3D sketching in immersive environments: Shape from disordered ribbon strokes

Georges-Pierre Bonneau, Stefanie Hahmann

Author's Version to appear in Author's Version / Computers & Graphics

3D Sketching in Immersive Environments: Shape from Disordered Ribbon Strokes

Georges-Pierre Bonneau*, Stefanie Hahmann

Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France

ARTICLE INFO

Article history:

Keywords: 3D sketching, shape modeling, immersive design, Interactive techniques, surface reconstruction, sketch based modeling

ABSTRACT

Immersive environments with head mounted displays (HMD) and hand-held controllers, either in Virtual or Augmented Reality (VR/AR), offer new possibilities for the creation of artistic 3D content. Some of them are exploited by mid-air drawing applications: the user's hand trajectory generates a set of stylized curves or ribbons in space, giving the impression of painting or drawing in 3D. We propose a method to extend this approach to the sketching of surfaces with a VR controller. The idea is to favor shape exploration, offering a tool, where the user creates a surface just by painting ribbons. These ribbons are not constrained to form patch boundaries for example or to completely cover the shape. They can be very sparse, disordered, overlap or not, intersect or not. The shape is computed simultaneously, starting with the first piece of ribbon drawn by the user and continuing to evolve in real-time as long as the user continues sketching. Our method involves minimizing an energy function based on the projections of the ribbon strokes on a proxy surface by taking the controller's orientations into account. The current implementation considers elevation surfaces. In addition to many examples, we evaluate the time performance of the dynamic shape modeling with respect to an increasing number of input ribbon strokes. Finally, we present images of an artistic creation that combines stylized curve drawings in VR with our surface sketching tool created by a professional artist.

Author's Version Preprint.

1. Introduction

The challenge of making modeling transparent to the user in order to facilitate the creativity of designers and engineers is still relevant today. It is even becoming more important thanks to the democratization of 3D modeling that we have been experiencing for the last decade. The recent trend is to give access to 3D modeling and manufacturing systems (3D printing) to the general public and not only to professional users.

A new approach to facilitate 3D creation is to start from hand-drawn sketches, since everyone can draw, and develop algorithms that automatically transpose a 2D sketch into a virtual 3D object in the computer. Reconstructing 3D shapes from a single view is, however, an ill-posed problem, since a 2D image can represent an infinite number of different 3D surfaces. Using annotated contours, previous work has succeeded in solving this problem in a few specific cases by complementing the minimization of the re-projection error with specific geometric constraints. These constraints include parallelism, orthogonality, developability, exact mirror symmetry or orthogonality of cross sections in the case of industrial design sketches.

The work presented here aims to develop a 3D modeling system that exploits freehand sketching in the air in virtual reality (VR). The increasing accessibility of VR devices to the general public has given rise to a range of new opportunities for interaction with 3D virtual environments. One could easily think of video games or cinematography, but new applications for creat-

*Corresponding author:
e-mail: georges-pierre.bonneau@inria.fr (G.-P. Bonneau)

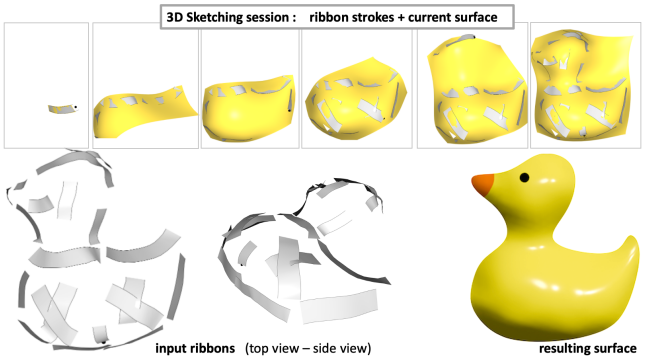

Fig. 1: Interactive exploration of design ideas. During 3D sketching of ribbon strokes (gray) the surface is immediately synthesized (yellow).

ing 3D content are also gaining popular interest. Most of them use a headset for visualization and two joysticks for the shape editing process, but assumptions about the design approach are diverse. The bet of Shapelab [1] and Oculus Medium [2] is to mimic the sculpting process while other systems see the user as a virtual painter. For example, Google's Tilt Brush [3] system allows the user to paint brushstrokes with a specific width, color and texture within the VR environment. This also opens the door to unprecedented styles for generating artistic 3D content, as evidenced by the work of the online platform SketchFab [4]. Most existing 3D shape modeling systems in VR, however, follow the standard shape generation metaphors known from CAD and graphics tablet modeling systems where the user begins by generating a network of 3D curves before filling the cycles with surfaces. However, creativity is often limited by cumbersome and very complex 3D modeling procedures.

