Interactive swept surface modeling in virtual reality with motion-tracked controllers

Tim McGraw
Purdue University
401 N Grant Street
West Lafayette, Indiana 47907
tmcgraw@purdue.edu

Esteban Garcia
Purdue University
401 N Grant Street
West Lafayette, Indiana 47907
garcia0@purdue.edu

Drew Sumner
Purdue University
401 N Grant Street
West Lafayette, Indiana 47907
sumnera@purdue.edu

ABSTRACT

In this work we describe an interactive technique which enables gestural curve and surface design in an immersive virtual environment. We use a pair of motion tracked controllers to allow the user to intuitively control a Hermite spline curve which can be swept through space to create surfaces, or to define the location and orientation of cloned meshes. A head mounted display and tracked controllers replace the traditional keyboard and mouse for view selection and object interaction. Natural and expressive body motions allow the user to specify the parameters which define the curves and surfaces. Results are demonstrated using the HTC Vive VR system.

CCS CONCEPTS

•Computing methodologies → Virtual reality; •Human-centered computing → Gestural input;

KEYWORDS

Virtual reality, computational Aesthetics, modeling,

ACM Reference format:
Tim McGraw, Esteban Garcia, and Drew Sumner. 2017. Interactive swept surface modeling in virtual reality with motion-tracked controllers. In *Proceedings of SBIM'17, Los Angeles, CA, USA, July 28-29, 2017,* 9 pages.
DOI: http://dx.doi.org/10.1145/3092907.3092908

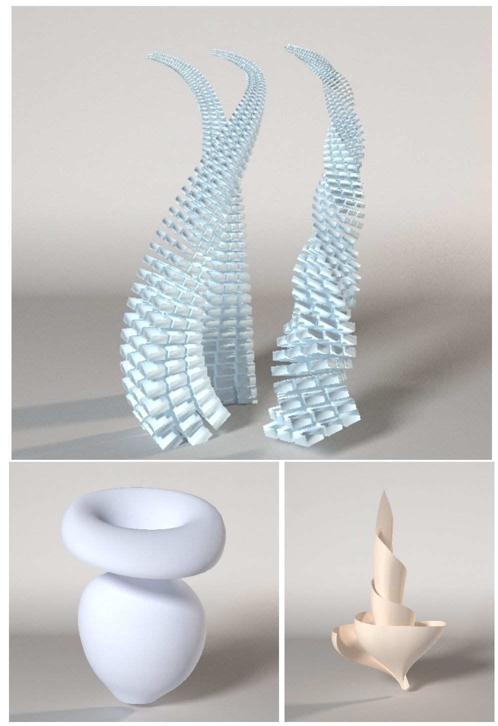

Figure 1: Rendered images of example models which can easily be generated with our technique. Open surfaces, tubes and cloned meshes

1 INTRODUCTION

Low-cost commercial VR systems with high-resolution motion-tracked controllers enable the development of 3D design tools. But defining how the user's gestures should be mapped to object creation is an open problem. It is often difficult for users to achieve their desired results by simply freehand sketching in the air, especially when trying to precisely draw smooth curves [Arora et al. 2017]. Many previous systems have been limited to creating collections of loose brush strokes [Keefe et al. 2001; Machkovech 2016]. For 3D surface modeling a new set of gestures and metaphors are required to realize the potential of VR in modeling.

In this paper we will describe the implementation of an immersive modeling tool which enables artists and designers to create sculptural forms by interacting with virtual objects and tools. In our system users' hand motions tools generate surfaces which can be exported as 3D digital models. We present results as rendered images and fabricated models. Potential applications include: modeling for virtual environments, design of assets for games and animation, and the creation of sculptural objects.

Creating meshes vertex-by-vertex would be a laborious task considering that meshes may consist of millions of vertices. In this work we demonstrate the use of the Hermite spline as a modeling and control primitive. It offers intuitive control, guaranteed geometric smoothness and the ability to be easily extended to higher-order open and closed surfaces. It can also serve as a canvas for cloning collections of other meshes. Example surfaces modeled with our technique are shown in Figure (1).

SBIM'17, Los Angeles, CA, USA

DOI: http://dx.doi.org/10.1145/3092907.3092908

For simple sketching in the air it is only necessary to track a single 3D point through space. Even though controller orientation can be easily be retrieved it is often not used in 3D sketching applications. Using a pair of controllers in tandem for 3D design means that a pair of 3D points and orientation matrices can be used as control parameters. Our technique uses all of these pieces of data to specify the endpoints and tangents of the Hermite curve, as well as additional controller inputs to define a reparameterization function which gives the user precise and intuitive control over the position and shape of the curve. A rotation minimizing frame is defined along the Hermite curve for use in distributing cloned meshes. The frame and reparameterization function allow the user to control cloned mesh locations, orientations and scales.

The techniques we propose in this work are not a solution for all modeling needs. Just as traditional desktop artists use a variety of techniques (e.g. NURBS, subdivision surfaces, virtual sculpting) we feel our technique could be one useful component within a larger toolbox of immersive modeling techniques.

The main contributions of our work are

- An expressive interaction technique which enables the user to define a 3D space curve for modeling applications
- Methods for converting the moving curve to higher-level modeling primitives, such as surfaces, tubes and replicated meshes

In the remainder of this paper we will review the related literature, describe the implementation details of our approach, present the results and discuss conclusions and future work.

2 RELATED WORK

Our technique is closely related to prior work in content creation in virtual reality and interaction design, specifically work involving virtual painting and sculpting. Creating shapes with gestural motion is analogous to sculpting with real materials like clay or plaster. Similar to the way that the potter on the wheel shapes vases by applying pressure to the surface, users of our application can sculpt shapes by moving their hands. Users can control the extrusion of material by pulling the trigger on the controller and moving their hands and arms. They can apply pressure by moving the controllers. The tilt of the controller affects the orientation of the surface. In this section, we will review some of the relevant approaches of gestural and virtual modeling.

