle$:** Interpolated linearly (Lerp) between corresponding values at P_i and P_{i+1}.

Algorithm 1 formally defines the stroke generation process.

4.3.2 World-Space Instantiation

The final world-space vertex $\mathbf{V}_j$ at sample t is calculated by transforming the morphed local vertex $\mathbf{p}_j(t)$ (Equation 4.1) using the local frame derived from the quaternion $\mathbf{Q}(t)$:

$$\mathbf{V}_j = \mathbf{X}(t) + s(t) \cdot (\mathbf{Q}(t) \otimes \mathbf{p}_{j,3\mathrm{D}}(t) \otimes \mathbf{Q}(t)^{-1}) \tag{4.4}$$

where $\mathbf{p}_{j,3\mathrm{D}}(t) = [p_{j,x}(t), p_{j,y}(t), 0]^T$ is the 2D cross-section coordinate mapped to the XY plane, and $\otimes$ denotes quaternion multiplication. This approach ensures that the cross-section is always correctly oriented and scaled along the 3D trajectory.

4.3.3 Triangulation

The connectivity of the 3D mesh is directly derived from the cross-section graph E. For each edge $(a, b) \in E$, a quad is constructed between consecutive samples i and $i + 1$. Specifically, two triangles are generated connecting the world-space vertices $(\mathbf{V}_{a,i}, \mathbf{V}_{b,i}, \mathbf{V}_{a,i+1}, \mathbf{V}_{b,i+1})$. This edge-based sweep approach naturally handles non-closed topologies, branching structures, and intersecting edges, ensuring the generated 3D surface strictly adheres to the user's geometric definition.

Algorithm 1 Stroke Mesh Generation

Require: Control points $\{P_0, \ldots, P_{n-1}\}$ with attributes $\mathbf{x}, \mathbf{q}, s, \boldsymbol{\lambda}$
Require: Base shape $\mathcal{S}_0$, morph targets $\{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$
Ensure: Triangle mesh $\mathcal{M}$ (vertices $\mathcal{V}$, triangles $\mathcal{T}$)

samples $\leftarrow \emptyset$
for each segment $i = 0$ to $n - 2$ **do**
    for $j = 0$ to subdivLevel **do**
        $t \leftarrow j/\text{subdivLevel}$
        $\mathbf{X}(t) \leftarrow \text{CatmullRom}(P_{i-1}, P_i, P_{i+1}, P_{i+2}, t)$
        $\mathbf{Q}(t) \leftarrow \text{Slerp}(\mathbf{q}_i, \mathbf{q}_{i+1}, t)$
        $s(t) \leftarrow \text{Lerp}(s_i, s_{i+1}, t)$
        $\boldsymbol{\lambda}(t) \leftarrow \text{Lerp}(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{i+1}, t)$
        samples.append$(\mathbf{X}(t), \mathbf{Q}(t), s(t), \boldsymbol{\lambda}(t))$
    end for
end for
for each sample s in samples **do**
    $\mathbf{p}_j(t) \leftarrow \mathcal{S}_0 + \sum \lambda_i(\mathcal{S}_i - \mathcal{S}_0)$ ▷ LBS Morphing
    $\mathcal{V}_j \leftarrow \text{Transform}(\mathbf{p}_j(t), \mathbf{X}(t), \mathbf{Q}(t), s(t))$ ▷ Equation 4.4
end for
Connect corresponding vertices in $\mathcal{V}$ using graph edges E to form triangles $\mathcal{T}$
return mesh $\mathcal{M}(\mathcal{V}, \mathcal{T})$

Chapter 5

User Interfaces

This chapter describes the three interaction interfaces that comprise the proposed system and the relationships between them from a data structure perspective. Each interface manages a distinct part of the hierarchical model, ensuring that changes to basic geometric definitions or deformation targets are consistently reflected in the final drawing environment.

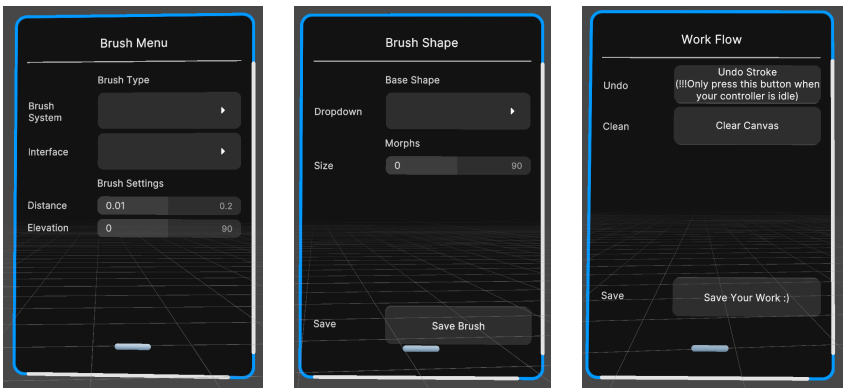


Figure 5.1: The Control Panel in VR. Users can use the dropdown list on the left to select the drawing interface or switch system modes. The sliders in the middle are used to adjust the brush size and morph weights. The panel on the right provides system operation shortcuts such as undo, save, and load.

Figure 5.1 shows the complete control panel layout used for mode switching and parameter adjustment during operation.

The relationship between user interaction and the geometric model is summarized in Figure 5.2. The core principle of the proposed system is the hierarchical decomposition of complex 3D strokes into a series of interpolated cross-sections. The base shape editor is used to define the fundamental

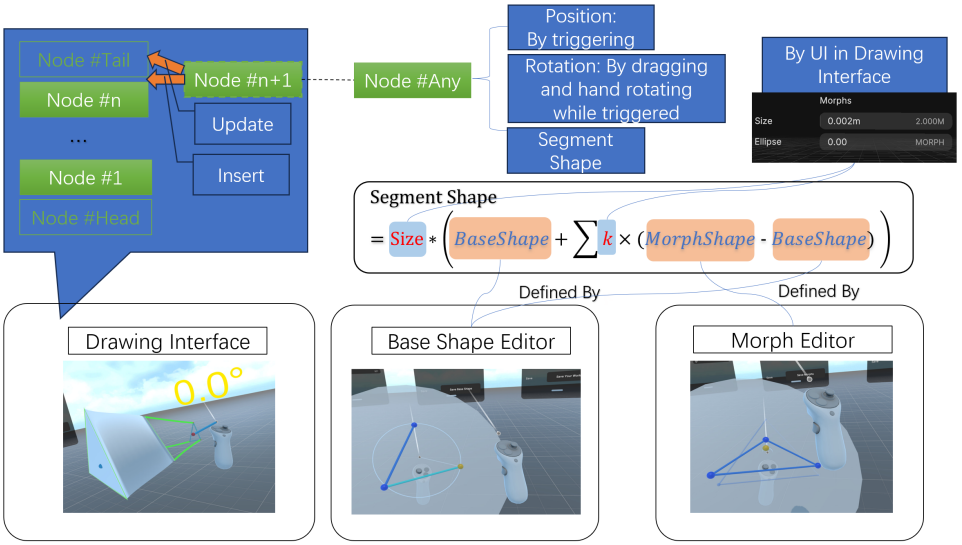


Figure 5.2: Three types of interaction interfaces and their relationship in data structure. The base shape editor and morph editor define the static geometric components (the graph structure and vertex offsets), which are then dynamically combined and swept along trajectories managed by the drawing interface.

topology and geometry of the cross-section, which is shared across all instances. The morph editor then defines displacement vectors for this shared topology. Finally, the drawing interface instantiates these components into a specific spatial context by defining the sweeping trajectory and per-point parameters.

5.1 Drawing Interface

The drawing interface shown in Figure 5.3 allows users to create 3D strokes by placing control points and controlling their twist angles. A sequence of control points, each with its associated position and rolling angle, defines the trajectory and orientation information needed to generate the complete 3D shape through the generalized cylinder sweeping process described in Chapter 4.

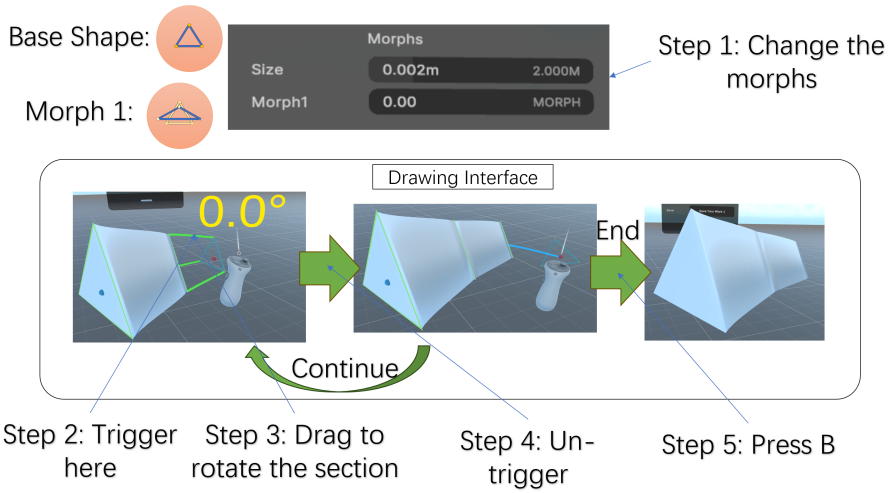


Figure 5.3: The drawing interface in VR, showing the stroke creation process.

