Vision, Modeling, and Visualization (2019)
H.-J. Schulz, M. Teschner, and M. Wimmer (Eds.)

RodMesh: Two-handed 3D Surface Modeling in Virtual Reality

Floor Verhoeven and Olga Sorkine-Hornung

ETH Zurich, Switzerland

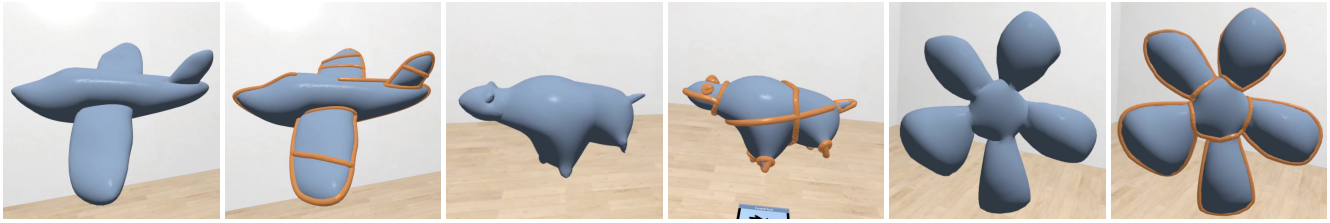


Figure 1: *Manifold mesh models created using RodMesh on Oculus Rift, with and without the deformable control rods visualized. Users can define and manipulate these control rods to edit the mesh surface.*

Abstract

User interfaces for 3D shape modeling in Virtual Reality (VR), unlike basic tasks such as text input and item selection, have been less explored in research so far. Shape modeling in 3D lends itself very well to VR, since the 3D immersion provides the user with richer spatial feedback and depth perception when compared to traditional 2D displays. That said, currently existing 3D modeling applications do not focus on optimizing the modeling interaction techniques for VR, but instead mostly merely port standard interaction paradigms. Our approach utilizes a popular sketch-based surface modeling algorithm in VR by rethinking the user interface in order to benefit from the 3D immersive environment and its inherent support of two-handed input. We propose a bimanual interaction technique that allows users to create 3D models via virtual deformable rods. These rods are bendable into outline shapes that are automatically inflated into manifold mesh surfaces, and can then be incrementally edited to further refine the shape.

CCS Concepts

• ***Human-centered computing*** → *Virtual reality;* • ***Computing methodologies*** → *Shape modeling;*

1. Introduction

Digital modeling of freeform 3D shapes is a challenging task for casual users and remains time consuming even for experts. The traditionally cited chief reasons for this are the required mentally taxing translation between the imagined 3D shape and its 2D rendering on a 2D display, as well as the 2D input devices (mouse, stylus, touchpad), which cannot directly provide 3D spatial information to the modeling system. Virtual reality enables convincing immersion of the user in the digital 3D space and hence holds the promise to alleviate the aforementioned limitations. However, new challenges and open questions emerge with regards to input modalities and interaction techniques in VR to facilitate efficient and at the same time precise user control of the 3D shape. In the traditional 2D desktop setup, sketch based 3D modeling systems [IMT99, NISA07] capitalize on commonly possessed 2D drawing skills of users and provide an intuitive and very economical way to specify 3D shapes: they enable users to span a whole surface in 3D by drawing and manipulating a sparse set of 2D curves with the mouse or the stylus. However, because drawing 3D curves in space is not a commonly practiced skill, and because it is imprecise along the z (depth) direction, a one-to-one translation to drawing 3D curves in VR is unsuitable.

We propose a solution to this problem by benefiting both from the 3D immersion advantages of VR and the economical, sparse input of traditional curve-based surface modeling systems. Our core idea is to exploit the unique availability of two-handed spatial input modality in VR for this purpose. We take FiberMesh [NISA07], an algorithm for modeling a manifold mesh surface from a set of curves in 3D, as the basis, and we propose to replace the drawing of the curves by two-handed bending and stretching of virtual rods that represent the curves. In the physical reality, the user holds a tracked 6 DOF controller in each hand, while in VR a curve is ren-

DOI: 10.2312/vmv.20191312

dered as a rod, and the user appears to grasp the rod at its ends with the virtual hands. The rod can hence be bent and stretched by movements of the hands, and users are given the tactile impression of actually holding the rod's ends, since in reality they are holding the controllers. The backbone algorithm, taken from FiberMesh, then spawns the surface mesh that interpolates the 3D curve(s) with a plausible, smooth geometry. We describe additional tools to facilitate the editing and incremental updating of the modeled shape.These tools are inspired by FiberMesh but have been adapted for more intuitive control in VR. We demonstrate that our RodMesh system enables users to quickly and intuitively create a variety of high quality surfaces in VR.

Similarly to the FiberMesh system [NISA07], RodMesh is not intended for producing intricately detailed surfaces, but rather aims at enabling quick drafts and experimentation with 3D shape ideas. The output of RodMesh is a manifold triangle mesh that can serve as the starting point for adding geometry and raster textures, creating subdivision surfaces and other advanced modeling operations.

2. Related work

A significant amount of professional software for the purpose of 3D modeling is available; some of the most known mesh modeling packages include Autodesk 3ds MAX [3ds19], AutoCAD [Aut19], Blender [Ble19] and Maya [May19]. Typically these programs provide the user with very fine-grained control over the final mesh by providing a plethora of editing tools and even allowing to modify the mesh geometry and topology on a vertex, edge or face level. Although this makes these programs very powerful and versatile, using the programs demands a steep learning curve and requires the user to specify accurate and detailed input. A common observation from users is that one must already have a precise idea of the desired shape in mind, developed previously by other means, and then using these professional tools one can input this idea into the computer [NISA07]. By contrast, the aim of our work is to empower non-expert users to quickly create 3D models and experiment with various shape ideas directly in the digital modeling application. Therefore in this section we focus on reviewing the more intuitive and easy-to-use 3D modeling paradigms meant for rough drafts, as well as methods for creating 3D models specifically in VR.