2.1 Virtual reality

VR is an appealing platform for artistic applications because it can provide the user with convincing senses of depth, parallax and other spatial cues [Bryson 1996]. The diverse applications of VR technology include entertainment, scientific visualization [Bryson and Levit 1991], archeology and preservation of cultural heritage [Guidi et al. 2014].

Stereoscopic head-mounted displays (HMDs) have been used as display devices for virtual reality since the 1960s [Sutherland 1968], but factors such as high weight, low field-of-view (FOV), and slow tracking limited their commercial viability for many years [Earnshaw 2014]. Other display technologies, such as the Cave automatic virtual environment (CAVE) [Cruz-Neira et al. 1993] based on multiple projectors, and desktop fishtank VR [Ware et al. 1993] systems using multiple monitors have also been developed to give users an immersive 3D view of a virtual environment. In the modern VR era, low-cost lightweight consumer devices featuring high-FOV HMDs are becoming common.

Many of these systems, such as the HTC Vive and the Oculus Rift, are capable of low-latency, high-precision tracking of the both the HMD and handheld controllers which permit intuitive interaction with virtual environments. Other standalone technologies, such as the MagicLeap system for hand tracking can easily be incorporated into a VR system which doesn't feature integrated motion-tracked controllers. Early user studies on interaction in CAVE and fishtank VR systems [Demiralp et al. 2006] found that handheld tracked controls led to better performance on visual search tasks than keyboard and mouse. Theart at al. [Theart et al. 2017] found that handtracking outperformed gamepad interaction on interaction tasks in the context of microscopy image exploration. In this work we explore the use of a pair of controllers in tandem to facilitate intuitive curve and surface design.

The content of virtual environments is usually created on a PC using mouse and keyboard and standard desktop modeling tools, or in the case of *VRgram* [Wong and Wong 2016], on and a phone or tablet.

Sketching has been shown to be a useful interface to curve [McCrae and Singh 2009] and surface [Bae et al. 2008] design, but recent research [Arora et al. 2017] found that freehand sketching can be difficult in VR due to the natural tendency of the arm to move along arcs defined by joint locations, and the absence of supporting surfaces for the user's hand or arm. We hypothesize that by using both controllers to specify the parameters of a 3D curve we can overcome the difficulties posed by freehand sketching in the air.

2.2 Immersive drawing and painting

Tilt Brush [Machkovech 2016] is a recent VR drawing application which allows the user to create freehand painted curves. The user's strokes are extruded into ribbons which can be textured with a variety of patterns and colors, but no parametric curves, surfaces or solids can be drawn.

Drawing on Air [Keefe et al. 2007] featured a two-hand "tape drawing mode" which allowed a user to simultaneously specify a position and local tangent direction for a stroke. However, they found that the technique required force-feedback from an integrated Sensable Phantom device to provide the user with proper resistance to make the technique usable.

The *CavePainting* system [Keefe et al. 2001] was implemented with a CAVE and used tracked physical objects such as paint brushes and buckets to perform painting. *Surface Drawing* [Schkolne et al. 2001] is a 3D immersive modeling tool based on single-handed freehand strokes and other tools to manipulate the sketched surfaces, such as scaling, moving and erasing. The COVIRDS system [Dani and Gadh 1997] used both speech and gestures to interact with a CAD modeling system.

Henrikson et al. [Henrikson et al. 2016] developed a tool for collaborative development of cinematic narratives for VR. Their tool permits a storyboard artist (sketching on a tablet) and a director (wearing a HMD) to simultaneously develop layered storyboards.

2.3 Gestures and motion

Motion tracking technologies have allowed the exploration the relationships between virtual creations, the artists creating them and the 3D space which they inhabit.

Bodygraphe [Garcia and McGraw 2016] was a system for generating dynamic 3D surfaces from motion for the purposes of dance and music visualization. It acquired a stream of human skeleton data from a Kinect sensor and generated smooth swept surfaces in response to the dancer's motion.

Spatial Correlation [Nam and Keefe 2016] captured the process of sculpting with gestures from multiple cameras. In the gallery a viewer could see the performance and the sculpture from multiple view points.

In *Osmose* [Davies and Harrison 1996], tilt sensors and a custom-built breathing vest were used for navigation through a virtual environment. By leaning their body, the participant could move in the 2D plane. Expansion and contraction of the chest during inhalation and exhalation allowed the user to rise and fall in the vertical direction.

2.4 Virtual sculpting

Virtual sculpting [Sclaroff et al. 1990] is not always done in an immersive virtual environment. But it has long been a goal of many modeling tool developers. Many take the approach of trying to create an accurate physical simulation of the clay sculpting process, relying heavily on accurate collision detection and numerical simulation of material deformation. Some proposed methods use haptic feedback [Dachille IX et al. 1999; Gunn 2006] to provide the user with a sense of the forces involved. Imposing geometric constraints, such as volume preservation and avoiding self-intersections [Angelidis et al. 2006] can also give a realistic feel to the sculpting process.

The first example of virtual sculpting dates back to 1968 when Robert Mallary developed TRAN2, a computer graphics program to make sculpture [Mallary 1970]. Similar to our application, TRAN2 generated shapes by using ellipsoids as a design primitives for extrusion. However, TRAN2 was entirely numerical and did not have a visual interface. The input drawn contours were converted into code and then plotted on paper. Then each slice design was transferred to laminated wood or metal and stacked to create a physical 3D object. Today, programs like OpenSCAD use code as an interface to 3D modeling, an allow 2D geometric outlines to be extruded [Kintel and Wolf 2011].

The workflow in *Lift-off* [Jackson and Keefe 2016] starts with 2D pencil sketches. An artist in the CAVE then lifts the sketched lines into 3D, where the curves are interpreted as boundaries of surfaces.

Several commercial modeling tools, such as Zbrush [Spencer 2011], present the user with interaction metaphors like sculpting and chiseling wood, stone and other materials.