5.1.1 Stroke Generation from Control Point Sequence

As users place control points, a sequence of control points $\{P_0, P_1, \ldots, P_{n-1}\}$ is built up, where each P_i stores the position, tangent direction, rolling angle, size, and morph weights. The mesh generation algorithm (described in Chapter 4) processes this sequence to:

1. Generate smooth trajectory curves using Catmull-Rom interpolation between control points
2. Compute sample points along each curve segment at a specified subdivision level
3. Interpolate size and morph weights linearly at each sample point
4. Apply the morphed cross-section at each sample point, oriented according to the interpolated rolling angle
5. Connect adjacent cross-sections to form a swept mesh surface

This process transforms the discrete control point sequence into a continuous 3D shape.

5.1.2 Control Point Placement and Rolling Direction Control

The drawing process follows a trigger-based workflow.

1. **Idle state:** When the trigger is not pressed, users can freely position the controller. A preview of the cross-section shape is displayed at the controller position, showing how the shape will appear when placed.

2. **Trigger press:** Pressing the trigger places a new control point at the current controller position. This marks the beginning of the control point definition.

3. **Rotation during placement:** Rotating the controller controls the rolling angle (twist) of the cross-section at that point. This is the key interaction that enables continuous rolling direction control.

4. **Release:** Releasing the trigger confirms the control point with its position and rolling angle. The mesh is regenerated to include the new control point.

This approach provides more precise control compared to continuous sampling methods used in tools like AdaptiBrush [1], though it requires more deliberate interaction.

Rolling Angle Computation. The rolling angle at each control point is determined by the controller's rotation. The system captures the controller's up direction at the moment of placement, which defines the local up vector for the cross-section at that control point. This up vector is stored with the control point and used during mesh generation to orient the cross-section correctly.

5.1.3 Real-time Deformation Adjustment

Morph weights λ_i and the size parameter s can be adjusted via UI sliders at any time during the drawing process. The current values are captured when the trigger is pressed, ensuring consistent cross-section parameters for each control point.

When generating the final mesh, different interpolation methods are used for different parameters:

- **Size interpolation:** The size parameter s at each sample point is computed using linear interpolation between adjacent control points: $s(t) = (1-t) \cdot s_i + t \cdot s_{i+1}$
- **Morph weight interpolation:** Each morph weight λ_j is also linearly interpolated: $\lambda_j(t) = (1-t) \cdot \lambda_{j,i} + t \cdot \lambda_{j,i+1}$

- **Position interpolation:** The trajectory position uses Catmull-Rom spline interpolation for smooth C^1-continuous curves, as described in Chapter 4

This design allows users to pre-set morph weights before starting a stroke for consistent appearance, vary morph weights between control points for gradual shape transitions, and experiment with different morph combinations in real-time.

5.2 Base Shape Editor

The base shape editor provides the interface for defining the fundamental cross-section geometry used in the drawing process. Through this editor, users design custom 2D cross-section shapes that will be swept along trajectories to form 3D strokes. This interface operates within the 3D VR space but provides a 2D editing canvas for cross-section definition.

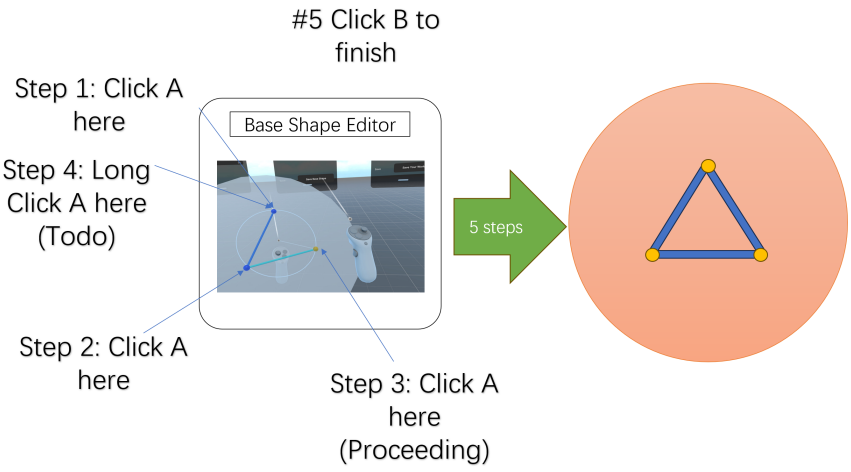


Figure 5.4: Base shape editor interface for creating graph-based cross-sections in VR.

Figure 5.4 illustrates the graph-based editing view used to construct cross-section topology.

5.2.1 Graph-based Shape Construction

The shape is represented as a graph $\mathcal{G} = (V, E)$ where vertices are connected by edges. The system supports three primary vertex operations:

1. **Place Vertex (A button short press):** Places a new vertex at the pointing position and automatically connects it to the previously placed vertex (if one exists), forming a continuous edge chain.

2. **Snap Vertex (A button long press):** Connects the most recently placed vertex to an existing vertex in the graph, completing a closed loop or creating a branch. This operation also updates the previous vertex pointer to the snapped vertex, so subsequent placements will connect from there.

3. **Clear Connection (B button):** Clears the previous vertex pointer, allowing the user to start a new disconnected component or branch without connecting to the previous vertex chain.

These operations enable flexible shape construction, supporting open polylines for ribbon-like or curved structures, closed polygons by snapping back to the starting vertex, branching structures such as stars or trees by clearing and restarting from a different vertex, and multiple disconnected components by using the clear connection operation.

5.2.2 Radial Disk Interface

The editor displays a circular disk attached to the left controller, providing a polar-coordinate-like visual representation. The disk serves as the editing canvas, with concentric circles and radial lines providing visual reference for shape proportions.

Users create vertices by pointing with the right controller and clicking. The right controller's position is projected onto the editing plane:

$$\mathbf{p}_{\text{proj}} = \mathbf{p}_{\text{ctrl}} - d \cdot \mathbf{n}_{\text{plane}} \tag{5.1}$$

where $\mathbf{p}_{\text{proj}}$ is the projected position on the editing plane, $\mathbf{p}_{\text{ctrl}}$ is the controller position, $\mathbf{n}_{\text{plane}}$ is the normal vector of the editing plane, and $d = (\mathbf{p}_{\text{ctrl}} - \mathbf{p}_{\text{plane}}) \cdot \mathbf{n}_{\text{plane}}$ is the signed distance from the controller to the plane (with $\mathbf{p}_{\text{plane}}$ being any point on the plane).

This projection ensures that vertices are always placed on the editing plane, regardless of the exact position of the right controller.

5.3 Morph Editor

The morph editor provides the interface for defining shape deformation targets that can be applied during the drawing process. By creating morph

targets, users specify alternative configurations of the cross-section vertices. During drawing, users can smoothly interpolate between the base shape and these morph targets by adjusting morph weights, enabling the cross-section to transform along the stroke trajectory. Morph targets must maintain the same vertex count and connectivity as the base shape to ensure valid interpolation.

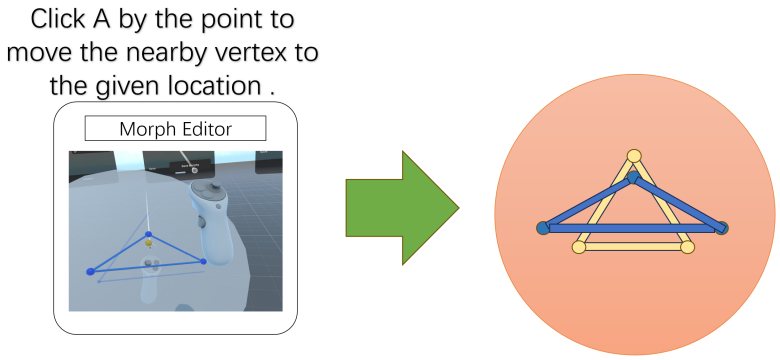


Figure 5.5: Morph editor interface for defining deformation targets of the cross-section. At the right part of the figure, the yellow shape is the base shape, and the blue shape is the morphed shape.

Figure 5.5 shows the dual-shape visualization used to define and preview morph target deformations.