The goal of our work is not to transpose existing CAD systems into 3D, but to invent more intuitive and expressive tools to create 3D content. We propose to explore a 3D sketching approach, which consists of creating a smooth surface by mimicking the artist's gesture when painting an object in VR with a brush. Even when an observer can easily recognize the object painted by an artist with brushes that form ribbon-like strokes (ruled surfaces), it is a complex problem to create a surface from these unstructured ribbons [5], especially if the resulting surfaces are intended for further processing for product design or manufacturing.

Such a 3D sketching tool is to be used in the early design stage, also called **ideation phase** [6]. Within this stage, a designer or an artist will generate tens of freehand sketches per hour, most of which are discarded. Similarly, Perkunder et al. and literature cited in [7] point out the importance of sketches in product design serving as an externalization of mental concepts and being part of a reflective process in order to generate new ideas. Once a conceptual design sketch has been selected, final production drawings are carefully constructed for presentation to clients or management, and then passed on to trained modelers who translate the drawings into 3D models using professional modeling tools.

We therefore remainfocused on sketching as a metaphor for modeling which involves quickly outlining or conceptualizing an idea without delving deeply into specific details. In particular, we are enhancing 3D sketching with a new tool that dynamically generates real-time "surface" feedback on the user's gesture. Our tool joins a series of recent 3D sketching tools [8, 5, 6, 9], or shape drafting tools [10, 11], all of which strive to support the user's expressiveness and creativity, rather than being a precise design tool. The quality, the beauty and precision of the resulting shape are not measures of success and are not the primary goal.

Our overall **contribution** is a new VR sketch-based surface modeling tool that enables users to rapidly conceptualize design ideas. The main features are:

- **surface painting.** The user creates a surface by simply painting ribbon-like brushes (ruled surfaces) thus imitating the artist's gesture when painting. The system converts them into a smooth elevation surface.
- **sparse and disordered ribbons.** The input brushes are not constrained to form patch boundaries or to completely cover the shape. They can be very sparse, disordered, overlap or not, intersect or not, see Figures 1 and 3.
- **on-the-fly surface generation**: During 3D ribbon sketching the system dynamically synthesizes a surface to the current set of sketched ribbons. Interactive frame rates give users immediate visual feedback on their sketching gestures.

2. Related works

Our work is related to sketch-based VR modeling interfaces, 3D sketching and 3D curve drawing methods.

Sketch-based VR drawing and modeling systems. The creation and modeling of 3D shapes directly in VR/AR is once again attracting growing interest as new HMD systems become affordable for the general public. Our focus is on VR/AR devices where the user is equipped with two hand-held controllers enabling them to draw and interact in mid-air with the 3D model with immediate feedback, while moving almost freely around the 3D model. VR modeling systems cover a wide range of modeling metaphors ranging from freehand painting with CavePainting, SculptrVR or Tilt Brush [12, 3] and coarse polyhedra modeling [13, 14] for the general public to sophisticated modeling and sculpting tools [15, 1] for expert users. Our work draws on the metaphor of painting to describe a shape by using unorganized ribbon brushes as input and transforming them into a freeform surface.

The more a modeling tool allows the user to control the shape and create details, the more complex the system becomes to use. More control can also mean more constraints, because the user has to create his own imaginary model according to the available tools. A diversity of free-form surface modelers in VR enable to create controlled models but require the user to imagine the shape as a collection of non-overlapping surface

patches, bounded by curves that must be drawn before fitting a surface. This is the case for Lift-Off [8] and JustDrawIt [16] using sweep surfaces, where 2 or 3 curves serve as guidelines, for Nurbs-based modeling and lofting with GravitySketch [17], where control points of curves can be manipulated and for surfacing methods such as FreeDrawer [18]. Surfacing methods such as Cassie [6] allow the user to draw 3D curves free-hand, which then have to be connected by some snapping method before a cycle detection algorithm finds the patch boundaries. Using a proxy surface close to the set of unstructured 3D curves [9] enables to automatically recover sharp edges in the sketch. The surfacing method takes several minutes to compute. Our approach does not constrain the user to draw patch boundaries in form of 3D curves that intersect properly and form cycles in the network to be filled by patches. Our focus is more on translating the hand gesture when describing a shape by painting it with disordered ribbon strokes.