3 METHODS

In this section we describe our parametric curve control system, and demonstrate how it is applied to the modeling process.

3.1 Parametric curve control in virtual reality

Recent consumer VR systems, such as the HTC Vive, consist of a head-mounted display (HMD) and motion-tracked controllers which provide a natural way of interacting with virtual environments. Using a standard mouse and keyboard with a HMD is inconvenient, but hand-held controllers permit the developer of a VR application to respond to user gestures, touches and button presses. Though primarily developed with entertainment applications in mind, many companies and the academic research community have recognized the potential for VR to display situationally relevant information to a user and provide cognitive support for demanding tasks. Commercial VR systems and freely-available programming interfaces, such as OpenVR [Corporation 2017], permit researchers create VR systems without the need to develop their own hardware or low-level software.

We aim to simplify the process of modeling by exploiting the user's sense of proprioception. The body-centered interaction design principle [Slater and Usoh 1994] can lead to more intuitive interaction by tightly coupling the human proprioceptive sense and the visual representation of the virtual environment. By attaching objects to a motion-tracked controller the user can easily bring the object into view when needed by simply moving their hand into their field-of-view while holding the controller, and just as easily hide it again. No knowledge of the buttons on the controller is required. Object zoom and rotation can be realized by simply moving the tracked controller. This on-demand display of objects reduces user dependence on the user interface and results in less visual clutter on the screen.

In this section we describe how we use a parametric curve which is controlled by the two tracked controllers to create 3D surfaces. Specifically, we formulate the surface in terms of a cubic Hermite spline. A common complaint about using Hermite splines for general curve and surface design is that tangent vectors must be specified at the endpoints, which can be less intuitive than simply specifying additional control points. However, in our interactive application we feel it is natural to map the two spline control points to controller position and map tangent vectors to controller orientations. Providing immediate visual feedback to the user makes it easy to learn the space of shapes which can be specified. The Hermite spline is not as flexible as the nonuniform rational B-splines (NURBS) which are popular in modeling packages such as Maya [Palamar 2013]. From the Hermite spline we further define modeling approaches based on swept surfaces, tubes and replicated meshes.

3.1.1 *Tracked controllers.*

As the user moves the controllers through the virtual environment the tracked controllers provide new transformation matrices M_{left} and M_{right} representing position and orientation. Letting $\mathbf{z} = [0, 0, 0, 1]^T$ and $\mathbf{r} = [1, 0, 0, 0]^T$, we define the endpoints and tangents of the Hermite curve in world coordinates as

$$\begin{aligned} \mathbf{p}_0 &= M_{\text{left}}\mathbf{z}, \\ \mathbf{p}_1 &= M_{\text{right}}\mathbf{z}, \\ \mathbf{m}_0 &= M_{\text{left}}\mathbf{r}, \\ \mathbf{m}_1 &= M_{\text{right}}\mathbf{r}. \end{aligned} \tag{1}$$

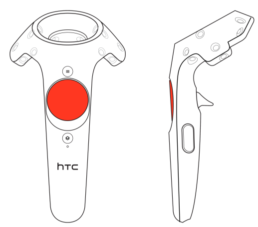

Figure 2: The Vive controller trackpad.

The Vive controller trackpad, shown in Figure (2), is a high-resolution touch-sensitive surface that supports touch gestures and clicking interactions. A trackpad is positioned on each controller where the user's thumb can easily access it. The controller also features an analog trigger and other buttons which can be programmed for other functions. Later in this section we describe how we allow the user to reparameterize the Hermite curve using the trackpad.

3.1.2 Hermite spline. The cubic Hermite curve [Farin 2014], $\mathbf{s}(u)$, has endpoints $\mathbf{p}_0$ and $\mathbf{p}_1$. At the endpoints the curve is tangent to the given vectors $\mathbf{m}_0$ and $\mathbf{m}_1$. The parametric equation of the curve is given by

$$\mathbf{s}(u) = \mathbf{p}_0 H_0(u) + \mathbf{m}_0 H_1(u) + \mathbf{m}_1 H_2(u) + \mathbf{p}_1 H_3(u) \quad (2)$$

with the parameter $u \in [0, 1]$ and basis functions given by

$$\begin{aligned} H_0(u) &= 2u^3 - 3u^2 + 1, \\ H_1(u) &= u^3 - 2u^2 + u, \\ H_2(u) &= -2u^3 + 3u^2, \\ H_3(u) &= u^3 - u^2. \end{aligned} \quad (3)$$

The curve, $\mathbf{s}(u)$, will be used to specify swept surfaces and the location of cloned (or replicated) meshes. In order to uniquely define the cloned mesh orientations we will create a coordinate frame in terms of $\mathbf{s}(u)$. In the following subsection we will describe how we choose the basis vectors $\tilde{T}(u), \tilde{B}(u), \tilde{N}(u)$ which define that frame. The tangent vector to the curve determines $\tilde{T}(u)$, and we can compute it analytically by differentiating the cubic Hermite in Equation (2) to obtain

$$\mathbf{s}'(u) = \mathbf{p}_0 H_0'(u) + \mathbf{m}_0 H_1'(u) + \mathbf{m}_1 H_2'(u) + \mathbf{p}_1 H_3'(u) \quad (4)$$

where the derivatives of the basis functions in Equation (3) are given by

$$\begin{aligned} H_0'(u) &= 6u^2 - 6u, \\ H_1'(u) &= 3u^2 - 4u + 1, \\ H_2'(u) &= -6u^2 + 6u, \\ H_3'(u) &= 3u^2 - 2u. \end{aligned} \quad (5)$$

The unit tangent to the cubic Hermite curve is given by $\mathbf{T}(u) = \mathbf{s}'(u)/||\mathbf{s}'(u)||$. As we will demonstrate, simply using the Frenet frame (tangent, binormal, normal) to define rotations which align meshes with the curve provides unsatisfactory results. For this reason we use the tilde notation, $\{\tilde{T}, \tilde{B}, \tilde{N}\}$, to denote that our basis vectors may differ from the actual Frenet frame $\{T, B, N\}$.