5.3.1 Automatic Morph Computation

When saving, the system computes displacement vectors for each vertex:

$$\Delta_j = \mathbf{m}_j - \mathbf{o}_j, \quad j = 0, 1, \ldots, m-1 \tag{5.2}$$

where Δ_j is the displacement vector for vertex j, $\mathbf{m}_j \in \mathbb{R}^2$ is the position of vertex j in the morph target, $\mathbf{o}_j \in \mathbb{R}^2$ is the position of vertex j in the original base shape, and m is the total number of vertices.

These displacements are stored alongside the base shape, enabling additive blending during drawing (see Equation 4.1).

5.3.2 Dual Visualization System

The editor displays both the original base shape (as a semi-transparent shadow) and the editable copy simultaneously. This provides constant visual reference for the deformation magnitude, clear indication of which vertices correspond between base and target, and intuitive feedback for maintaining shape coherence.

5.3.3 Vertex Manipulation

Users move vertices by pressing the A button, which snaps the nearest vertex to the controller's projected position on the editing plane. This snap-based interaction provides direct control over vertex placement.

5.4 Controller Mapping

Table 5.1 summarizes the controller mappings for each interface.

Table 5.1: Controller Mappings by Interface

Interface	Action	Controller Input
Drawing	Place control point	Right trigger press
	Control rolling	Right controller rotation
	Adjust morph	UI sliders
	End stroke	Right B button
Base Shape	Add vertex	Right A button (short press)
	Select/connect vertex	Right A button (long press)
	End chain	Right B button
Morph Editor	Move vertex	Right A button

Chapter 6

Implementation Details

This chapter describes the implementation details of the proposed system, including the development platform, system architecture, and technical considerations for real-time performance in virtual reality environments.

6.1 Development Platform

The proposed system is developed using the Unity game engine (version 2022.3 LTS) [21] with the OpenXR framework [22] for cross-platform VR support. The system runs on a Windows PC, with the Meta Quest 3s headset serving as the display and input device via USB connection during development and experimentation.

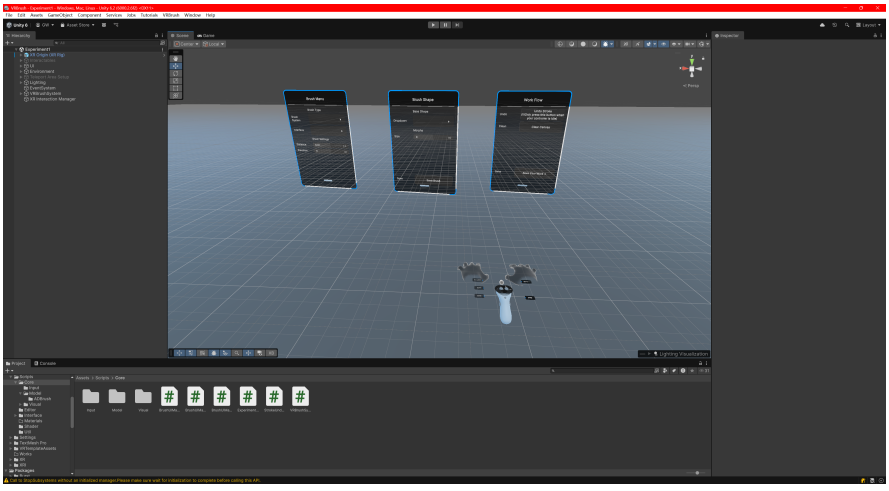


Figure 6.1: Development environment in the Unity Editor (screenshot).

Figure 6.1 shows the Unity-based development environment used to implement and test the system.

6.1.1 Hardware Configuration

The development and experimental environment consists of:

- **Host PC**: Windows desktop with dedicated GPU for rendering
- **VR Headset**: Meta Quest 3s connected via USB cable (Quest Link) for development and experimentation; Meta Quest 2 was also used during early development phases
- Touch controllers with 6DoF tracking, buttons, triggers, and thumbsticks

Since the application runs on the Windows PC rather than the Quest 3s's standalone processor, the system is not constrained by mobile hardware limitations. However, maintaining stable frame rates remains important for comfortable VR experience.

6.1.2 Software Framework

The system utilizes following Unity packages:

- **XR Interaction Toolkit**: Provides high-level abstractions for VR interactions, including ray-casting, grab interactions, and locomotion
- **XR Hands**: Enables hand tracking as an alternative input modality (though the current implementation primarily uses controllers)
- **OpenXR Plugin**: Ensures compatibility with various VR runtimes and devices

6.2 System Architecture

The overall software architecture of the proposed system is illustrated in the architectural overview diagram (Figure 6.2). This diagram depicts the system from a software engineering perspective, highlighting the modular decomposition and the relationships between functional components.

The proposed system follows a modular architecture with clear separation between data representation, interaction logic, and rendering.

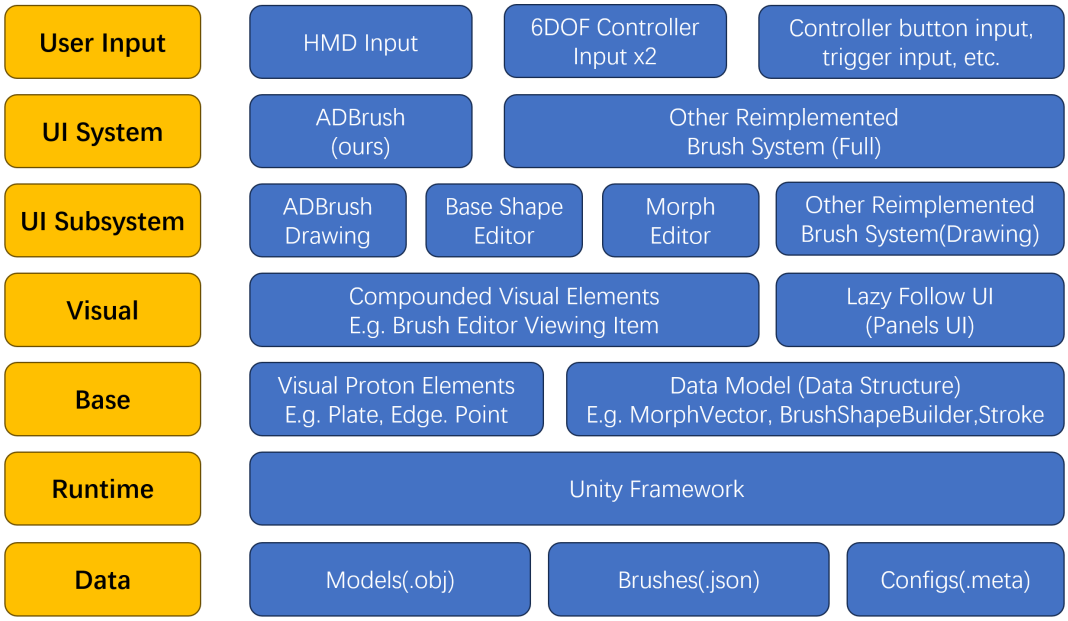


Figure 6.2: Software Framework. This schematic outlines the internal logic and engineering structure, showing the components in the program that interact within the Unity framework to handle VR input and persistent storage.

6.3 Shape File Format

The system uses a JSON-based format for shape persistence. Base shapes are stored as:

```
{
  "name": "custom_shape",
  "type": "cartesian",
  "points": [
    {"x": 0.5, "y": 0.0},
    {"x": 0.35, "y": 0.35},
    {"x": 0.0, "y": 0.5},
    ...
  ],
  "edges": [
    {"a": 0, "b": 1},
    {"a": 1, "b": 2},
    ...
  ]
```

```
}
```

Morph targets extend this format with displacement vectors:

```
{
  "name": "custom_shape_morph",
  "base": "custom_shape",
  "displacements": [
    {"dx": 0.1, "dy": -0.05},
    {"dx": 0.0, "dy": 0.1},
    ...
  ]
}
```

This data structure is determined by the system's functional requirements: it directly supports the base shape definition and morphing capabilities described in the previous chapters.

6.4 Model Export

The system allows users to export their created 3D models in OBJ format for use in other 3D modeling and rendering applications. This export functionality generates standard mesh geometry from the swept cross-sections, preserving the final triangulated mesh structure, enabling users to save their work, load previous sessions, and share creations with others.

Chapter 7

Evaluation

This chapter presents the evaluation of the proposed system through two experiments. Experiment 1 evaluates system usability and creative modeling capability (N=12). Experiment 2 is a comparative preference study (N=4, randomly selected from the 12 participants in Experiment 1) that compares the proposed system with baseline ribbon-alike VR drawing tools.