2.1. Two-handed input

Bimanual input has been adressed in various research works in the context of computer graphics and geometric modeling in particular. Grossman et al. [GBS03] study two-handed input specifically for manipulating 3D curves. They propose a physical interaction device called ShapeTape, a fixed-length tape-like construct that contains fiber optic sensors that can reconstruct the shape of the tape in 3D space. They employ it for specifying deformations of 3D curves: Users can bend and twist the physical tape to control a digital representation of the curve. In addition to purely specifying the shape of a curve, ShapeTape can also be used to specify gestures that are mapped to specific editing actions. The modular physical articulation device by Glauser et al. [GMP*16] can also be used to construct an input device for capturing a 3D curve shape by connecting their multiple universal joints into a chain.

Subsequent research by Grossman et al. [GBK*01] employs the ShapeTape device to create wireframe models. Their study shows that bimanual input for working with curves is a successful interaction technique, and that defining curves with ShapeTape is significantly less complicated than the traditional methods for creating curves. This approach shares similarities with our work in the sense that the tool presents the user with a direct mapping between the sensory feedback and the behavior of the digital curve. Although RodMesh merely equips the user with a virtual deformable rod instead of a physical one, the two hand-held controllers do elicit the sensation of holding a rod-like object. An advantage of our approach is that users do not have to work against gravity that is pulling on the physical rod, and they are in general nearly unconstrained by the physics of the real world and the physical input device. In RodMesh, users can "let go" of the virtual rod without it falling to the floor; the middle portion of our virtual rod does not fall downwards when holding the rod by its endpoints, and the rod can be stretched to various lengths, unlike the physical ShapeTape.

Two-handed touch input for 3D modeling or deformation operations has been studied in multiple works. Igarashi et al. [IMH05] use multitouch input on a SmartSkin touchpad to deform existing 2D shapes in an as-rigid-as-possible manner. Palleis et al. [PWH16] apply two-handed touch input to CAD-like edge-loop scaling and extrusion operations, which users perceive as easy to learn and some find it comparable to modeling with clay. Kang et al. [KKSH15] develop a multitouch 3D modeling system that converts sketch input to CAD-like surfaces. With simple gestures, users can quickly prototype 3D models that can be then further refined on a PC. All these works utilize 2D touch interfaces. Expanding on the purely touch-based input space, Mockup builder [DACJ12] allows bimanual input from both multitouch interaction on a flat surface as well as 3D positional finger tracking above the surface for a set of 3D modeling operations (e.g., extrusion along a curve, scaling, offsetting planes).

Llamas et al. [LKG*03] develop an approach for two-handed 6 DOF editing of 3D shapes by means of space warping, called Twister. Their approach allows users to use both hands simultaneously to define positional and orientation constraints on rigid handles, which are converted to a space warp that decays proportionally to the distance from the handles. This results in a real-time interaction technique that is very straightforward to use. Our approach also allows the user to define both positional and orientation constraints with both hands. The difference is that we restrict the handles to be on the control rods that define the surface via an energy optimization, whereas Twister [LKG*03] is an FFD, or space warp deformation that controls the surface shape in an indirect way, leading to different effects.

A study by Buxton and Meyers [BM86] suggests that two-handed input provides most advantage for tasks that would require a lot of movement or "travelling" when performed with one hand. This applies to our application scenario, since (symmetrically) bending a rod with one hand would require a sequence of deformations performed on different parts of the rod.

Besançon et al. [BIAI17] discover that mouse-based, touch-based and tangible input devices are all equally well suited for precise 3D positioning tasks (docking a virtual 3D object at a prede-

fined position). They also find that the tangible input device results in the fastest completion times and has the most intuitive mapping. However, compared to mouse- and touch-based input, physical fatigue is more prevalent.

2.2. Sketch-based 3D shape modeling

Sketch-based 3D modeling is a technique that aims to transfer the way that people draw shapes with pen and paper to a method of modeling 3D shapes on the computer using a 2D mouse or stylus. The user draws 2D curves on the screen, which are then processed and inflated into 3D meshes using automatic surface creation algorithms. Typically the user then has the option to further edit this initial mesh by adding further sketched lines.

Teddy [IMT99] is a milestone work on interfaces for sketch-based 3D design that allows the user to sketch 2D input strokes, which are then automatically translated into 3D shapes. FiberMesh [NISA07] generalizes Teddy by additionally keeping the user-defined curves visible after the initial inflation, allowing the curves to be used for further shape editing, such as stretching the mesh by pulling on one of the curves. ShapeShop [SWSJ06] extends these sketch-based modeling tools with the usage of BlobTrees and CSG operations, allowing users to create combinations of smooth freeform and CAD-style 3D models, with a high level of detail. As stated earlier, we opt to employ the surface modeling algorithm of FiberMesh [NISA07] in our system due to its versatility and encapsulation of 3D curved based modeling, but we replace the sketch-based input modality, since it is less intuitive in 3D VR.

Shtof et al. [SAG*13] show another approach, where the user defines the basic outlines of a 3D shape by sketching, and the actual 3D model is created by assigning predefined 3D primitives to the different semantic parts of the model. Although the initial input is sketch-based and sparse, the required subsequent refinement demands a lot of input from the user. Additionally, the user is restricted to models that can be composed of the predefined primitive shapes, and occluded parts cannot be generated since the modeling only occurs from one viewpoint.