Closer to the spirit of sketching but providing less control, Perkunder et al. [7] and RodMesh [11] transpose Fibermesh [10] to VR. Sketched 3D curves, not necessarily connected, serve to create an initial inflated surface and to edit it then. The resulting shapes are rough and organic looking. While sharing the same sketching spirit as RodMesh and SculpVR, where the focus is on shape ideation, we present a new approach to shape modeling that also allows for rapid sketching and experimentation with 3D shape ideas based on the brush painting metaphor.

Other VR systems allow non-expert users to conceptualize an imaginary shape in a less controlled manner by merging adjacent surface ribbons to a rough surface. Schkolne and Schröder [19] pioneered this drawing paradigm by capturing the hand's gesture moving along an imaginary shape and producing a slender piece of surface mesh which can then incrementally grow. SurfaceBrush [5] builds on this 3D painting methaphor where the user covers the shape with dense adjacent and partially overlapping ribbons which are then post-processed off-line to compute the final surface. Both approaches are particularly suitable for creating organic shapes in a very intuitive way and obviously particularly appreciated by artists [20]. We are in line with both 3D sketching approaches, not only in the use of ribbon strokes to sketch a surface, but also in the type of output surface: rough and loose. With our approach, however, it is neither necessary for the ribbons entered to be dense nor adjacent to each other. The user is free to sketch disordered, unstructured and overlapping ribbon strokes to sketch a surface. Oversketching is also supported. Another equal important feature of our method which contrasts with previous works, is that the user receives **immediate feedback while drawing** ribbons strokes, the surface immediately grows and adapts during sketching.
A book collecting survey papers on 3D sketching was recently published [21].

2D Sketch-based modeling (SBM). SBM infers 3D geometry from 2D sketches. In order to overcome the depth ambiguity some prior knowledge about drawing conventions, application domain (man-made objects, organic shapes, human bodies, garments, architecture, flowers, etc) or geometric/perceptive cues applied either on the sketch or the resulting 3D object are used, e.g., parallelism and scaffolds [22], symmetry [23], cross-sections [24, 25], minimal distorsion and planarity [26], curvature field and orthogonality [27], silhouettes, developability [28], stylized fashion sketches [29]. A comparative study can be found in the survey of Ding and Liu [30]. While our work utilizes mid-air 3D drawing, the sparsity of our input ribbon strokes connects to the spirit of sketching. We take advantage of sketching directly in 3D, so that the geometric and perceptive cues are not required anymore, but keep within the spirit of quickly (roughly) drawing a shape without a lot of details using only a sparse set of strokes, ribbon strokes in our setting.

The survey [31] categorizes 2D SBM approaches in single-view and multi-view. The methods cited above belong all to the first category. Whereas in Fibermesh [10], ILoveSketch [32] and SecondSkin [33], the user interactively adds 2D strokes or modifies strokes when applying frequent view modifications. Here we are sketching directly in 3D and want to take advantage of an immersive interaction environment.

3D curves drawings. The increasing popularity of applications such as Tilt Brush [3] and Quill [2] demonstrates the significant potential of drawing in air and presents a profound opportunity to revolutionize interactive 3D graphics. It is well known and can be observed in all VR drawing publications cited so far that it is very difficult to accurately draw or align curves or ribbons in the air. Arora et al. [34] investigated these observations with user-studies. Another problem occurs when drawing ribbons with a hand-held controller, because the controller's coordinate frame has to be translated into a local ribbon frame. The user may be forced to make uncomfortable rotations with their wrist. Rosales et al. [35] propose an algorithm that solves the problem by adaptively calculating the local frame. To get a grip on imprecision, various tools are used, such as lifting 2D curves from an image into 3D [8], drawing on virtual or physical objects [36], using mobile sensors [37, 38], projecting curves onto the surface [39], snapping of curves in proximity [6], physical guidelines or haptic feedback [40]. We do not fight against inaccuracies, but accept them as part of the design ideation process that forms the framework of our approach.

There are no depth cues in sparse drawings so that the volume of an object is difficult to appreciate. Models created with Tilt Brush or similar tools use ribbons, which are tiny ruled surfaces, instead of curves, because the denser the ribbons the more the drawing resembles to a surface [35]. Our approach is different: we keep in the spirit of sparse and roughly sketching ribbon strokes mimicking the gesture of sliding/painting across a surface.