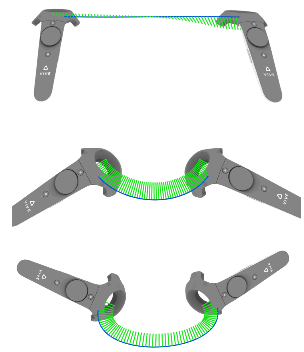

Figure 3: Hermite spline (blue) and the Frenet normal vector (green). Note in the top image that the Frenet normal may rotate about the curve when it is straight.

3.1.3 Frenet frame. The Frenet frame [Farin 2014] defines an orthonormal set of basis vectors which move along the curve as the parameter, u, changes. The binormal vector, $\mathbf{B}$(u), and normal vector, $\mathbf{N}$(u), are given by

$$\begin{aligned} \mathbf{B}(u) &= \frac{\mathbf{s}'(u) \times \mathbf{s}''(u)}{||\mathbf{s}'(u) \times \mathbf{s}''(u)||} \quad (6) \\ \mathbf{N}(u) &= \mathbf{B}(u) \times \mathbf{T}(u). \quad (7) \end{aligned}$$

However, the binormal and normal vectors can behave in unintuitive ways when the curve becomes flat and the magnitude of $\mathbf{s}''(u)$ becomes small, as shown in Figure 3. In fact, for a straight line the choice of $\mathbf{B}$ and $\mathbf{N}$ become arbitrary as long as the basis vectors are mutually orthogonal. So the Frenet frame does not provide a suitable coordinate frame for defining mesh orientations along the Hermite spline controller.

3.1.4 Quaternion Slerp frame. An alternative approach to using the Frenet frame is to interpolate the orientations of the controllers. This can be implemented by converting the controller matrices M_{left} and M_{right} to quaternions q_{left} and q_{right}, respectively. The spherical linear interpolation (Slerp) of quaternions can smoothly interpolate rotations, and is given by

$$Slerp(q_0, q_1, u) = \frac{q_0 \sin((1-u)\theta) + q_1 \sin(u\theta)}{\sin(\theta)}, \quad (8)$$

where $\theta = \arccos(q_0, q_1)$. We can then compute a rotation matrix for any value of u by simply converting $Slerp(q_{\text{left}}, q_{\text{right}}, u)$ back into a matrix, $M(u)$.

To align the rotated meshes with the curve we let the first basis vector be given by $\tilde{T}(u) = T(u)$, and let the estimated second basis vector be given by $b(u) = M(u)\mathbf{u}$ where $\mathbf{u} = [0, 1, 0, 0]^T$. To maintain orthogonality of basis vectors we subtract the tangential component to obtain $\tilde{B}(u) = b(u) - (b(u) \cdot \tilde{T}(u))\tilde{T}(u)$. Then the third basis vector can be defined as $\tilde{N}(u) = \tilde{B}(u) \times \tilde{T}(u)$.

This approach solves the problem of defining orientations along a flat curve. However, the results are unsatisfactory when the controllers rotate more than 180° relative to each other. Since the

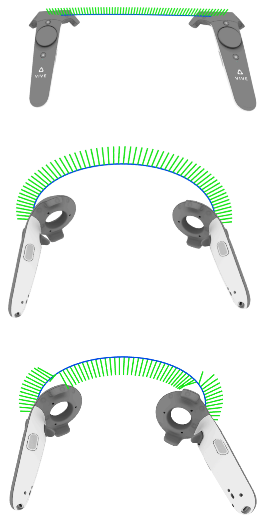

Figure 4: Hermite spline (blue) and a basis vector (green) computed with quaternion Slerp. Note the sudden flip in basis vector when interpolating through more than 180° in the bottom image.

quaternion Slerp constructs the shortest path between two rotations there will be discontinuity as the orientations pass through a 180° rotation. This behavior can be seen in Figure (4).

3.1.5 Rotation minimizing frame. We employ a rotation minimizing frame [Wang et al. 2008] to resolve the problems with the Frenet and Slerp approaches. From that frame we compute a rotation matrix which can be applied to cloned meshes. Rotation minimizing frames are often used to define the geometry of extruded tubular structures and for defining camera paths. In both of these applications the drawbacks of the Frenet and quaternion approaches can have serious consequences. For example, interpolated camera paths defined in terms of the Frenet frame can rotate erratically during linear motion. The "double reflection" algorithm for computing the rotation minimizing frame has be rewritten below in terms of the Hermite spline (see Algorithm 1).

3.1.6 Interpolating the end conditions. The implementation described by Wang et al. [Wang et al. 2008] constructs a rotation minimizing frame incrementally for a discretized curve by computing a rotation minimizing normal vector $\tilde{\mathbf{N}}(u_{i+1})$ from $\mathbf{s}(u_i)$, $\mathbf{s}(u_{i+1})$, $\mathbf{T}(u_i)$ and $\mathbf{N}(u_i)$. In our implementation we recover a continuous frame by analytically computing $\mathbf{T}(u)$, and interpolating $\tilde{\mathbf{N}}(u)$, then orthogonalizing the frame using the Gram-Schmidt process.