More recently, sketch-based 3D modeling has also been combined with deep neural networks in an attempt to create more accurate 3D models from 2D inputs that are provided from a single viewpoint [LPL*18]. However, this method requires various amounts of additional input from the user in order to provide the system with information about the relative depth in the model to be constructed.

2.3. 3D Modeling in VR

Recently, a number of VR applications for creating digital 3D content have appeared [Goo19, Til19, Gra18, Ocu19, PIA10]. These applications can roughly be split into two categories: creating 3D "paintings" and creating 3D models, assembled from a structure of vertices, edges and faces. We focus on the latter category, since the former typically does not result in manifold surface objects.

Commercially available apps for 3D modeling in head mounted VR are for example Google Blocks [Goo19], Gravity Sketch VR [Gra18] and Oculus Medium [Ocu19]. Their concepts vary from playful modeling [Goo19] to allowing high-detail modeling with a very extensive tool set [Gra18, Ocu19]. Google Blocks [Goo19] is the simplest of the three, allowing users to assemble different primitives (cubes, cones, spheres) into a new model. Gravity Sketch [Gra18] and Oculus Medium [Ocu19] provide a wider variety of modeling tools, with Gravity Sketch focusing more on a CAD-like approach, whereas Oculus Medium mimics modeling with clay and employs implicit surfaces.

In addition to the commercial apps, in academia Perkunder et al. [PIA10] created a version of FiberMesh [NISA07] for an immersive 3D environment, a CAVE with five rear-projection walls and active stereo LCD shutter glasses. Their user study shows that users find modeling in an immersive 3D environment more supportive of the creative process compared to performing the same modeling task in a non-immersive environment. At the same time, their work is a direct one-to-one adaptation of FiberMesh and its original user interface [NISA07] to a CAVE setting, without exploiting the unique benefits of VR and optimizing the interface to 3D immersion.

In concurrent work to our own, Rosales et al. present SurfaceBrush [RRS19], a VR application that has an interaction paradigm very similar to Google TiltBrush [Til19]. Whereas TiltBrush generates 3D paintings only, SurfaceBrush is capable of creating actual manifold surfaces from the 3D strokes. Users can create surfaces by drawing ribbon strokes in the shape of the intended surface. The center line positions and normal orientations (as determined by the orientation of the VR controller) of these ribbon strokes are used to compute the corresponding surface. The application is shown to be useful for both amateurs and artists to effectively communicate shapes.

Although Google Blocks is very accessible to novice users, the iterative process can be lengthy and tedious, and the resulting meshes have low polygonal count and resemble 3D pixel art style. In comparison, our application enables the user to quickly create mesh models with a smooth appearance. Oculus Medium and Gravity Sketch require much more (precise) user input compared to our solution. Although this gives the user more control over the final appearance of the models, it also makes it difficult to learn how to use the tools.

3. Method

We implement RodMesh for the Oculus Rift with Oculus Touch controllers (see Figure 2). The controllers are tracked by the Oculus Rift system, such that the position and orientation of each controller is known and can be mapped to the position and pose of the hand in the virtual 3D environment. The base of each controller fits entirely in the palm of an average human hand, which gives a realistic experience of holding a rod.

We employ the surface modeling algorithm from FiberMesh [NISA07], with some adjustments for our setting. Smooth surfaces are created by interpolating the control curves via nonlinear functional optimization. Deformation of curves (and the attached mesh) leads to sequentially performed curve deformation and surface optimization steps, in which the resulting positions

Image source: Samwalton9 on WikiMedia Commons

Figure 2: *Oculus Touch controllers used for RodMesh.*

Figure 3: *The menu for selecting the editing tool. From left to right, top to bottom: new starting rod (requires a reset of the scene); deformation tool; add rod tool; cutting tool; extrusion tool; erase rod tool; save scene; reset to empty scene; load scene from file.*

from the curve deformation step are fed into the surface optimization as positional constraints. Surface mesh optimization consists in numerically solving a 6-th order partial differential equation, and is performed by iteratively solving two sparse linear systems. Their size is on the order of the number of vertices of the mesh, and both systems have constant system matrices (only the right hand side varies). This allows for one-time factorization in a short preprocessing step and facilitates realtime updates of the mesh surface when curve constraints are modified. For further details on the surface optimization algorithm we refer the reader to [NISA07].

Our curve deformation step differs in its details from FiberMesh [NISA07] in order to make the curve feel more rod-like. Internally, we represent the rods as piecewise linear curves that correspond to the midlines of the virtual rods. We store a rod thickness parameter for rendering purposes. Users have the option to either pinch the rod with thumb and index finger, or to grab a part of the rod by making a fist. Pinching and moving the pinched hand constrains the position of the selected vertex and one adjacent edge, while grabbing the rod constrains all vertices and edges that are inside the fist. By twisting their hands, users can specify the desired orientation of the selected edge(s), enabling bending of the rod. This way, the orientation of the controller is implicitly mapped to the local frame of the constrained part of the curve. Note that we do not implement local twisting of the curve about its tangent direction; this could be added as an easy extension by using e.g. Cosserat rod representation of the local frame [SMSH18]. The twisting of the rod based on hand orientation is not present in the original FiberMesh work and would also be hard to operate in a 2D setting.