Mid-air drawing in 3D of large objects or complex scenes quickly becomes confusing, as there are no hidden line or hidden surface representations. Our approach implicitly solves the problems as it translates one-dimensional curves and ribbons in 2-dimensional surfaces. In the result section 7 we show that this feature can be of great importance when it comes to large scenes depicting many objects or characters. We show an artwork produced by a professional artist using our system.

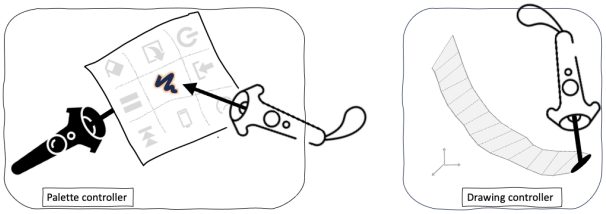

Fig. 2: The minimal interface consists of two handheld controllers serving as palette holder and drawing tool. The palette interface can be displayed by the user to select simple drawing options. The drawing controller is presented to the user with a tiny cylinder-shaped tip drawing the ribbon. Its length is adjustable.

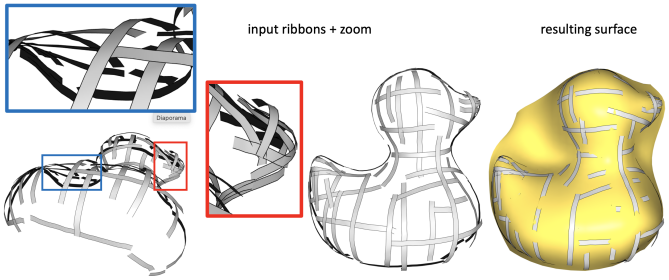

Fig. 3: Disordered ribbon strokes - input to our method. Ribbon strokes do not need to be connected (see zoom). They can be dense or sparse, noisy or smooth.

3. System setup and overview

Our surface modeling framework is designed to be used in a context of drawing in VR. It works in symbiosis with a user-controlled 3D stroke drawing tool in such a way that the surface evolves in real-time with the ribbon-shaped stroke sketching. When the user starts drawing, the surface builds up and evolves until the user stops sketching. In what follows, we briefly describe the context of use, and then focus primarily on the problem of shape modeling: How do you deduce plausible surface geometry from very sparse, disorganized and imprecise 3D ribbon strokes that arrive on the fly?

The immersive environment. It consists of a VR 3D stroke drawing system working with a HMD and 2 hand-held controllers for user-interaction. We use a homemade system based on OpenVR [41] whose user-interface is kept simple and intuitive, similar to what is used in [6]. More sophisticated interfaces can be found in commercial VR applications [3, 2]. Figure 2 shows the interface as implemented for two HTC Vive controllers. One controller serves as drawing-mode selection palette, the other as drawing controller. On the palette, the user can modify the width and color of the ribbon strokes, remove strokes and choose the surface resolution. The drawing controller with six degrees of freedom (6DOF) captures position and orientation in 3D space, which are subsequently translated by the drawing system into vertex positions of a 3D polyline and the ribbon geometry. **Ribbon strokes** are rendered from raw data without any smoothing as ruled surface strips centered on the polyline and defined by sweeping a straight line as ruling with fixed width along the spatial trajectory (Fig. 2). For better user control, we render the end of the controller as a thin cylinder indicating the direction of the current ruling and the width of the ribbon stroke. The controller's orientation will be used in order to define the ribbons rulings directions.

Choice of surface model. Let us formalize the objectives of our surface modeling system.
(1) Painting-metaphor. The idea is to allow the user to create an imagined surface as if painting it with small brushstrokes in ribbon form, without imposing restrictions on how the ribbons should be arranged in relation to each other. The strokes can be particularly disordered, they can intersect, but do not have to, they do not have to be connected, they do not have to form clean patch networks, they can be dense or sparse, noisy or smooth, see Figure 3. There is no limitation in the number or length of the strokes, neither in the kind of arrangement they form.

(2) Ergonomics and precision. Drawing in 3D with a hand-held controller is a difficult task for several reasons. First, it is well known, that strokes drawn in VR are not very precise. For example, the study of [34] reveals that users were not able to draw precise circles in the air with a hand-held controller. We cope with the imprecision of strokes by proposing a least-squares surface approximation model. Second, the space of possible controller orientations is naturally limited by the angular constraints of human body joint movements [42] such as the shoulder, elbow and wrist as the hand moves through space. In addition to these biomechanical limitations, drawing in 3D can also reveal problems of ergonomics. Instead of walking around the model when drawing one can decompose it in several pieces [8]. We adopt the same strategy and propose to draw only what you see in front of you. Such a surface looks like an elevation field. A closed surface can then be composed of several patches.
(3) Real-time interactivity. We also want the surface model to generate a manifold surface on-the-fly with real-time feedback to the user. While the user is drawing, the depicted geometry from the ribbon-strokes should be translated into a plausible surface geometry in real time.