The incremental nature of Algorithm 1 means that the left controller orientation, U_0, is interpolated, but the right controller ,U_n, is not. In order to guarantee that the computed frame at the right controller matches the right controller orientation, the authors [Wang et al. 2008] suggest determining the rotation angle, α, required at the right endpoint of the curve, and defining the linear angle function $\alpha(u) = \alpha u$. This function describes the rotation angle about the tangent direction to be applied when computing the rotation

Algorithm 1 Computing the rotation minimizing frame

procedure DOUBLE REFLECTION
Input: Discretized parameter values u_i, $i = 0, 1, \ldots n$ and an initial frame $U_0 = \{\tilde{T}_0, \tilde{B}_0, \tilde{N}_0\}$.
Output: $U_i = \{\tilde{T}_i, \tilde{B}_i, \tilde{N}_i\}$, $i = 0, 1, \ldots n$
  for $i = 0$ to $n - 1$ **do**
    $v_1 = \mathbf{s}(u_{i+1}) - \mathbf{s}(u_i)$
    $c_1 = c_1 \cdot c_1$
    $N_i^L = \tilde{N}_i - (2/c_1) * (v_1 \cdot \tilde{N}_i) * v_1$
    $T_i^L = T(u_i) - (2/c_1) * (v_1 \cdot T(u_i)) * v_1$
    $v_2 = T(u_{i+1}) - T_i^L$
    $c_2 = v_2 \cdot v_2$
    $\tilde{N}_{i+1} = N_i^L - (2/c_2) * (v_2 \cdot r_i^L) * v_2$
    $\tilde{B}_{i+1} = T(u_{i+1}) \times \tilde{N}_{i+1}$
    $U_{i+1} = \{\tilde{T}(u_{i+1}), \tilde{B}_{i+1}, \tilde{N}_{i+1}\}$
  end for
end procedure

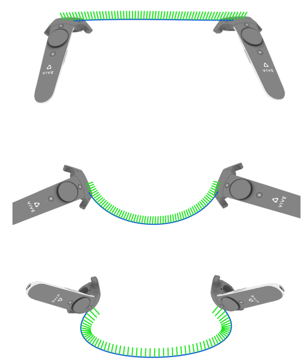

Figure 5: Hermite spline (blue) and a basis vector (green) of the rotation minimizing frame with end condition interpolation.

minimizing $\tilde{N}(u)$. Results of computing the rotation minimizing frame with endpoint interpolation is shown in Figure (5).

Lastly, the translation matrix $D(u)$ is computed from $\mathbf{s}(u)$, and the rotation matrix is constructed column-wise $R(u) = [\tilde{T}(u)|\tilde{B}(u)|\tilde{N}(u)]$. Replicated meshes are rendered with $D(u)R(u)$ as the object-to-world transformation matrix.

3.1.7 Tracking twist angle. Unlike the quaternion solution, the rotation minimizing frame does not suffer from discontinuous rotations when rotating more than 180 degrees, but the rotation will snap back to 0° when rotating past 360°. To handle an extended range of rotation angles we keep track of k - how many times the right controller has rotated through 360° relative to the left controller and incorporate that count in the function $\alpha(u) = (\alpha + 360°k)u$. The result of rotating the right controller multiple times relative to the left controller is shown in Figure (6).

3.1.8 Arc-length parameterization. We specify unit tangent vectors m_0 and m_1 when defining the Hermite curve, but in order to give us better control over object positioning we iteratively refine

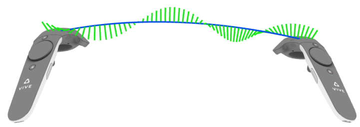

Figure 6: The rotation minimizing frame can interpolate through multiple full rotations of the twist angle. In this example the right controller has been rotated two full turns relative to the left controller.

the vector magnitudes to give us an approximate arc-length parameterization of $\mathbf{s}(u)$. The Hermite tangent vectors are computed as

$$\begin{aligned} m_0 &= c_0 M_{\text{left}} \mathbf{r}, \\ m_1 &= c_1 M_{\text{right}} \mathbf{r}. \end{aligned} \tag{9}$$

where c_0 and c_1 are initialized to 1. Then we use the simple heuristic in Equation (10) to adjust the lengths of the end tangents m_0 and m_1 to approximate an arc length parameterization,

$$\begin{aligned} c_0 &= c_0 \frac{||s'(\frac{1}{2})||}{||s'(0)||}, \\ c_1 &= c_1 \frac{||s'(\frac{1}{2})||}{||s'(1)||}. \end{aligned} \tag{10}$$

Our rationale behind this procedure is that if the speed of the parameterization at an endpoint ($||s'(0)||$ or $||s'(1)||$) is greater than the speed at the middle of the curve ($||s'(\frac{1}{2})||$) then the length of the tangent at that endpoint should be decreased. Similarly, if the speed at the endpoint is slower than the speed at the midpoint the corresponding tangent vector should be lengthened. In our experiments this iterative procedure evaluated once per frame is stable and results in a visually uniform spacing.

3.1.9 Reparameterization of $s(u)$. The curve $\mathbf{s}(u)$ defines the spacing of cloned meshes along the Hermite spline. To permit variable spacing we define the reparameterized curve $\mathbf{s}(v(u))$. Reparameterization will not change the shape of the curve, unlike changing the length of the end tangents. We define $v(u)$ in terms of the Hermite cubic function

$$v(u) = v_0 H_0(u) + w_0 H_1(u) + w_1 H_2(u) + v_1 H_3(u). \tag{11}$$

Since we must maintain the range of parameter values $v \in [0, 1]$ the values $v_0 = 0$ and $v_1 = 1$ are fixed. Some example $v(u)$ curves are shown in Figure (7). When $w_0 = w_1 = 1$ we obtain $v(u) = u$ which results in no reparameterization. As the value of w decreases the curve $v(u)$ flattens out at the corresponding endpoint slowing down the parameterization speed and making more u values map to the same v. This causes the replicated meshes to be bunched together at one end of the curve. As parameterization speed increases the meshes spread apart. Example reparameterized curves with variable spacing along the curve are shown in Figure (8).

The parameters w_0 and w_1 are free parameters which we permit the user to control by swiping the controller trackpads. Swiping in an outward fashion (to the right on the right controller, or to the left on the left controller) increases the speed of parameterization

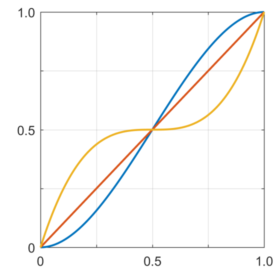

Figure 7: A variety of reparameterization functions $v(u)$ are possible given end slopes w_0 and w_1 in the range $[0, 3]$.