The deformation of the 3D curve representing the rod is computed by solving Equation (2) in the FiberMesh paper [NISA07]. For completeness and to facilitate our explanation of the implementation, we reproduce the equation here. The vertices of the piecewise linear curve are denoted as $\mathbf{v}_1, \ldots, \mathbf{v}_n$ and the set of the curve's edges as E. The optimization problem to solve in each iteration is

$$\arg\min_{\mathbf{v},\mathbf{r}} \left\{ \sum_{i=1}^{n} \|\mathbf{L}(\mathbf{v}_i) - \mathbf{r}_i\mathbf{R}_i\delta_i\|^2 + \sum_{i\in C_1} \|\mathbf{v}_i - \mathbf{v}_i'\|^2 + \sum_{(i,j)\in E} \|\mathbf{r}_i\mathbf{R}_i - \mathbf{r}_j\mathbf{R}_j\|_F^2 + \sum_{i\in C_2} \|\mathbf{r}_i\mathbf{R}_i - \mathbf{R}_i'\|_F^2 \right\},$$

where C_1, C_2 are the sets of constrained vertices and edges, respectively; $\mathbf{L}$ is the discrete differential operator on the curve; $\mathbf{R}_i$ is the gross rotation obtained from the previous iteration step and fixed in each minimization step; $\mathbf{R}_i'$ are the orientation constraints; δ_i is the differential coordinate of vertex i in the original (undeformed) curve, obtained by applying the operator $\mathbf{L}$ to the original vertex positions; $\mathbf{v}_i'$ are the constrained vertex positions, and $\mathbf{r}_i$ is a linearized incremental rotation represented as a skew symmetric matrix with three unknowns:

$$\mathbf{r}_i = \begin{bmatrix} 1 & -r_{iz} & r_{iy} \\ r_{iz} & 1 & -r_{ix} \\ -r_{iy} & r_{ix} & 1 \end{bmatrix}.$$

Solving the above sparse linear system provides optimal vertex positions and incremental rotations $\mathbf{r}_i$. The target gross rotations $\mathbf{R}_i$ are updated with these incremental rotations and then renormalized via polar decomposition. The iterations are repeated a fixed number of times (twice in our implementation), depending on the required interactive performance and available computational resources. In our prototype the curve deformation runs at an interactive rate of 60 fps.

In our setting, we define the region-of-interest (ROI) to contain all vertices of the rod that the user is deforming, instead of applying the "peeling" approach in [NISA07], since deforming the entire rod appears closer to physically realistic behavior and feels more intuitive. As the set of fixed vertices C_1 we define the handle vertices (or single handle vertex in the case of one-handed deformation); in case the rod is not attached to a mesh yet, we also add the endpoints of the rod as positional constraints. The set of constrained edges C_2 always contains the edge(s) adjacent to the handle(s), and in the case of a grabbing deformation we also add the other edges that fall within the grip of the hand(s). Once a mesh is in place, the rod becomes attached to the mesh (essentially becoming a chain of mesh edges), and we define extra positional constraints for single-handed deformation, because the endpoints of the rod are no longer fixed in place. For this purpose, we fix the rod-vertex that is furthest away from the current handle, giving the user a kind of anchor point to pull against. In the case of two-handed deformation this is not necessary, and we instead simply constrain the two handle vertices,

initial rod | bending the rod into a loop | initial inflated mesh | deforming the mesh via the rod

Figure 4: *Creation of the initial mesh and deforming it.*

starting model | pulling a rod deforms connected rods | two-handed deform | the connected rod moves along

Figure 5: *Examples of pulling on a rod hierarchy.*

and depending on the pulling gesture we fix one or more edges per hand.

As soon as the user releases the rod with the two endpoints of the rod sufficiently close to each other, we sample the gap between the two endpoints and create a double-faced triangulation of the resulting closed polygon, as required for the inflation algorithm [NISA07]. The rod remains open just as it was, and becomes logically attached to the mesh. Keeping the rod open gives the user the possibility to manipulate two points of the rod more independently from each other.

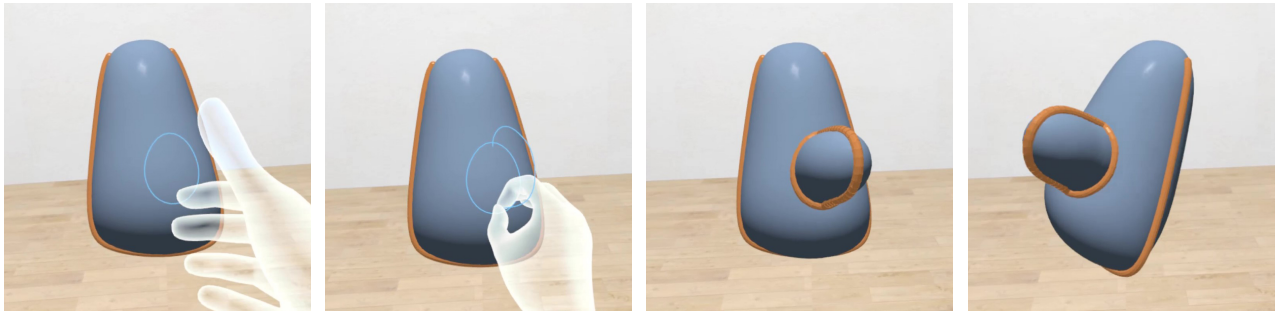
Based on this initial surface mesh, the user can perform further editing operations, such as adding further rods on the mesh surface, which can then again be deformed. Several editing tools can be selected from a menu (Figure 3), as described below and demonstrated in the accompanying video.