7.1 Study Design

To evaluate the proposed system, two experiments were conducted. In Experiment 1, participants completed creative modeling tasks using the full workflow of the proposed system after demonstration and practice. In Experiment 2, a subset of participants compared the proposed system against two baseline VR drawing tools and provided preference ratings.

7.1.1 Objectives

The evaluation aims to assess two complementary aspects of the system. First, the System Usability Scale (SUS) questionnaire measures the system's overall usability, including ease of use, learnability, and whether users feel confident operating the interface with rolling direction control. Second, the Creativity Support Index (CSI) questionnaire evaluates the system's effectiveness as a creative tool, examining whether the custom cross-section shapes and morphing capabilities support exploration, immersion, and expressive output. Together, these standardized instruments provide a comprehensive picture of both the pragmatic usability and the creative potential of the proposed VR spatial drawing system.

7.1.2 Participants

Twelve participants were randomly recruited as volunteers from the university for Experiment 1, aged between 20 and 30 years old. Participants had varying levels of experience with VR and 3D modeling. All participants provided informed consent before participating in the study. For Experiment 2, 4 participants from the Experiment 1 group were randomly invited to return for a comparative preference study.

7.1.3 Apparatus

The study is conducted using the setup shown in Figure 7.1:

- Windows PC with dedicated GPU
- Meta Quest 3s headset connected via USB (Quest Link). Quest 2 was used for some participants as a backup device due to connectivity issues with the host, likely caused by system updates.
- Touch controllers for 6DoF input
- Standing play area of 4m $\times$ 4m

7.1.4 Procedure

Each study session follows a structured procedure. Experiment 1 and Experiment 2 share the same introduction, demonstration, and practice setup; Experiment 2 is then conducted as an additional comparative phase for the selected subset of participants.

The session begins with an introduction and consent phase (approximately 5 minutes), providing an overview of the study purpose and consent process. This is followed by a demonstration phase where the experimenter demonstrates the system features, including the drawing interface (placing control points, rotating for rolling control), base shape editor (creating custom cross-sections), morph editor (defining deformation targets), and UI controls (adjusting size and morph weights). Participants then engage in a practice session (15-20 minutes) using the system with experimenter guidance to become familiar with the interaction methods. For Experiment 1, participants complete predefined creative modeling tasks using the full system workflow, followed by the CSI and SUS questionnaires. For Experiment 2, selected participants try three VR drawing systems (the proposed system and two baselines), then complete the preference questionnaire and provide qualitative reasoning.

Figure 7.1: Experimental setup showing the participant's environment in the study. The person photographed is the author.

7.2 Experimental Tasks

7.2.1 Experiment 1: Creative Modeling Tasks

In Experiment 1, participants used the full workflow of the proposed system (drawing interface, base shape editor, and morph editor) to complete creative modeling tasks. Each participant completed three drawings, choosing one subject per difficulty level (easy, medium, hard). The system recorded the created shapes for later analysis.

To reduce ambiguity and keep the task prompts consistent, the study presented the reference prompts in two groups using semi-transparent guide graphics in VR. After finishing each drawing, participants clicked **Next** using the left controller to proceed. The available subjects included grass or a cylinder for the easy level, a twig or beer bottle for the medium level, and a cannon or croissant for the hard level.

After completing the three drawings, participants filled out Questionnaire 1 (CSI) and Questionnaire 2 (SUS). The verbatim participant instruction text is provided in Appendix A.1.

7.2.2 Experiment 2: Comparative Preference Study

To gain further insights into the system's strengths relative to existing VR drawing tools, a second experiment was conducted. From the 12 participants in Experiment 1, 4 participants were randomly invited to return for a comparative preference study. Each participant used three different VR drawing systems to model the same subject (either a predefined prompt or a subject of their own choosing):

1. **The Proposed System** (ADBrush): The generalized cylinder-based system with custom cross-sections and rolling control.
2. **CavePainting-style ribbon brush (reproduced)**: A ribbon-alike brush was reproduced emulating the classic CavePainting approach [14] (originally designed for CAVE environments, an early VR system using surround projection screens rather than modern head-mounted displays). *Note: This is not the original commercial software but a custom implementation that replicates only the ribbon-stroke paradigm for controlled comparison.*
3. **GravitySketch-style ribbon brush (reproduced)**: A ribbon-alike brush was reproduced emulating the GravitySketch [9] stroke style. *Note: This is not the original commercial software but a custom implementation that replicates only the ribbon-stroke paradigm for controlled comparison.*

In these reproduced implementations, both baseline systems use fixed ribbon brushes with line-segment cross-sections. The only difference between the two baselines lies in how orientation is controlled: CavePainting-style uses direct wrist pose mapping, while GravitySketch-style incorporates slightly different orientation handling. Neither baseline allows custom cross-section shapes, providing a controlled comparison to highlight the advantages of the proposed generalized cylinder approach.

Participants could choose from existing task prompts or propose their own drawing subjects. After completing the same modeling task with each of the three systems, participants rated their experience using a 5-point Likert scale and provided qualitative reasoning for their preferences.

7.3 Data and Metrics

Questionnaires

Two standardized questionnaire instruments were used to evaluate the system. The Creativity Support Index (CSI) measures how well a system supports creative work across five dimensions: exploration, immersion, quality, intuition, and playfulness. The System Usability Scale (SUS) is a widely-used 10-item questionnaire that assesses overall system usability, including ease of use, learnability, and user confidence. For Experiment 2, preference ratings were additionally collected for each of the three systems using a 5-point Likert scale, along with qualitative reasoning for participants' preferences. The complete questionnaire items are provided in Appendix A.3.

7.4 Results

This section presents the results from our two experiments. Experiment 1 evaluated overall system usability and creative support with 12 participants completing creative modeling tasks. Experiment 2 compared our system with reproduced baseline ribbon-brush tools through a preference study with 4 participants.

7.4.1 Questionnaire Results

The mean ratings for the CSI and SUS questionnaires are summarized in Figure 7.2 and Figure 7.3, respectively.

During Experiment 1, participants created diverse 3D shapes using our system across three difficulty levels. Figure 7.4 shows representative examples of user-generated shapes including Easy (grass, cylinder), Medium (twig, bottle), and Hard (cannon, croissant) difficulty levels. These examples demonstrate the variety of shapes achievable with the custom cross-section and morphing paradigm.

Table 7.1 summarizes the central tendency and variability of the main Experiment 1 questionnaire metrics.

The correlation between SUS and CSI scores was examined to understand the relationship between usability and creativity support. The Pearson correlation coefficient was $r = 0.307$, indicating a weak positive correlation.

This pattern, where CSI scores are relatively high (4.08/5, 5-point Likert scale, i.e., 77.1/100) while SUS scores are moderate (67.29/100), suggests that the system may have a higher barrier for operational mastery

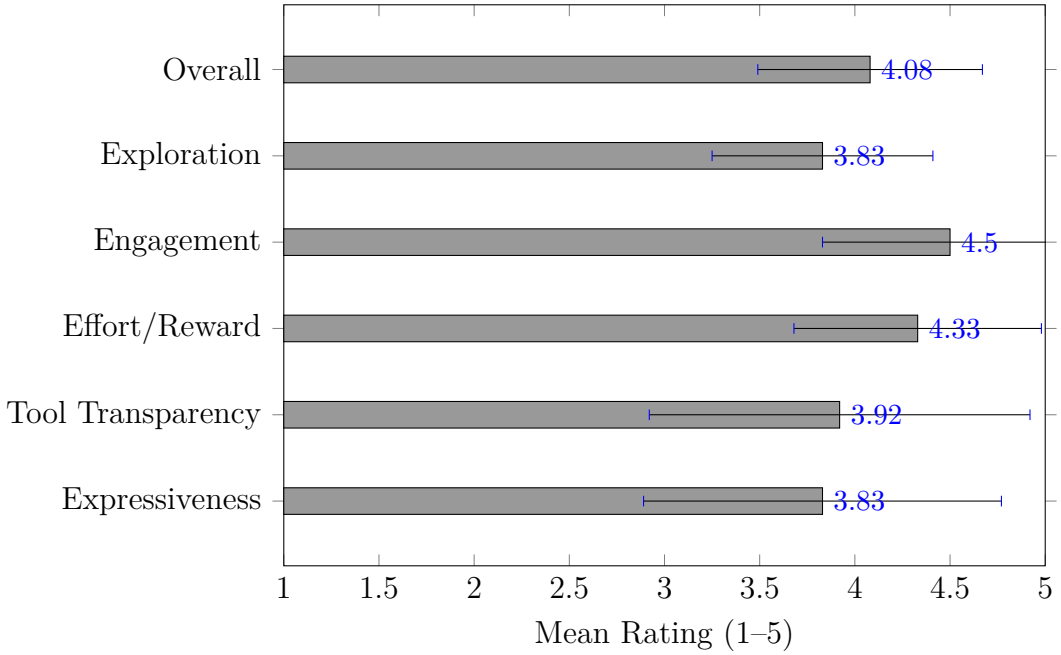


Figure 7.2: CSI per-item mean ratings (N=12) with ±1 SD error bars.