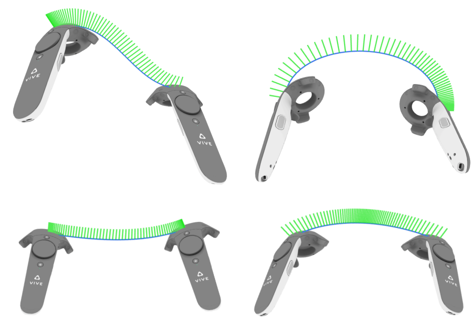

Figure 8: Example reparameterized curves of $\mathbf{s}(v(u))$. The basis vectors in green are plotted at equally spaced intervals in u.

at the corresponding end of the curve. This allows the user to slide the replicated meshes along the curve.

3.2 Modeling with Hermite motion controller

In this section we describe the three modeling modes that our experimental modeling technique supports: swept surfaces, generalized tubes and cloned meshes.

3.2.1 Swept surfaces. When the trigger on the right controller is pulled a swept surface is generated from a sequence of Hermite splines, $\{s_1, s_2, ...s_m\}$. An example surface is shown in Figure (9). A new curve, s_{m+1}, is appended to the swept surface when the maximum distance between corresponding points on the curves s_m and s_{m+1} exceeds a given threshold, ϵ. This allows the user to stop moving the controllers without multiple curves stacking together on the swept surface.

Since there is no parametric smoothness guaranteed along the n direction of the swept surface we give the user the option to smooth the surface in that direction by averaging neighboring curves. The smoothing procedure

$$s_i = (1 - \alpha)s_i + \frac{\alpha}{2}(s_{i-1} + s_{i+1}), \tag{12}$$

is applied to the curves s_2 through s_{m-1}. In our modeling application we use $\epsilon = 0.01$ and $\alpha = 0.1$. The low value of α results in a small amount of smoothing. Stronger smoothing results can be obtained through repeated smoothing.

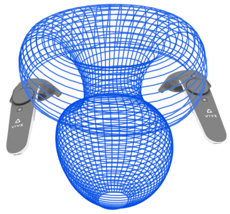

Figure 9: A swept surface generated by moving the motion-tracked controllers.

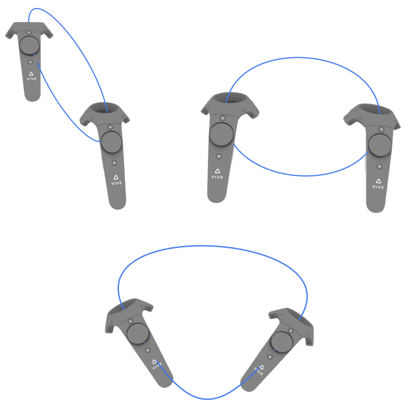

Figure 10: Example closed curves which can be generated by the Hermite motion controller.

3.2.2 Generalized tubes. The user may optionally choose to model with a closed Hermite spline, resulting in tubular structures. The second curve is automatically computed from the first, so the user isn't required to specify any additional parameters to control the closed curve. The second curve simply uses end tangent vectors $-\mathbf{m}_0$ and $-\mathbf{m}_1$. Sample closed curves are shown in Figure (10).

3.2.3 Cloned meshes. Our third modeling mode replicates other meshes which are loaded separately by the user. The meshes are distributed along the swept surface. The orientation of the meshes is aligned with the Hermite motion controller by using the rotation minimizing frame described in Section 3.1.5. Mesh spacing and scaling is determined by the reparameterization of the curve. Cloned meshes distributed along the Hermite curve with different parameterizations are shown in Figure (11). The mesh scaling factor is set to $\mathbf{s}'(v)$ which makes closely spaced meshes smaller to prevent them from overlapping. Replicated meshes are efficiently rendered using the geometry instancing functionality of OpenGL [Shreiner et al. 2013]. This feature reduces client application overhead by permitting multiple copies of a mesh to be rendered with a single draw call.

During interaction we display surfaces and meshes in wireframe to minimize visual occlusion, but in the Results section we will present rendered images of the shapes exported from our modeling system.

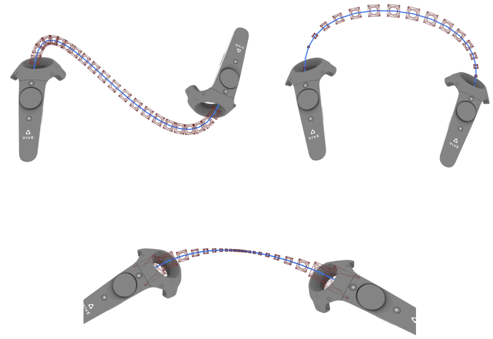

Figure 11: Cloned meshes distributed along the Hermite control curve. Reparameterizing the curve with the trackpad changes the spacing and scale of the meshes.

Figure 12: A user creating surfaces with our system.

4 RESULTS

Our modeling technique was implemented in C++, OpenGL and OpenVR on a PC with a 3.0 GHz Intel Core i7-6800K CPU, 16 GB RAM, Nvidia GeForce GTX Titan X with 2688 shader cores and 12 GB GDDR5 dedicated video RAM, and an HTC Vive VR system.

Figure (12) shows some of the controller gestures made by a user when interacting with our experimental testbed. In informal demonstrations a user mentioned that the Hermite spline controller felt "like a Slinky" due to the way the curve naturally bends and the way the reparameterization stretches and compresses space, like the coils of a spring. This is surprising because the formulation is strictly geometric - no physics are being simulated. But the comment does suggest to us the possibility of incorporating gravity, damping and other physical effects in future work.