Deformation tool. The deformation tool allows the user to deform the control rods and the attached mesh parts by grabbing a part of the rod and pulling it to a new position, possibly also rotating the wrist to specify a new orientation of the rod. The user can either use a grabbing motion (making a fist) to constrain a small segment of the rod, or make a pinching motion with the index finger and thumb to constrain only a single vertex and an adjacent edge. The change in orientation of the controller is transferred to an orientation change of the adjacent edge. In case of a grabbing motion, the orientation change is propagated from the handle vertex outwards over the edges that appear to be in the hand. The existing edges are rotated according to the orientation change, and a translation is applied from the handle vertex outwards. Direct manipulation of the curve in 3D space is possible thanks to the tracking of the VR controllers, whilst FiberMesh only allows 2D manipulations of the curve per view. Figure 4 shows the simplest example of a rod deformation, where the initially straight rod is bent into a U-shape and subsequently meshed. This resulting mesh is then reshaped by deforming the control rod.

When there are multiple control rods defined on the mesh surface, performing a deformation action on one of them by grabbing or pinching propagates to other rods that are attached to it. When propagating down a hierarchy from the rod that is actively being deformed, any connecting vertices are constrained to the position that is computed in the optimization of the parent curve. Figure 5 shows examples of this hierarchical propagation of rod deformation.

In some cases we enforce extra constraints in addition to the constrained vertex and edge(s) at the handles, to ensure deterministic and intuitive behavior. For example, when pulling on a closed rod using only one hand, we constrain the position of the vertex that is furthest away from the handle vertex, to prevent translation of the entire rod and the attached mesh, which automatically happens when only one vertex of a closed rod is fixed. For any closed rod that is (indirectly) attached to the rod that is being deformed, if it has fewer than three constrained vertices, we fix one of its edges that is incident on the vertex that intersects a closed rod higher up in the hierarchy. If none of the rods higher up in the hierarchy are closed, we simply fix all of the current rod's edges that are incident on any of the intersection points with any open rod higher up in the hierarchy. Finally, for all open rods that are (indirectly) attached to the rod at the top of the hierarchy (the one that the handle belongs to), we constrain all edges of the rod that are incident on the connection point to another rod higher up in the hierarchy. The fixing of vertices down the the rod hierarchy is necessary to avoid underdetermined linear systems in the rod deformation optimization (i.e., to prevent situations with too many degrees of freedom).

Cutting tool. The cutting tool is rendered like a laser beam emanating from the pointed forefinger. It can be used to cut away a part of the mesh, see an example in Figure 7 and the video. The user "draws" the cut over the mesh surface, and our system tries to wrap it around on the backside of the mesh based on the direction at which the cutting ray hit the front side of the mesh. Because we use the orientation of the 6-DOF controllers to define a direction on the cutting ray, the user can perform this cutting action with-



defining the base of extrusion | defining the silhouette rod | result, frontal view | result, side view

Figure 6: *An example of an extrusion operation.*

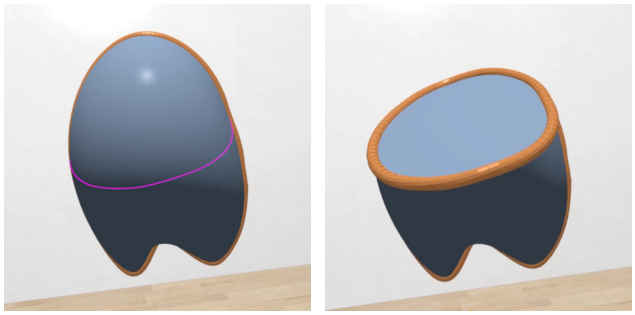


Figure 7: *An example of a cutting operation. Please see the accompanying video for a better visibility of the laser knife.*

out having to frontally face the mesh, which would be required by a traditional 2D cutting input paradigm, where the cutting curve is projected along the viewing direction. The cut curve becomes a rod, and the user points at and clicks on the part of the mesh that is to be removed. The resulting smooth cut surface is variable in two directions, since it is defined both by the position of the cutting curve on the mesh as well as the direction of the laser being used. This creates more interesting surface than what would be possible in a 2D setting.

Extrusion tool. We implement the extrusion tool with the same paradigm as in FiberMesh [NISA07], which works slightly differently from traditional CAD extrusion tools, where the user first defines a cross-section outline and then sweeps the cross-section along a curved path to create a volume. In the FiberMesh surface definition, the surface geometry is entirely determined by interpolating the rods, hence the user defines the base rod for the extrusion on the existing mesh surface with the laser beam interface and then specifies a silhouette rod for the extrusion, drawn from the index finger, such that the silhouette rod attaches to the base at two points. Both sides of the silhouette rod, along with matching parts of the base rod, are triangulated, and their mesh geometry is optimized as usual. Compared to the interface for extrusion silhouette input in FiberMesh, our adapted version does not require the silhouette to be defined from a viewpoint that is orthogonal to the extrusion base, since the 3D position of the curve does not need to be computed – it can be directly read from the VR controllers. See Figure 6 and the video for an illustration.

Rod addition and removal tools. The rod addition and removal tools are also ported from FiberMesh in a straightforward manner. These may be necessary to enable finer control of the 3D shape. Using the laser beam interface, the user can either define new constraint rods that wrap around the entire existing shape (similar to the rod that is generated by a cutting operation) or open rods that lie on the mesh surface. Figure 8 shows examples of both kinds of added rods.

Rods can also be removed again, by simply clicking on the rod that should be removed twice. After a rod is removed, the surface is optimized using the remaining rods. Figure 9 shows an example of removing a rod and the subsequent surface optimization.