Table 7.1: Summary of Experiment 1 Usability and Creativity Metrics (N=12)

Metric	**Mean**	**SD**	**Median**	**Range**	**95% CI**
SUS Overall (1–5)	3.69	0.34	3.75	[3.10, 4.30]	[3.48, 3.91]
CSI (1–5)	4.08	0.59	4.30	[2.80, 4.80]	[3.71, 4.46]

Note: All metrics use 5-point Likert scale. SUS 3.69/5 = 67.29/100; CSI 4.08/5 = 77.1/100.

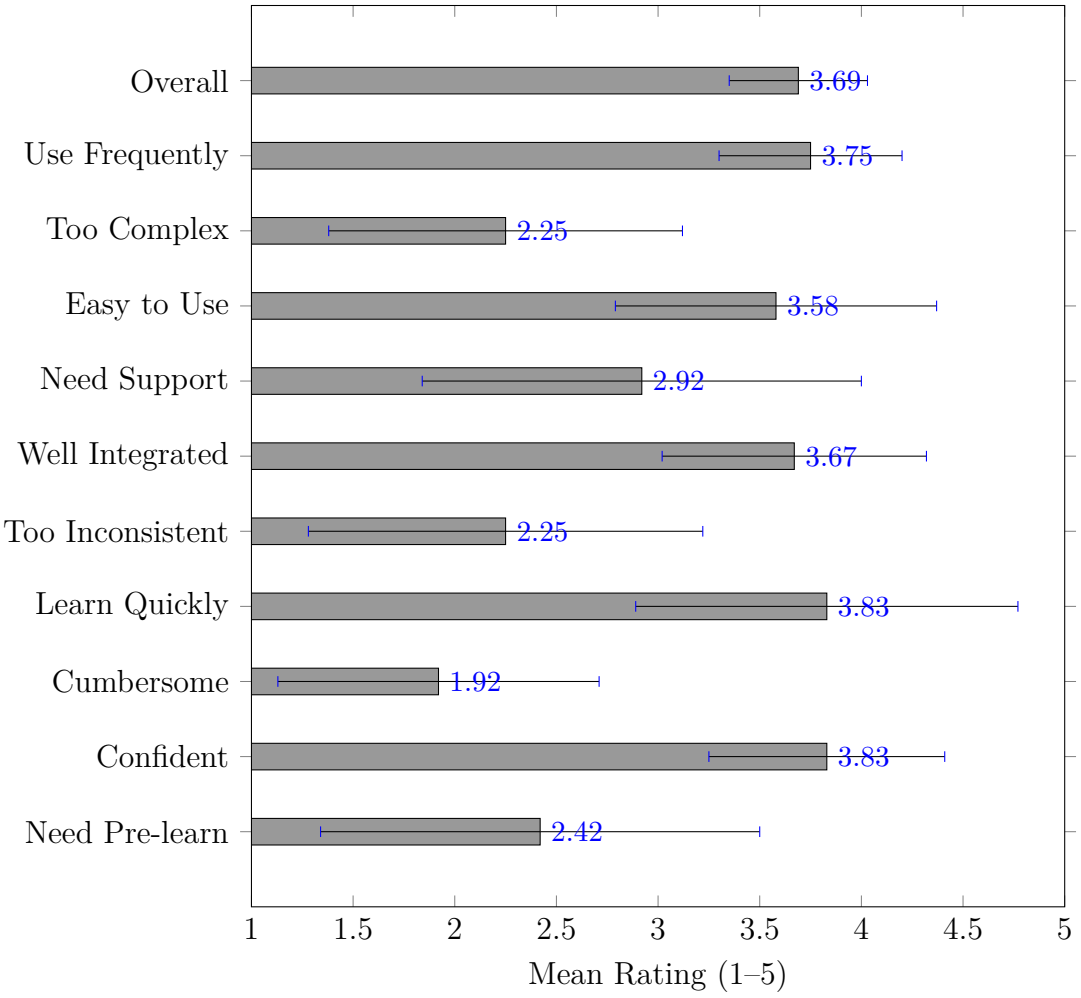


Figure 7.3: SUS per-item mean ratings (N=12) on a 1–5 scale. Odd-numbered items are positive; even-numbered items are negative (shown as raw scores). The overall SUS score is 67.29/100.

but provides strong creative capabilities once users become familiar with the interface. This is consistent with user feedback indicating that the system offers powerful creative tools (high exploration, expressiveness) while requiring some learning effort for the interaction methods.

Table 7.2 reports per-participant preference ratings for Experiment 2.

Table 7.2: Preference ratings, Experiment 2, four participants (scale 1–5).

System	**P1**	**P2**	**P3**	**P4**	**Mean**
Proposed	5	4	4	4	4.25
CavePainting	2	1	3	2	2.00
GravitySketch	2	2	3	2	2.25

7.4.2 Feedback Results

Participants in both experiments reported similar qualitative observations. Below is a concise summary of the verbatim feedback (see Experiment-Data/Data Collected/DataCollected.txt); only textual feedback is presented here.

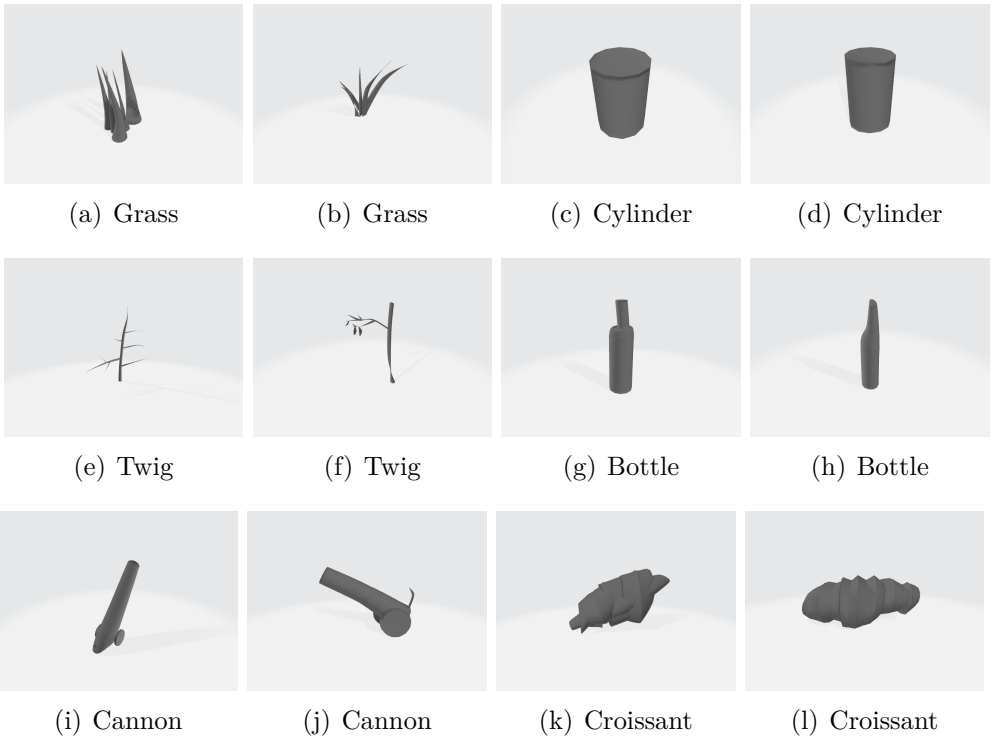
- **Learning curve and efficiency:** Many participants reported an initial learning cost but noted substantial efficiency gains after practice; several participants also described the modeling logic as clear and easy to pick up.
- **Control and precision:** Hand tremors and difficulty controlling precise angles in 3D sometimes affected modeling stability. Participants suggested precision aids such as horizontal/vertical snapping, grid guides, distance indicators, and automatic alignment of segments.
- **Interaction suggestions:** Requests included improving undo/redo flow, refining rotation controls, exposing quick-operations (select/cut/paste), and mapping the left thumbstick to brush-size adjustment during drawing. Allowing brush saving and redefinition without restarting the system was also recommended to speed creative iteration.
- **Modeling-paradigm preference:** Several participants emphasized that 3D modeling requires volume rather than only surface; when users can choose or customize appropriate cross-sections, the system better supports creative expression and produces more predictable results.

Compared to simple ribbon-style brushes, a volume/cross-section approach offers advantages in detail and controllability.

This subsection presents only participants' textual feedback to guide interaction and precision improvements.