The solution we propose avoids costly processing steps and enables the user to interact directly and immediately with the shape model. It has three steps, which are iterated continuously during the drawing session, each time the user moves the controller at a certain distance. Given a current set of sketched strokes drawn up to this point, we first compute a best fitting plane and mesh a rectangle in this plane, then we parameterize the strokes on this planar mesh and finally deform it to approximate the strokes by minimizing a distance/fairness energy. The energy minimization problem is linear and is solved in a few milliseconds by an iterative solver starting from the previous solution. This allows to render the updated surface and give the user an immediate feedback.

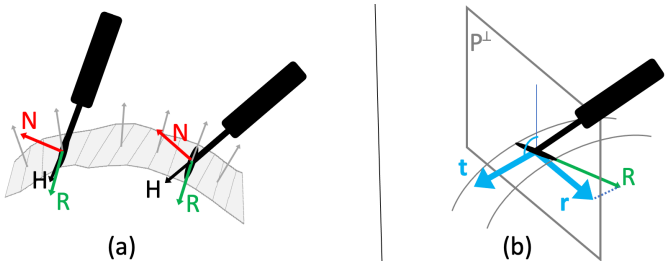

Fig. 4: (a) orthogonal frame $\{H, R, N\}$ of controller. (b) Ribbon tangent vector $\mathbf{t}$ and ruling direction $\mathbf{r}$.

4. Ribbon drawing

The aim is to display 3D ribbons progressively in response to the user's gestures with the controller. From the moment the user starts drawing a ribbon by pressing the trigger, the 3D positions and orientations of the controller are captured sequentially as the user translates the controller in space until the trigger is released. The positions $\mathbf{c}$ are captured in the middle of the cylinder tip and the orientations are captured as an orthogonal frame of unit vectors $\{H, R, N\}$ for Handle, Right, and Normal directions as illustrated in Figure 4(a). When the distance between the last registered point $\mathbf{c}_i$ and the following captured points exceeds a threshold τ, the next point $\mathbf{c}_{i+1}$ and its orientation are registered. We use $\tau = w/2$ to avoid creating too many points that will serve as input to the dynamic surface modeling. Similar to other existing ribbon drawing tools in VR [3, 5], we define a ribbon stroke as a discrete ruled surface represented by a quad strip, where the transverse edges are the rulings. Each ruling can have its own constant width w in our setting. The sequence of points $\mathbf{c}$ defines the ribbon centerline, also denoted ribbon trajectory. The sequence of rulings is calculated progressively during drawing along the centerline from the captured orientations so that they correspond to the user's hand gesture.

The ruling computing algorithm is therefore a key ingredient of a ribbon drawing tool. Following [35], two types of algorithms exist, depending on whether the ruling is computed using a fixed or a flexible mapping from the controller coordinate-frame. The fixed-mapping methods are very fast to compute but may suffer from unexpected situations (flip of normal direction) when the trajectory of the ribbon aligns with a fixed frame direction. In contrary, the flexible AdaptiBrush method [35] is the most performing in terms of ribbon quality, but is more computationally demanding (constraint optimization) even though it is still working at interactive frame rates. In our context, where the type of surface does not require extreme drawing gestures and, more importantly, maximum computational effort must be reserved for continuous surface updating, we implemented a simplified flexible mapping method.

Let us first state, that the normal vector N of the controller cannot be taken as ribbon normal vector. It is obvious from Figure 4(a), that the hand gesture's trajectory can be in contradiction to the controller's normal output.

We compute the ruling $\mathbf{r}$ at point $\mathbf{c}$ from the controller orientation by projecting R onto the plane $P^\perp$ orthogonal to the ribbon trajectory, as illustrated in Figure 4(b). This amounts to rotate R in the plane spanned by $(R, \mathbf{t})$, where $\mathbf{t}$ is the unit tangent of the trajectory. The ruling of unit length is then computed by

$$\mathbf{r} = \frac{\mathbf{t} \times (R \times \mathbf{t})}{(\mathbf{t} \cdot \mathbf{t})}. \tag{1}$$