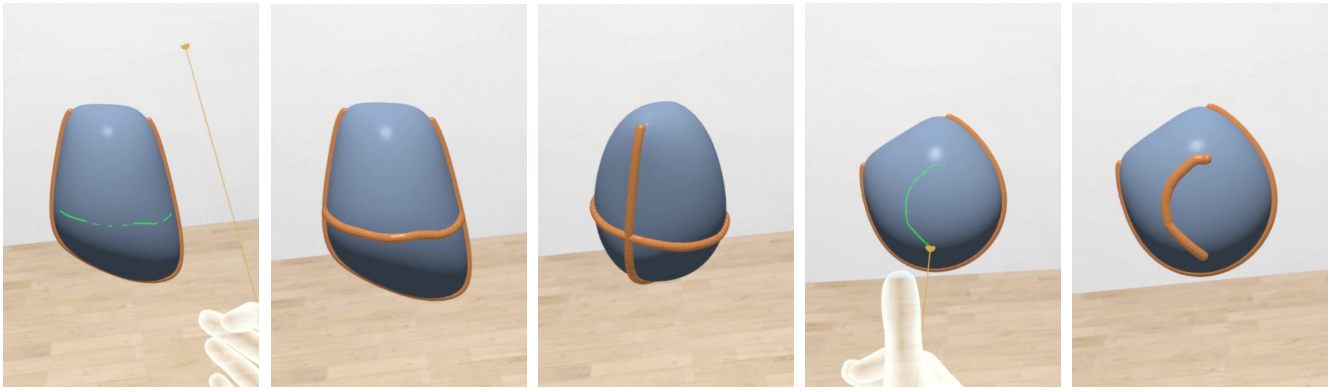
4. Results

We have evaluated our prototype with an informal user study. Test users were given instructions on how to use the application and then requested to provide oral feedback while trying it out.

From our observations, we noticed that users utilize two-handed rod deformation particularly often for bending the initial rod into a closeable shape by grabbing or pinching the endpoints of the rod. Stretching an existing mesh (such as in Figure 5) was also mostly done with two-handed deformation. When working on smaller details (such as small extrusions like the ears, legs and tail of the fantasy animal in the top row of Figure 11), users resorted to one-handed deformation. The main reason for this was that the size of the relevant details was too small for the two Oculus Touch controllers to operate on them simultaneously, because the tracking rings attached to the controllers (see Figure 2) collide.

Similarly to the user study of FiberMesh [NISA07], test users commented that it would be useful to be able to merge model parts that have been created separately from each other. Users suggested that this could simplify the extrusion process, which some of them found difficult to control when performed directly on the mesh surface. Another suggestion was to remove the fixation of the initial rod's endpoints, as they described this as "feeling unnatural".

Our prototype implementation is written in C++ using the Ocu-



adding a closed rod | resulting rod | side view | adding an open rod | resulting rod

Figure 8: *Example of adding constraint rods.*

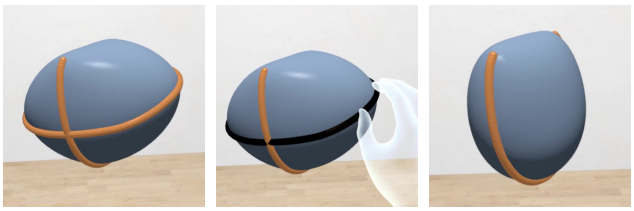


Figure 9: *Removing a constraint rod.*

lus SDK and `libigl` [JPO17], and runs on Windows. Mesh processing operations and linear system solvers come from `libigl`. The rendering and mesh operations run on two separate threads, so that rendering continues during costlier operations like triangulation. Curve deformation and surface optimization run at interactive rates on the examples shown here (3000 to 4000 vertices). Some slowdown is noticeable as more rods are added, which is due to the fact that the rods are rendered as chains of actual cylinder and sphere meshes. This slowdown could be alleviated by implementing a cylinder/sphere shader to give the illusion of volumetric rods instead. Curve deformation does not show any significant slowdown, since the linear systems only need to be factorized after operations that change the mesh topology.

Figure 11 shows eight examples of models created with RodMesh by novice users with no artistic design experience. The gummi bear model took approximately 15 minutes to create and consists of the main body, four extrusions for the legs and arms and an extrusion for the head. Two extra open control rods were added on the head to create the effect of a nose and to provide extra control over the shape of the ears. The fantasy animal in the same figure consists of the main body with an extra control rod to create a waist, an extrusion for the head, which in turn has two extrusions for the ears, plus four extrusions for the feet and one for the tail. The extra control rod in the middle of the body of the animal gives the effect of a transition from upper to lower body. Modeling this fantasy animal took approximately 7 minutes, and the modeling process is shown in the accompanying video. The airplane in Figure 11 was made in an 8 minute modeling session. The initial mesh was stretched to form the body of the airplane. Three extrusions were added to create the wings and rudder of the plane. In order to make these parts thinner, the extrusions have been deflated using bimanual deformation on the looped control rods that are wrapped around them. The flower model was made by adding five extrusions to the flower heart as petals, and deforming the control rods of each of the petals into the desired shape. The modeling time of 4 minutes shows that our application can be used to very quickly create model prototypes. Creating the bird model took approximately 8 minutes and consisted of a series of extrusions and rod deformations. The boat and mushroom models took between 1 and 2 minutes to complete. Both use a cut surface as a flat base for an extrusion (the sail and the mushroom stem). The lamp was incrementally designed starting from the mushroom model, cutting the top to be flat and adding control rods to define the shape of the lamp shade. The base of the lamp was deformed to resemble an ellipse shape.