7.4.3 User-Drawn Results

Representative examples from Experiment 1 and the Experiment 2 comparative images are placed below (figures moved to ensure they appear before the following section).



(a) Grass (b) Grass (c) Cylinder (d) Cylinder

(e) Twig (f) Twig (g) Bottle (h) Bottle

(i) Cannon (j) Cannon (k) Croissant (l) Croissant

Figure 7.4: Representative user-generated shapes from Experiment 1 (N=12), organized by difficulty (Rows: Easy, Medium, Hard).

(a) S1-AD (b) S1-CP (c) S1-GS

(d) S2-AD (e) S2-CP (f) S2-GS

(g) S3-AD (h) S3-CP (i) S3-GS

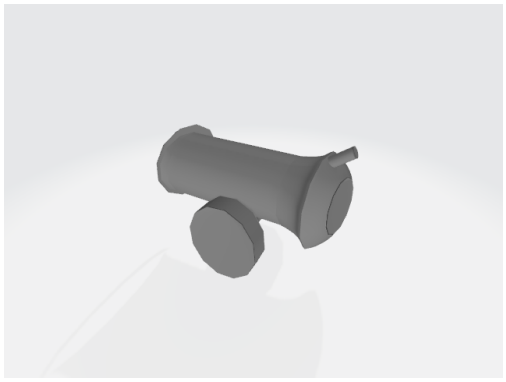
(j) S4-AD (k) S4-CP (l) S4-GS

Figure 7.5: Visual comparison of results from Experiment 2 across four participants (Rows) and three systems (Columns: The Proposed System, CavePainting, GravitySketch).

The first figure above (Figure 7.4) shows representative user-created shapes from Experiment 1, followed by a visual comparison of outcomes across systems for Experiment 2 (Figure 7.5). These figures illustrate both the variety of forms produced with the proposed system and differences observed in the comparative study.

7.5 Other Results

Due to the inherent complexity of the proposed system and the relatively short duration of the user study, participants had limited time to fully adapt to the interaction paradigm and explore the system's more advanced creative features. Consequently, it was difficult for them to produce highly complex or specialized works within the constraints of the study. This section presents a gallery of works created by the author to demonstrate the application potential of the system in producing complex, organic, and artistic 3D forms, as shown in Figure 7.6.



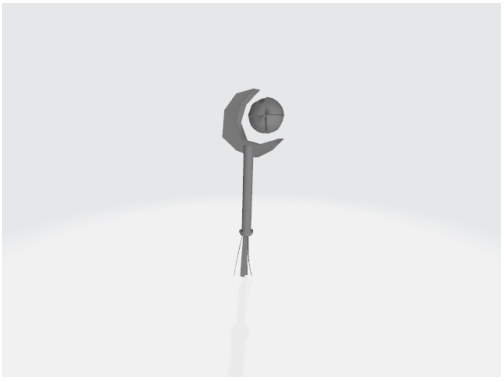
(a) Cannon

(b) Tesla Coil

(c) Double Tube Lamp

(d) Branch

(e) Coconut Trees



(f) Staff

Figure 7.6: Representative works created by the author using the proposed system (non-experimental examples).

Chapter 8

Conclusions

This thesis presented a VR spatial drawing system that applies the generalized cylinder modeling paradigm in virtual reality environments. By extending the traditional ribbon-brush approach with user-definable cross-sections and real-time morphing capabilities, the proposed system significantly expands the geometric expressiveness available to VR spatial drawing users. The evaluation results demonstrate that the proposed approach provides considerable creativity support (CSI: 77.10/100) with acceptable system usability (SUS: 67.29/100), and users consistently preferred the proposed system over baseline ribbon brushes in comparative studies (100% preference rate). These findings validate the effectiveness of combining the generalized cylinder paradigm with VR's immersive 6DOF interaction, offering a new direction for VR-based creative modeling tools.

The key insight from this research is that volumetric control through custom cross-sections addresses a fundamental limitation of existing ribbon-brush approaches. As participant feedback consistently indicated, 3D modeling fundamentally requires volume rather than just surfaces, and the proposed system's ability to define arbitrary cross-section shapes enables significantly greater creative expression and more predictable modeling results. The separation of trajectory definition from orientation control further addresses the challenges of simultaneous control that make continuous drawing approaches difficult to master, allowing users to focus on each aspect independently.

This chapter further discusses the limitations of the current system and outlines directions for future research.

8.1 Summary of Contributions

The primary contributions of this work are summarized as follows:

1. **Development of a Unified VR-based Generalized Cylinder Framework:** This research establishes a comprehensive framework for 3D spatial modeling that leverages the immersion and 6DOF interaction of VR to execute generalized cylinder modeling. By formalizing the relationship between spatial trajectories (represented by splines) and arbitrary cross-sections (represented by graph structures), a systematic method for generating volumetric meshes in real-time is provided. This framework bridges the gap between traditional sweep-based CAD tools, which often suffer from indirect 2D-to-3D mapping, and freehand VR drawing tools, which typically lack volumetric precision. This approach demonstrates that complex geometric modeling can be made intuitive through immersive spatial interactions.

2. **Expansion of Geometric Expressiveness through Advanced Cross-section and Orientation Control:** A core contribution of this work is the implementation of a flexible shape definition system that supports user-definable, graph-based cross-sections. By separating the drawing phase from the orientation and deformation phases, the challenges of simultaneous trajectory and orientation control that plague continuous drawing approaches are overcome. This separation enables continuous, smooth, and precise cross-section orientation control (particularly rolling), making it possible to accurately create intentionally twisted shapes or maintain consistent surface normals. Furthermore, the integration of real-time morphing via linear blend shapes allows for the creation of evolving geometries that transition smoothly between different profiles along a single stroke. This capability significantly expands the design space for VR spatial drawing, allowing users to move beyond simple lines or uniform ribbon-alike shapes to create diverse, complex, and organic 3D forms.

3. **Empirical Validation of Creativity Support and User Preference:** A rigorous evaluation of the proposed paradigm through two distinct user studies is provided. The quantitative results from Experiment 1 (N=12) confirmed that the proposed system provides strong support for creative work, as evidenced by a high Creativity Support Index (CSI) score of 77.10/100. Qualitative feedback and comparative results from Experiment 2 (N=4) further validated that the generalized cylinder approach is significantly more effective than traditional

brushes drawing ribbon-alike shapes. Users showed a 100% preference for the proposed system, highlighting its superiority in volumetric modeling, fine-grained control over surface orientation, and the ability to express detailed design intentions through custom shapes. These findings suggest that the techniques presented in this thesis provide a robust foundation for next-generation immersive creative tools.

8.2 Limitations

Based on participants' oral feedback during the drawing process, questionnaire responses, and direct observations of user operations, several limitations in the proposed system and the experimental process are identified. These limitations are discussed from three perspectives: interaction design, implementation, and experimental design.

8.2.1 Interaction Design Limitations

This section describes issues in the user interface and interaction logic that may hinder user experience or creative expression.

Indirect Parameter Control. Some participants noted that using sliders on the spatial Control Panel to adjust brush size is not sufficiently direct, leading to high control overhead. Users suggested that more immediate input methods, such as using controller thumbsticks to adjust size and morph parameters during the drawing process, would be more intuitive.

Alignment and Perception Support. Participants expressed a need for alignment and snapping features. For instance, creating short but wide cylinders (e.g., wheels) is difficult without cross-section alignment tools. Furthermore, users often need to observe the placement from multiple viewpoints to accurately perceive 3D spatial relationships, such as ensuring a curve is perfectly vertical to the ground. Users also requested functionality to close trajectories by snapping the head and tail of a curve together, which is currently missing.

Structure Reuse. Users suggested adding selection and duplication capabilities. Such features would allow users to reuse complex structures (e.g., leaves) across different parts of a model, significantly improving creative efficiency.

Undo Granularity and Post-Editing. Observation revealed frustration when a single mistake mid-stroke forced users to undo the entire stroke. Implementing per-point undo (removing the most recent control point) or

introducing post-draw editing capabilities would resolve this issue and is technically feasible within the proposed framework.

Shape Editor Complexity. Relatively few users utilized the custom base shape editor, suggesting the interface may be too complex for casual users. Lowering the entry barrier by providing more diverse shape presets and intuitive 2D/3D previews during selection is necessary.

8.2.2 Implementation Limitations

This section discusses technical shortcomings and software engineering oversights that impacted the user experience.

Dragging Direction Ambiguity. A specific interaction bug occurs when a user confirms a tangent direction (by dragging) that is nearly opposite to the direction of the previous point. This results in a pinched, funnel-like mesh structure (see Figure 8.1). This phenomenon caused confusion among some users, requiring experimenters to emphasize that dragging must always follow the intended path of the stroke.