4.1 Timing

Timing results are presented in Table (1). Results are expected to be view-dependent and also depend on the degree of tessellation of the Hermite curve (n), the number of curves (m), and the number of vertices in the cloned meshes (p), so timing values for several different scenarios are presented. If cloned meshes are not being used then $p = 0$. The number of cloned meshes is nm. Note that these are the times required to render a stereo pair of images including

Table 1: Timing results

Model complexity	Frames per second (fps)
$n = 48, m = 271, p = 0$	116
$n = 48, m = 2131, p = 0$	114
$n = 64, m = 125, p = 240$	97.2
$n = 64, m = 644, p = 240$	96.4

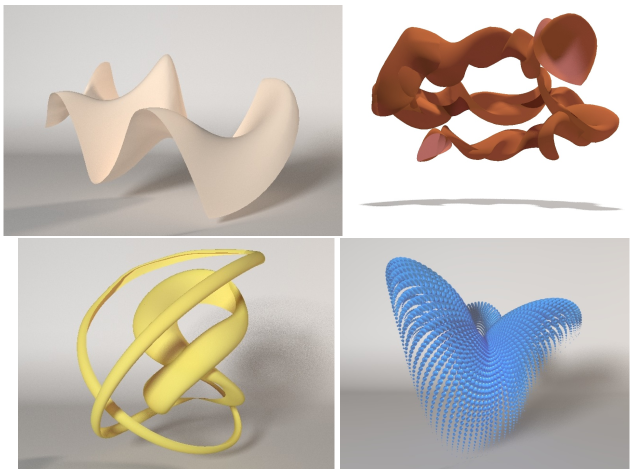

Figure 13: Swept surface (top-left) and swept tubes (top-right, bottom-left) and cloned meshes (bottom-right).

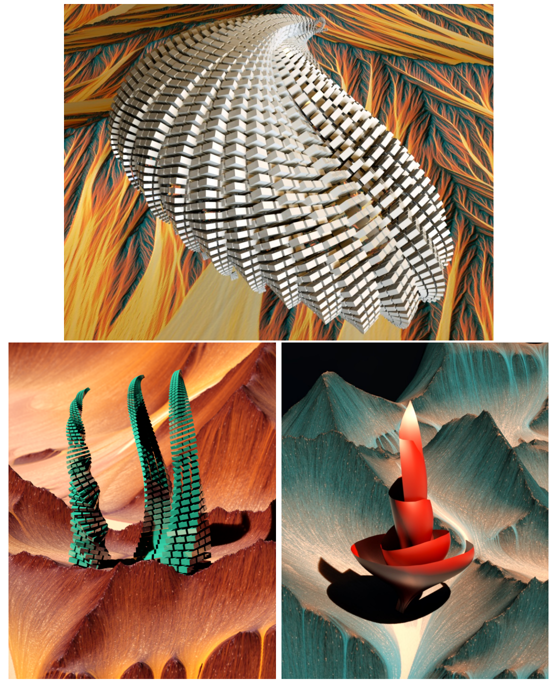

Figure 14: An artist incorporated several of our meshes into landscapes generated in SideFX Houdini. The meshes designed using our technique are in the foreground.

our GUI, apply postprocessing to correct for barrel distortion and chromatic aberration, and respond to user input. In our application cloned meshes are only rendered along the control spline during modeling to avoid visual clutter. Note that all of these framerates are above the 90 fps threshold recommended to avoid simulator sickness [Earnshaw 2014].

4.2 Rendered results

Our VR-based modeling technique with the Hermite spline controller described in Section 3.1 was used by the authors to interactively create the models shown in Figures (1, 13, and 14). Results are shown without texture mapping, but the surfaces we generate are trivial to texture map since they are based on parametric curves.

Rendering results created by an artist combining meshes designed in VR with procedural landscapes are shown in Figure 14. The landscapes were generated in SideFX Houdini and rendered with Otoy OctaneRender.

A family of smooth abstract forms are easy to model with this technique, but representational modeling and meshes with creases and corners will require additional tools.

4.3 Digital fabrication

For this project we also exported 3D models and manufactured them through two different techniques: 3D printing and laser cutting. The .OBJ file sculpted in the virtual reality world was easily transfered to the physical world through readily available fabrication hardware. We were inspired by Robert Mallary's TRAN2 [Mallary 1970] concept of using the computer as a tool for modeling real objects. The meshes were preprocessed to ensure that the orientation of each polygon's normal was facing outwards. Then we proceeded to run our 3D model through Autodesk Netfabb to optimize the 3D slices. The first example shown in Figure 15 (top), was printed using a LulzBot TAZ 5 3D printer and the material was PLA. We used more infill and support material due to the high complexity of the mesh. The two examples on the bottom of Figure 15 are made of acrylic sheets and cut using a Boss Laser cutter LS-2436. Autodesk 123D Make software was used to create 2D slices from the 3D models designed in VR.

5 CONCLUSIONS AND FUTURE WORK

In this work we have described and demonstrated an interactive curve and surface modeling technique which can be used for virtual surface modeling using a pair of motion-tracked VR controllers. An OpenGL and OpenVR implementation of the mesh modeling process was described, and preliminary modeling results were presented. Our implementation is able to maintain the high framerates required for VR applications.

Our system is still in an experimental state, but after further development and refinement we intend to perform user studies in order to assess the usability of the technique and guide further interaction design. In our preliminary experiments we did not encounter significant performance issues, but supporting very large meshes will likely require additional optimizations. A full featured

Figure 15: Examples of our sculptures fabricated with a 3D printer and laser-cut acrylic

modeling system will require us to develop new rendering techniques and interaction approaches. We intend to design additional modeling modes to augment the three presented here.

REFERENCES

Alexis Angelidis, Marie-Paule Cani, Geoff Wyvill, and Scott King. 2006. Swirling-sweepers: Constant-volume modeling. *Graphical Models* 68, 4 (2006), 324–332.

Rahul Arora, Rubaiat Habib Kazi, Fraser Anderson, Tovi Grossman, Karan Singh, and George Fitzmaurice. 2017. Experimental Evaluation of Sketching on Surfaces in VR. (2017).

Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. ACM, 151–160.

Steve Bryson. 1996. Virtual reality in scientific visualization. *Commun. ACM* 39, 5 (1996), 62–71.

Steve Bryson and Creon Levit. 1991. The virtual windtunnel-An environment for the exploration of three-dimensional unsteady flows. In *Visualization, 1991. Visualization'91, Proceedings., IEEE Conference on*. IEEE, 17–24.

Valve Corporation. 2017. OpenVR. https://github.com/ValveSoftware/openvr. (2017).

Carolina Cruz-Neira, Daniel J Sandin, and Thomas A DeFanti. 1993. Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. ACM, 135–142.

Frank Dachille IX, Hong Qin, Arie Kaufman, and Jihad El-Sana. 1999. Haptic sculpting of dynamic surfaces. In *Proceedings of the 1999 symposium on Interactive 3D graphics*. ACM, 103–110.

Tushar H Dani and Rajit Gadh. 1997. Creation of concept shape designs via a virtual reality interface. *Computer-Aided Design* 29, 8 (1997), 555–563.

Char Davies and John Harrison. 1996. Osmose: towards broadening the aesthetics of virtual reality. (1996).

Cagatay Demiralp, Cullen D Jackson, David B Karelitz, Song Zhang, and David H Laidlaw. 2006. Cave and fishtank virtual-reality displays: A qualitative and quantitative comparison. *IEEE Transactions on Visualization and Computer Graphics* 12, 3 (2006), 323–330.

Rae A Earnshaw. 2014. *Virtual reality systems*. Academic press.

Gerald Farin. 2014. *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*. Elsevier.

E. Garcia and T. McGraw. 2016. Bodygraphe: Gestural Computing for Visual Music. In *Expressive 2016 - Posters, Artworks, and Bridging Papers*. The Eurographics Association. DOI:http://dx.doi.org/10.2312/exp.20161263

Gabriele Guidi, Michele Russo, and Davide Angheleddu. 2014. 3D survey and virtual reconstruction of archeological sites. *Digital Applications in Archaeology and Cultural Heritage* 1, 2 (2014), 55–69.

Chris Gunn. 2006. Collaborative virtual sculpting with haptic feedback. *Virtual Reality* 10, 2 (2006), 73–83.

Rorik Henrikson, Bruno Araujo, Fanny Chevalier, Karan Singh, and Ravin Balakrishnan. 2016. Multi-Device Storyboards for Cinematic Narratives in VR. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, 787–796.

Bret Jackson and Daniel F Keefe. 2016. Lift-Off: Using Reference Imagery and Freehand Sketching to Create 3D Models in VR. *IEEE transactions on visualization and computer graphics* 22, 4 (2016), 1442–1451.

Daniel F Keefe, Daniel Acevedo Feliz, Tomer Moscovich, David H Laidlaw, and Joseph J LaViola Jr. 2001. CavePainting: a fully immersive 3D artistic medium and interactive experience. In *Proceedings of the 2001 symposium on Interactive 3D graphics*. ACM, 85–93.

Daniel F Keefe, Robert C Zeleznik, and David H Laidlaw. 2007. Drawing on air: Input techniques for controlled 3D line illustration. *IEEE Transactions on Visualization and Computer Graphics* 13, 5 (2007), 1067–1081.

Marius Kintel and Clifford Wolf. 2011. OpenSCAD, The Programmers Solid 3D CAD Modeller. (2011).

Sam Machkovech. 2016. Learning how to VR with Tilt Brush, HTC Vivefis killer app. *Ars Technica* 5 (2016).

Robert Mallary. 1970. TRAN2: A Computer Graphics Program to Make Sculpture. In *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference (AFIPS '70 (Fall))*. ACM, New York, NY, USA, 451–460. DOI:http://dx.doi.org/10.1145/1478462.1478527

James McCrae and Karan Singh. 2009. Sketching piecewise clothoid curves. *Computers & Graphics* 33, 4 (2009), 452–461.

Jung Nam and Daniel F Keefe. 2016. Spatial correlation: An interactive display of virtual gesture sculpture. *Leonardo* (2016).

Todd Palamar. 2013. *Mastering Autodesk Maya 2014: Autodesk Official Press*. John Wiley & Sons.

Steven Schkolne, Michael Pruett, and Peter Schröder. 2001. Surface drawing: creating organic 3D shapes with the hand and tangible tools. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 261–268.

Stanley E Sclaroff, Alex Pentland, Irfan Essa, Martin Friedmann, and Bradley Horowitz. 1990. The thingworld modeling system: Virtual sculpting by modal forces. In *ACM SIGGRAPH Computer Graphics*, Vol. 24. ACM, 143–144.

Dave Shreiner, Graham Sellers, John M Kessenich, and Bill M Licea-Kane. 2013. *OpenGL programming guide: The Official guide to learning OpenGL, version 4.3*. Addison-Wesley Professional.

Mel Slater and Martin Usoh. 1994. Body centred interaction in immersive virtual environments. *Artificial life and virtual reality* 1 (1994), 125–148.

Scott Spencer. 2011. *ZBrush Character Creation: Advanced Digital Sculpting*. John Wiley & Sons.

Ivan E Sutherland. 1968. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*. ACM, 757–764.

Rensu Theart, Ben Loos, and Thomas Niesler. 2017. Virtual reality assisted microscopy data visualization and colocalization analysis. *BMC Bioinformatics* 18, 64 (2017).

Wenping Wang, Bert Jüttler, Dayue Zheng, and Yang Liu. 2008. Computation of rotation minimizing frames. *ACM Transactions on Graphics (TOG)* 27, 1 (2008), 2.

Colin Ware, Kevin Arthur, and Kellogg S Booth. 1993. Fish tank virtual reality. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*. ACM, 37–42.

Tamar Hei Ting Wong and Kee Ka Ki Wong. 2016. VRgram. In *SIGGRAPH ASIA 2016 VR Showcase*. ACM, 9.