Figure 10 shows a comparison between 3 models created with FiberMesh and RodMesh. The models created with the original non-VR application and our adapted two-handed VR version show high correspondence. Creation times for the models using RodMesh were significantly shorter than with FiberMesh. The models created with RodMesh have somewhat less smooth curves in some locations due to a decreased rod resolution, which is necessary for the interactive rates required for VR.

5. Discussion, Limitations and Future Work

In this work we attempt to intelligently port a powerful 3D modeling paradigm developed for a keyboard, mouse and 2D display setup into immersive VR. In 2D, we can rather precisely sketch and manipulate objects by clicking and dragging the mouse or stylus when using the dominant hand, but two-handed input in this setting suffers from asymmetry because the non-dominant hand is less precise. Conversely, in 3D VR, sketching in the air is tiring and imprecise, all the more so with the non-dominant hand. Our idea to use rods as control primitives enables a more symmetric and natural usage of both hands, bridging the dexterity gap. We no longer require accurate input, but rather the more coarse "holding and moving" of the held rod parts, which is what people normally

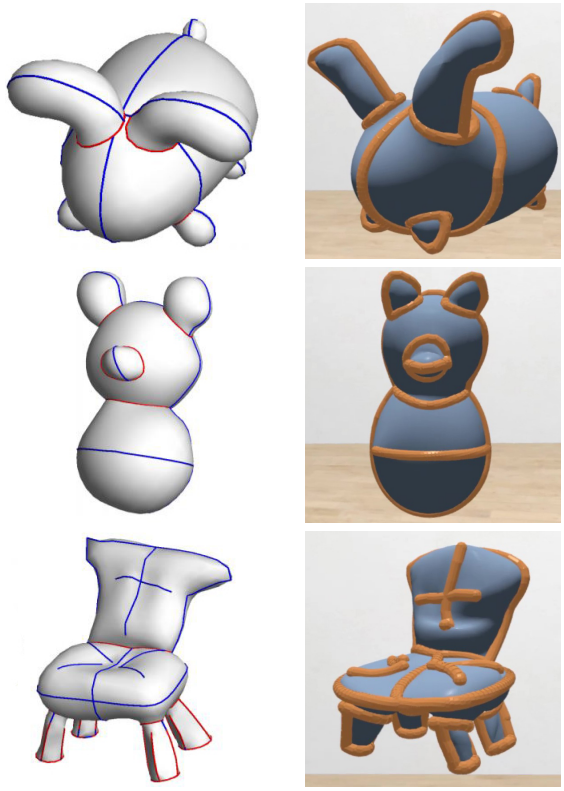


Figure 10: *Comparison of models created with FiberMesh [NISA07] (left) and RodMesh (right). Creation times from top to bottom: for FiberMesh: 10 min., 10 min., 20 min.; for RodMesh: 7 min., 6 min., 12 min.*

do in daily life using two hands. We delegate determining the geometry of the full rod to the curve deformation algorithm and gain the rather efficient, economical and intuitive two-handed manipulation.

The participants of our informal user study pointed out some limitations and suggestions for future improvements. One recurring remark was the desire to add some physical characteristics to the rods, such as limiting the maximal possible stretch, or providing some kind of haptic feedback to simulate the feeling of exerting an increasing amount of power to bend the rod. In future work, we would also like to explore a deformation model for the rods that takes thickness into account and allows twisting along the rod; the Frenet frames along the rod could then be incorporated and inform the surface optimization.

Some users commented that the current restriction to deform one rod at the time was limiting their actions. An example they gave was not being able to quickly bend connecting control rods towards each other (because their hierarchical attachment propagates the movement of the parent rod onto the child rod, which then moves in a similar manner as the parent). In our current implementation this restriction is present because of the hierarchical sequential processing of the deformations of the individual rods. A future implementation might explore solutions that avoid this sequential processing.

Although users found the laser beam easy to work with for the cutting and rod addition tools, they experienced more challenges when using it for the extrusion tool. Drawing a closed curve on the mesh surface requires more precision than drawing an open curve over the mesh surface in a sweeping motion. In addition, participants had difficulties sensing depth when defining the extrusion silhouette. One possible solution could be to allow the user to create the extrusion separately from the main model using the same mechanisms for rod based surface creation and editing, and then attach the part at the desired location on the main model.

Finally, our two-handed rod-based interaction technique could also be applied to other methods of surface definition and optimization. For example the current application could be adapted to use a thin film surface optimization, or to define outlines of Coons patches.

6. Conclusion

In this paper we presented RodMesh, an application that makes 3D modeling easier and more accessible for novice users by utilizing the advantages of immersive 3D VR. The usage of both hands for deformation of control structures allows users to complete various modeling tasks. Compared to traditional mouse and 2D display approaches, our application allows users to generate more interesting spawned surfaces and provides a faster method for creating extrusions. Compared to previous 3D modeling tools for VR, our application requires less (precise) user input and uses two-handed input for an interaction technique that closer resembles modeling techniques in real life. We hope that this work motivates further research into adapting existing 3D modeling paradigms for two-handed 6 DOF input and virtual reality.

7. Acknowledgements

This work was supported in part by gifts from Snap, Inc. and Facebook. We would like to thank the participants of our user study for their useful suggestions, the anonymous reviewers for their valuable feedback, Kaan Baki for helping with the video creation and Alexandra Ion and Katja Wolff for their insightful feedback on drafts of this paper.