The trajectory tangents are computed progressively by fitting a quadratic parametric curve to three consecutive points. Assuming that the points $\mathbf{c}_i$ are sampled on a C^1 continuous parametric curve, a quadratic curve is sufficient to estimate the tangent vectors, because we found that the tangent directions do not need to be smoother than the trajectory itself. Let $B(t) = \sum_{i=0}^{2} \mathbf{b}_i B_i^2(t), t \in [0, 1]$ be the quadratic Bézier curve interpolating $\mathbf{c}_{i-1}, \mathbf{c}_i, \mathbf{c}_{i+1}$. Its three control points are uniquely determined from the three interpolation conditions by

$$\mathbf{b}_0 = \mathbf{c}_{i-1}, \quad \mathbf{b}_1 = \frac{\mathbf{c}_i - b_0 B_0^2(\alpha) + b_2 B_2^2(\alpha)}{B_1^2(\alpha)}, \quad \mathbf{b}_2 = \mathbf{c}_{i+1},$$

where

$$\alpha = \frac{\|\mathbf{c}_i - \mathbf{c}_{i_1}\|}{\|\mathbf{c}_i - \mathbf{c}_{i-1}\| + \|\mathbf{c}_i - \mathbf{c}_{i+1}\|}$$

by taking chord length parameterization of the three sample points (all points are distinct). From $B(t)$ the tangent vector $\mathbf{t}_i$ at $\mathbf{c}_i$ can be determined by

$$\mathbf{t}_i = B'(\alpha) = 2(\mathbf{b}_1 - \mathbf{b}_0)(1 - \alpha) + 2(\mathbf{b}_2 - \mathbf{b}_1)\alpha. \tag{2}$$

In case of equidistant parameterization the tangent computation simplifies to $\mathbf{t}_i = \mathbf{c}_{i+1} - \mathbf{c}_{i-1}$. The tangents at the two enpoints are set to $\mathbf{t}_0 = \mathbf{c}_1 - \mathbf{c}_0$ and $\mathbf{t}_n = \mathbf{c}_n - \mathbf{c}_{n-1}$. The normal vectors along the ribbon trajectory are computed by

$$\mathbf{n} = \mathbf{t} \times \mathbf{r}. \tag{3}$$

Using the computed rulings we get two curves in addition to the ribbon trajectory generated from the start and end points of the rulings

$$\begin{aligned} \mathbf{c}^- &= \mathbf{c} - \frac{w}{2}\mathbf{r} \\ \mathbf{c}^+ &= \mathbf{c} + \frac{w}{2}\mathbf{r}. \end{aligned} \tag{4}$$

Remark 1: All rulings (1) are orthogonal to the ribbon trajectory, so that the ribbon has constant width w. Only in the case where the user slides the controller side wise, i.e. in the direction of the tip's cylinder axis, R_i becomes parallel to the tangent $\mathbf{t}_i$ so that $\mathbf{r}_i = 0$. In order to decide about parallelism, we set the threshold to 5 degrees. In any case, having some rulings of zero length, does only affect the visualization of the ribbons, which become a single line there. It does not prevent our surface modeling method from working properly, because the only data required are the captured ribbon curves $\mathbf{c}, \mathbf{c}^-, \mathbf{c}^+$ and not on the ribbon as a surface.

Remark 2: The shape of the ribbons is independent of the two other frame directions H_i and N_i resp., i.e. it is independent on how the user holds the controller with respect to the upward and sideward direction resp.

5. Dynamic surface modeling from sparse ribbons

We design the shape modeling algorithm with the specific objective of seamless integration with a 3D sketching tool to rapidly sketch draft surfaces. Two main challenges have to be faced: The first challenge is to ensure that the surface dynamically evolves alongside the ribbon-shaped stroke sketches in real-time. The ribbon-drawing gesture, which acts like a paintbrush, must have an immediate, visible effect on surface modification. The second challenge, is to infer a manifold variety (surface geometry) from the sparse one-dimensional disconnected and disordered 3D ribbon strokes. Unlike [5], where the ribbons are closely aligned side by side and thus directly suggest a surface geometry, we do not have such useful neighborhood information between the ribbons. As we have seen in Figure 3 the strokes can overlap, but they do not need to intersect. Nevertheless, as the ribbons are intended to represent an imagined shape, it should be possible to map them locally onto a piece of this surface.

Our idea, is to take advantage of a mapping on an intermediate surface in order to establish neighborhood relationships between ribbon strokes, and use is to fit a smooth surface to the ribbon points by minimizing an energy functional. The current implementation works with a moving planar surface for the projection step which dynamically adapts to the set of input strokes, we call it the *proxy plane*.