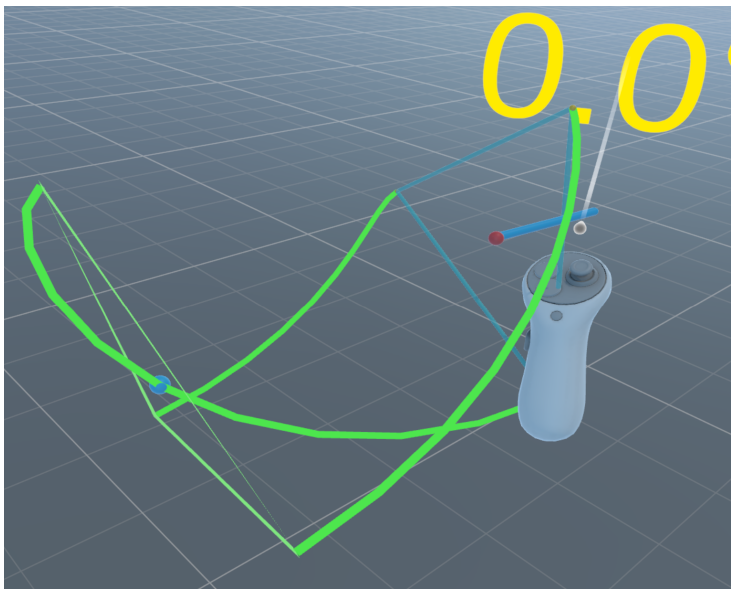


Figure 8.1: Ambiguity in dragging direction. When the drag direction is opposite to the previous point's tangent, the cross-section orientation becomes uncontrolled, resulting in a funnel-like distortion.

Frame Rate Performance. When a single stroke contains a large number of control points, the real-time mesh regeneration overhead causes noticeable frame rate drops. This performance bottleneck interrupts the creative

flow and impacts VR comfort.

Shape Management. The current system architecture prevents users from switching shape libraries during a session. Users must restart the application to load different sets of base shapes, which hinders the creation of complex scenes requiring multiple distinct brush types.

8.2.3 Experimental Design Limitations

The evaluation process also encountered several challenges.

Small Comparative Sample Size. While Experiment 2 showed a strong preference for the proposed system, the sample size (N=4) is relatively small, which may limit the generalizability of the comparative results.

Paradigm Differences. The proposed drawing paradigm (discrete control point placement) is fundamentally different from the continuous trigger-to-draw methods used in systems like AdaptiBrush. This difference in interaction models makes it challenging to conduct controlled feature-by-feature comparisons with existing tools.

Ablation Study Failure. Our initial attempt at an ablation study (varying "Custom Shape vs. Ribbon-alike" and "Rolling Control vs. No Rolling") was cancelled due to unexpected interaction artifacts. We found that a "Ribbon-alike" (1D segment) cross-section behaves differently under constraints compared to 2D planar shapes. Specifically, since a line segment is 1D, dragging it in 3D space introduces orientation ambiguities (see Figure 8.2). This results in unintended "twisting" even when rolling control is theoretically disabled. This made it impossible to maintain a valid control group when comparing 1D segments with 2D shapes, as users would accidentally create twisted ribbons through lateral hand movements.

8.3 Future Work

Building upon the identified limitations, future research will focus on the following areas:

8.3.1 Interaction Design Enhancements

The primary focus of future updates will be on more direct and assistive interaction methods. This includes:

- **Direct Input Mapping:** Mapping brush size and morph parameters to controller thumbsticks to allow for real-time adjustment without interacting with the spatial UI panel.

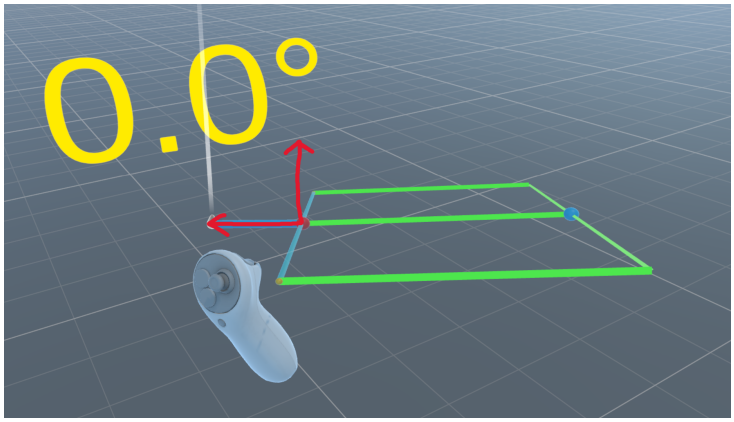


Figure 8.2: Orientation ambiguity for line segments (ribbons). Because the system treats the segment as a graph-based planar shape, dragging it in different directions (which are visually equivalent for a line) leads to different underlying orientation storage, causing uncontrollable twisting effects. In this example, leftward and upward drags produce identical 3D meshes, but the internal handling differs; this difference can manifest in subsequent strokes.

- **Alignment and Snapping Tools:** Implementing grid snapping, axis alignment, and point-to-point snapping (especially for closing trajectories) to support precise architectural and mechanical modeling.
- **Hierarchical Editing:** Introducing selection, grouping, and duplication tools to enable modular modeling workflows.
- **Refined Undo and Post-Editing:** Developing a per-operation undo system and a dedicated edit mode where users can reposition and rotate control points after the initial stroke is drawn.

8.3.2 Implementation and System Robustness

Technical bottlenecks identified during use will also be addressed:

- **Incremental Mesh Updates:** Optimizing the mesh generation engine to only recompute affected segments, thereby maintaining high frame rates regardless of stroke complexity.
- **Runtime Asset Management:** Redesigning the shape library system to allow for dynamic loading and switching of cross-section sets without restarting the application.

- **Heuristic Tangent Correction:** Implementing logic to detect and prevent funnel artifacts by automatically correcting dragging directions that conflict with the established trajectory.

8.4 Concluding Remarks

VR spatial drawing represents a compelling fusion of natural human gesture with digital creativity. The immersive nature of VR enables direct manipulation in three-dimensional space, offering artists and designers an intuitive environment for shape creation that differs fundamentally from traditional desktop interfaces.

By extending the generalized cylinder paradigm with user-definable shapes and morphing capabilities, the proposed system provides new tools for expressing creative visions in three dimensions. The click-drag-rotate interaction paradigm offers a balance between precision and expressiveness, while the base shape and morph editors enable customization that was previously unavailable in VR spatial drawing tools.

The experimental evaluation provides evidence for the effectiveness of this approach. The high creativity support scores (CSI: 4.083/5, 5-point Likert scale) demonstrate that the system successfully enables creative expression, with users appreciating the ability to explore different design options, achieve immersive engagement, and produce results worth their effort. The unanimous user preference in the comparative study (100% preference for the proposed system over baseline ribbon brushes) validates the advantages of the generalized cylinder paradigm for VR 3D modeling, particularly the importance of volumetric shape creation and fine-grained control through custom cross-sections.

While the moderate usability scores (SUS: 67.29/100) indicate areas for improvement in terms of learning curve and ease of use, the disparity between usability and creativity support scores suggests a tool with powerful capabilities that rewards user investment in learning. This pattern is not uncommon in professional creative tools, where expressive power may come at the cost of initial complexity.

As VR hardware continues to improve and become more accessible, growing interest in VR-native creative tools is anticipated. It is hoped that this work contributes to the ongoing evolution of VR as a medium for creative expression, and that the techniques presented here inspire further research in immersive 3D modeling interfaces. The positive user feedback on volumetric modeling capabilities and control precision suggests promising directions for future VR creative tools that go beyond surface-based approaches to embrace

richer geometric representations.

Acknowledgment

I would like to express my sincere gratitude to all those who supported me throughout the completion of this thesis.

First and foremost, I am deeply grateful to my supervisor, Professor Haoran Xie, for his invaluable guidance throughout this research. He introduced me to key literature that helped me find my research direction, including the 3-Sweep paper that inspired me to view our work from the perspective of generalized cylinder modeling. When my thinking was limited or heading in the wrong direction, his insights opened new perspectives and guided me toward more productive approaches. The reading list he provided served as a foundation for finding inspiration and positioning our research. While the subsequent organization, topic refinement, and development work relied primarily on my own efforts, his careful review and feedback at each stage ensured the quality of this work.

I would also like to thank the members of our research laboratory for their constructive feedback during our regular seminars and for creating a stimulating academic environment. The discussions and collaborations with my fellow students have enriched my understanding and broadened my perspectives.

Special thanks go to all the participants in our user studies, whose time and feedback were essential for evaluating the system. Their insights helped identify both the strengths and areas for improvement in ADBrush.