References

[3ds19] 3DS MAX®: Autodesk Inc., 2019. URL: https://www.autodesk.com/products/3ds-max/. 2

[Aut19] AUTOCAD®: Autodesk Inc., 2019. URL: https://www.autodesk.com/products/autocad/. 2

[BIAI17] BESANÇON L., ISSARTEL P., AMMI M., ISENBERG T.: Mouse, tactile, and tangible input for 3d manipulation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2017), CHI '17, ACM, pp. 4727–4740. 2

[Ble19] BLENDER®: Blender Foundation, 2019. https://www.blender.org. 2

[BM86] BUXTON W., MYERS B.: A study in two-handed input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1986), CHI '86, ACM, pp. 321–326. 2

Figure 11: *Some examples of models created with RodMesh. Creation times from left to right, top to bottom: 15 min. (teddy), 7 min. (animal), 8 min. (airplane), 4 min. (flower), 8 min. (bird), 2 min. (boat), 2 min. (mushroom), 6 min. (lamp).*

[DACJ12] DE ARAÙJO B. R., CASIEZ G., JORGE J. A.: Mockup builder: Direct 3d modeling on and above the surface in a continuous interaction space. In *Proceedings of Graphics Interface 2012* (Toronto, Ont., Canada, Canada, 2012), GI '12, Canadian Information Processing Society, pp. 173–180. 2

[GBK*01] GROSSMAN T., BALAKRISHNAN R., KURTENBACH G., FITZMAURICE G., KHAN A., BUXTON B.: Interaction techniques for 3D modeling on large displays. In *Proc. 2001 Symp. Interact. 3D Graph.* (New York, NY, USA, 2001), I3D '01, ACM, pp. 17–23. 2

[GBS03] GROSSMAN T., BALAKRISHNAN R., SINGH K.: An interface for creating and manipulating curves using a high degree-of-freedom curve input device. In *Proc. SIGCHI Conf. Human Factors in Computing Systems* (New York, NY, USA, 2003), CHI '03, ACM, pp. 185–192. 2

[GMP*16] GLAUSER O., MA W.-C., PANOZZO D., JACOBSON A., HILLIGES O., SORKINE-HORNUNG O.: Rig animation with a tangible and modular input device. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 35*, 4 (2016), 144:1–144:11. 2

[Goo19] GOOGLE BLOCKS: Google, 2019. https://vr.google.com/blocks/. 3

[Gra18] GRAVITY SKETCH: Gravity Sketch, 2018. https://www.gravitysketch.com/app/. 3

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3 (July 2005), 1134–1141. 2

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A Sketching Interface for 3D Freeform Design. In *Proc. 26th Annu. Conf. Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 409–416. 1, 3

[JPO17] JACOBSON A., PANOZZO D., OTHERS: libigl: A simple C++ geometry processing library, 2017. http://libigl.github.io/libigl/. 7

[KKSH15] KANG Y., KIM H., SUZUKI H., HAN S.: Editing 3d models on smart devices. *Computer-Aided Design 59* (2015), 229 – 238. 2

[LKG*03] LLAMAS I., KIM B., GARGUS J., ROSSIGNAC J., SHAW C.: Twister: a space-warp operator for the two-handed editing of 3d shapes. *ACM Trans. Graph. 22* (2003), 663–668. 2

[LPL*18] LI C., PAN H., LIU Y., TONG X., SHEFFER A., WANG W.: Robust flow-guided neural prediction for sketch-based freeform surface modeling. *ACM Transactions on Graphics 37*, 6 (dec 2018), 238:1–238:12. 3

[May19] MAYA®: Autodesk Inc., 2019. https://www.autodesk.com/products/maya/. 2

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: FiberMesh: Designing Freeform Surfaces with 3D Curves. *ACM Transactions on Graphics 26*, 3 (jul 2007), 41:1–41:9. 1, 2, 3, 4, 5, 6, 8

[Ocu19] OCULUS MEDIUM: Oculus, 2019. https://www.oculus.com/medium/. 3

[PIA10] PERKUNDER H., ISRAEL J. H., ALEXA M.: Shape modeling with sketched feature lines in immersive 3d environments. In *Proc. 7th Sketch-Based Interfaces and Modeling Symp.* (Aire-la-Ville, Switzerland, Switzerland, 2010), SBIM '10, Eurographics Association, pp. 127–134. 3

[PWH16] PALLEIS H., WAGNER J., HUSSMANN H.: Novel indirect touch input techniques applied to finger-forming 3d models. In *Proceedings of the International Working Conference on Advanced Visual Interfaces* (New York, NY, USA, 2016), AVI '16, ACM, pp. 228–235. 2

[RRS19] ROSALES E., RODRIGUEZ J., SHEFFER A.: Surfacebrush: From virtual reality drawings to manifold surfaces. *ACM Trans. Graph. 38*, 4 (July 2019), 96:1–96:15. URL: http://doi.acm.org/10.1145/3306346.3322970, doi:10.1145/3306346.3322970. 3

[SAG*13] SHTOF A., AGATHOS A., GINGOLD Y., SHAMIR A., COHEN-OR D.: Geosemantic snapping for sketch-based modeling. *Computer Graphics Forum 32*, 2pt2 (may 2013), 245–253. 3

[SMSH18] SOLER C., MARTIN T., SORKINE-HORNUNG O.: Cosserat rods with projective dynamics. *Computer Graphics Forum (proceedings of SCA issue) 37*, 8 (2018), 137–147. 4

[SWSJ06] SCHMIDT R. M., WYVILL B., SOUSA M. C., JORGE J. A.: Shapeshop: Sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), SIGGRAPH '06, ACM. 3

[Til19] TILT BRUSH: Google, 2019. https://www.tiltbrush.com/. 3