5.1. Our algorithm

The surface is computed progressively one frame at a time. At each frame, the input consists of the ribbon strokes drawn to date, each given by the boundary curves $\mathbf{c}^-, \mathbf{c}^+$ (4) and the ribbon normal vectors (3). The current **input data** consists thus of a set of 3D curve points denoted $\mathcal{P} = \{\mathbf{p}_l\}_{l=1}^m$ (two points for each ruling) and a set of associated normal vectors $\mathcal{N} = \{\mathbf{n}\}$ (one normal per ruling). At each iteration two main steps are performed:

1. Compute the current proxy plane, project each input point on it, find the triangle in the proxy plane containing this projection and compute its barycentric coordinates with respect to this triangle. This step is not visible to the user but essential for inducing a 2D-manifold variety from the sparse disordered 3D strokes.
2. Compute and display the current surface mesh by minimizing an energy combining an approximation term and a smoothing term. This step is the visible part providing immediate feedback to the user while drawing.

We start our algorithm with $m = 1$ and iterate for each new couple of points as long as the user continues to draw ribbon strokes depicting the same surface. The resulting surface is an open manifold surface mesh.

5.2. Proxy plane

The role of the proxy plane is to establish and preserve spatial coherency between the drawn strokes. Although invisible to the user, it forms the essence of our method. Each time a ribbon point is added by the user, we fit a plane onto which all points drawn up to that moment can be projected so that only ribbons intersecting on the targeted 3D surface have intersecting projections. Therefore, this plane needs continuous recalculations, which must be done rapidly.

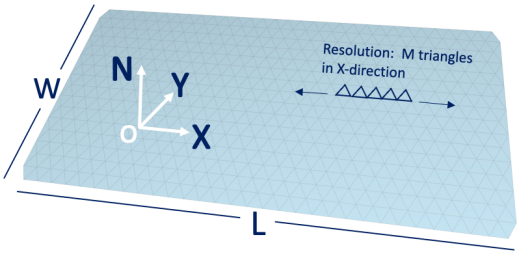

Fig. 5: Proxy plane: local frame, size and triangulation updated per iteration.

The proxy plane is defined as a bounded planar region in 3D space that is triangulated. We use a rectangular boundary in our current implementation, but other boundary shapes are possible. The triangulation is designed to determine the topology of the smooth surface, which in turn can be imagined as a deformed version of the proxy plane. Formally, the proxy plane is defined by

$$\Pi = (\{\mathbf{o}, \mathbf{N}, \mathbf{X}, \mathbf{Y}\}, \mathcal{T}^*, L, W, M)$$

where $\mathbf{N}$ is its unit normal vector, $\mathbf{X}, \mathbf{Y}$ a local orthonormal frame at origin $\mathbf{o} \in R^3$, L, W its size (length and width) in $\mathbf{X}$ and $\mathbf{Y}$ direction, and M an integer controlling the resolution of the triangulation $\mathcal{T}^*$, see Figure 5. The proxy plane Π is updated at each iteration according to the current ribbon input. It must meet the following requirements:

- The orientation and position in space of this plane must enable the projection of all ribbon curves without unwanted intersections.
- The size of the rectangular region must be large enough, so that all points $\mathcal{P}$ can be projected on it.
- Since the local frame defines the orientation of the rectangular plane in 3D space, its constant recalculation must minimize rotation, otherwise it could constantly rotate around its own axis. This would have an effect on the surface that would reduce user comfort.
- Be fast to compute.

Based on these requirements, we compute the origin as the barycenter of all points $\mathbf{o} = \frac{1}{m}\sum_{l=1}^m \mathbf{p}_l$. For computing the orientation of the proxy plane, there are many possibilities. One could for example compute its local frame based on the input points, using a PCA, or a best fitting plane to get the normal vector before searching for the in-plane frame. Both approaches are based on the input points only and do not take ribbon normals into account. They result in badly oriented planes for many simple situations.

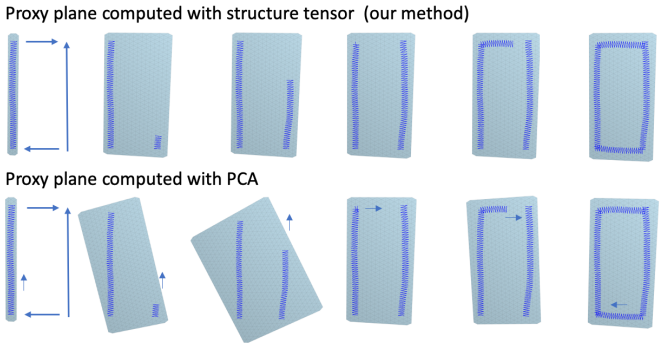

Fig. 6: Automatic adjustment of the proxy plane when 4 ribbons (dark blue) are drawn in the direction of the arrows. The computation of its local frame using the structure tensor (top) is much more stable, than using the PCA (bottom).