I am grateful to Japan Advanced Institute of Science and Technology (JAIST) for providing the facilities and resources necessary for this research, including access to VR equipment and computing infrastructure.

Finally, I would like to thank my family and friends for their unwavering support and understanding throughout my graduate studies. Their encouragement has been a constant source of motivation.

Yude Lu
March 2026

References

[1] E. Rosales, C. Araújo, J. Rodríguez, N. Vining, D. Yoon, and A. Sheffer, "Adaptibrush: adaptive general and predictable vr ribbon brush," *ACM Transactions on Graphics*, vol. 40, pp. 1–15, 12 2021.

[2] VRChat Inc., "VRChat," https://hello.vrchat.com/, accessed: January 2026.

[3] E. Yu, R. Arora, T. Stanko, J. A. Bærentzen, K. Singh, and A. Bousseau, "Cassie: Curve and surface sketching in immersive environments," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3411764.3445158

[4] T. Chen, Z. Zhu, A. Shamir, S.-M. Hu, and D. Cohen-Or, "3-sweep: extracting editable objects from a single photo," *ACM Trans. Graph.*, vol. 32, no. 6, Nov. 2013. [Online]. Available: https://doi.org/10.1145/2508363.2508378

[5] S. Chatterjee, "Free-form shape modeling in xr: A systematic review," 2024. [Online]. Available: https://arxiv.org/abs/2401.00924

[6] D. F. Keefe, R. C. Zeleznik, and D. H. Laidlaw, "Tech-note: Dynamic dragging for input of 3d trajectories," in *2008 IEEE Symposium on 3D User Interfaces*, 2008, pp. 51–54.

[7] E. Rosales, J. Rodriguez, and A. Sheffer, "Surfacebrush: From virtual reality drawings to manifold surfaces," *ACM Transaction on Graphics*, vol. 38, no. 4, 2019.

[8] Google, "Tilt Brush," https://https://github.com/googlevr/tilt-brush, accessed: January 2026.

[9] Gravity Sketch, "Gravity Sketch," https://gravitysketch.com/, accessed: January 2026.

[10] Blender Foundation, "Blender," https://www.blender.org/, accessed: January 2026.

[11] SuperHive Market, "Curves-to-Mesh (blender add-on)," https://superhivemarket.com/products/curves-to-mesh, accessed: January 2026.

[12] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Fibermesh: designing freeform surfaces with 3d curves," *ACM Trans. Graph.*, vol. 26, no. 3, p. 41–es, Jul. 2007. [Online]. Available: https://doi.org/10.1145/1276377.1276429

[13] H. Schulz, M. Teschner, and M. Wimmer, "Rodmesh: Two-handed 3d surface modeling in virtual reality," in *Vision, modeling and visualization*, 2019, pp. 1–10.

[14] D. Keefe, D. Acevedo, T. Moscovich, D. Laidlaw, and J. LaViola, "Cavepainting: A fully immersive 3d artistic medium and interactive experience," *Proceedings of the Symposium on Interactive 3D Graphics*, pp. 85–93, 01 2001.

[15] M. Kusunoki, R. Furuhama, R. Toshima, H. Mori, H. Xie, T.-Y. Wang, T. Yuizono, T. Sato, and K. Miyata, "Multibrush: 3d brush painting using multiple viewpoints in virtual reality," in *2023 9th International Conference on Virtual Reality (ICVR)*, 2023, pp. 481–486.

[16] T. O. Binford, "Visual perception by computer," in *Proceedings of the IEEE Conference on Systems and Control (Miami, FL)*, 1971.

[17] A. Nealen, O. Sorkine-Hornung, M. Alexa, and D. Cohen-Or, "A sketch-based interface for detail-preserving mesh editing," *ACM SIGGRAPH 2005 Papers*, 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:47445493

[18] B. Jackson and D. F. Keefe, "Lift-off: Using reference imagery and freehand sketching to create 3d models in vr," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 4, pp. 1442–1451, 2016.

[19] E. E. Catmull and R. Rom, "A class of local interpolating splines," *Computer Aided Geometric Design*, pp. 317–326, 1974. [Online]. Available: https://api.semanticscholar.org/CorpusID:118383557

[20] V. Blanz and T. Vetter, “A morphable model for the synthesis of 3d faces,” in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’99. USA: ACM Press/Addison-Wesley Publishing Co., 1999, p. 187–194. [Online]. Available: https://doi.org/10.1145/311535.311556

[21] Unity Technologies, “Unity,” https://unity.com/, accessed: January 2026.

[22] Khronos Group, “OpenXR specification,” https://www.khronos.org/openxr/, accessed: January 2026.

Appendix A

Experimental Materials

A.1 Experiment 1: Participant Instructions

The following text was shown/read to participants during Experiment 1:

> You need to use the full mode of this system to draw the following subjects.
>
> Each difficulty level only needs to be drawn once for one of the items.
>
> There is no forced time limit for the following processes. First, the experimenter will demonstrate how to use the system. You can observe the demonstrator's actions in the Unity debug interface. After completion, you will experience the system under the experimenter's guidance. Finally, you will perform the drawing tasks, but you can ask the experimenter questions at any time, and the experimenter will tell you how to operate.
>
> The drawing tasks will be divided into three groups based on difficulty. You need to use the full mode of this system to draw a self-chosen subject. Please follow the experimenter's instructions on how to start.
>
> Easy: Grass / A Cylinder
>
> Medium: Twig / Beer Bottle
>
> Hard: Cannon / Croissant
>
> The shapes you draw will be recorded.
>
> After completing the drawing of a figure, please save the shape.
>
> After completing all tasks, please provide evaluation.

A.2 Experiment 2: Participant Instructions

The following text was shown/read to participants during Experiment 2:

> You need to use three different tools: GravitySketch [9], CavePainting [14], and our system (ADBrush) to perform the experiments.
>
> You need to use these systems to draw the following subjects. Under the given conditions, you can perform arbitrarily.
>
> In the figures not drawn below, you can arbitrarily determine a subject, and then use different systems to draw. If you have any subject you want to draw, you can also propose it. After drawing, try to use different systems to draw the same subject. You can also try other subjects.
>
> Subjects: Grass, Table Lamp, Twig, Bottle, Cannon, Knife.
>
> After completion, please fill out a questionnaire.

A.3 Questionnaires

This section provides the complete questionnaire instruments used in our evaluation studies.

A.3.1 Questionnaire 1: Creativity Support Index (CSI)

The Creativity Support Index is a standardized instrument for evaluating how well a system supports creative work. We used a modified version with 5 items (1 unrelated item deleted from the original 6-item scale).

Participants rated each item from 1 (strongly disagree) to 5 (strongly agree).

A.3.2 Questionnaire 2: System Usability Scale (SUS)

The System Usability Scale is a widely-used 10-item questionnaire for assessing subjective usability of systems.

Participants rated each item from 1 (strongly disagree) to 5 (strongly agree). The final SUS score is calculated by converting the raw scores to a 0–100 scale using the standard SUS scoring formula.

Table A.1: CSI Questionnaire Items

No.	Question Description
Q1-1	**Exploration**: I was able to explore many different options, ideas, designs, or outcomes easily without a lot of tedious, repetitive work.
Q1-2	**Immersion**: I was very absorbed/engaged in the activity; I enjoyed the process and would do something similar again.
Q1-3	**Quality**: What I was able to produce was worth the effort I had to exert to produce it.
Q1-4	**Intuition**: The tool/interface/system "disappeared" while working and allowed me to concentrate on the work itself.
Q1-5	**Playfulness**: I was able to be very expressive and creative while doing the activity.

Table A.2: SUS Questionnaire Items

No.	Question Description
Q2-1	I think that I would like to use this system frequently.
Q2-2	I found the system unnecessarily complex.
Q2-3	I thought the system was easy to use.
Q2-4	I think that I would need the support of a technical person to be able to use this system.
Q2-5	I found the various functions in this system were well integrated.
Q2-6	I thought there was too much inconsistency in this system.
Q2-7	I would imagine that most people would learn to use this system very quickly.
Q2-8	I found the system very cumbersome to use.
Q2-9	I felt very confident using the system.
Q2-10	I needed to learn a lot of things before I could get going with this system.

A.3.3 Questionnaire 3: Preference Ratings (Experiment 2)

This questionnaire was used in Experiment 2 after participants had tried all three VR drawing systems.

Table A.3: Preference Questionnaire Items

No.	**Question Description**
Q3-1	Overall preference rating for **Our System**.
Q3-2	Overall preference rating for **CavePainting** (reproduced ribbon brush).
Q3-3	Overall preference rating for **GravitySketch** (reproduced ribbon brush).

Participants rated each system from 1 to 5 and reported which system they preferred most, with a brief explanation of their reasoning.