The inset shows two ribbons with the same middle curve and different normals. The left ribbon should produce a locally cylindrical shape, while the right ribbon indicates a locally flat surface. We therefore propose to take the ribbon's normals into account by choosing $\mathbf{N} = \sum \mathbf{n}_l / \| \sum \mathbf{n}_l \|$ to define the normal of the proxy plane. Then, we compute the inner local frame $\{\mathbf{X}, \mathbf{Y}\}$ by capturing the anisotropic behavior of the ribbons, i.e. aligning it so that the rectangular region matches the main directions of the ribbon curves, as illustrated in Figure 6 top row. To do this, we take inspiration from the structure tensor used in image processing [43]. The structure tensor is an indicator of the anisotropy of a set of vectors. We compute the structure tensor from the set of derivatives of the projected ribbon's centerlines, as explained now. Given the normal vector $\mathbf{N}$, and an orthonormal frame $\{\mathbf{N}, \mathbf{u}, \mathbf{v}\} \subset R^3$ obtained by the method of Hughes and Moller [44], the ribbon curves are projected into the $\mathbf{u}-\mathbf{v}$-plane. The direction of the discrete derivatives of the projected ribbon curves, expressed in local coordinates (u_l, v_l), are stored in a $2 \times m$ matrix $P' = [\mathbf{p}'_1, \ldots, \mathbf{p}'_m]$. The eigenvectors of the 2×2 structure tensor $[\,P'\,][\,P'\,]^T$ then provide the local coordinates of the three-dimensional orthogonal frame vectors $\mathbf{X}, \mathbf{Y}$. Therefore, $\mathbf{X}$ corresponds to the direction of largest extent of the set of projected ribbons. Figure 6 compares our method and a PCA-based method for the proxy plane update during drawing of four ribbons. The frames computed with the structure tensor in the top row do not rotate as is the case with PCA-based frames.
The length L and width W of Π are set to be 5% larger than the axis-aligned bounding-box of the projected ribbon points. Finally, a regular triangular mesh $\mathcal{T}^*$ of resolution M with almost equilateral triangles is generated, as detailed in Section 6. M is the number of triangles in the X or Y direction, depending on whether L or W is larger. Note, that the remeshing step is only necessary, when the size changes. Again, Figure 6 (top row) illustrates, that remeshing occurred only in the second and fourth plane.

Let $\mathcal{T}^* = (V^*, E^*, F^*)$ be the bounded and simply connected planar triangular mesh of Π, where

$V^* = \{\mathbf{v}_i^*\}_{i=1}^n$ denote the coordinates of the mesh vertices
$E^* = \{e_{kl}^*\}$ the mesh edges with vertex indices (k, l)
$F^* = \{f_{ijk}^*\}$ the mesh triangles with vertex indices (i, j, k).
We associate each curve point $\mathbf{p}$ to its orthogonal projection $\mathbf{p}^*$ onto the planar mesh $\mathcal{T}^*$, which enables us to assign it a unique triplet (α, β, γ) of barycentric coordinates with respect to its surrounding triangle f_{ijk}^*.

$$\mathbf{p}^* = \alpha \mathbf{v}_i^* + \beta \mathbf{v}_j^* + \gamma \mathbf{v}_k^*, \quad 1 = \alpha + \beta + \gamma, \quad \alpha, \beta, \gamma \geq 0. \tag{5}$$

In case a point projects onto a triangle edge or a vertex, the face with the smallest index is chosen. The proxy plane is always large enough to contain all projected points so that the barycentric coordinates are always positive or zero.

5.3. Surface fitting

The proxy plane provides the mesh topology (E^*, F^*) of the final surface and allows to associate each ribbon points to this mesh topology via the barycentric coordinates. These barycentric coordinates are used now to compute the best fitting surface. Our choice of a surface model was guided by the following two conditions: The algorithm must be very fast, so that it can update the shape in response to the user's gesture without lag, and it must be adapted to very sparse data. Therefore, we have chosen to minimize a quadratic weighted objective function composed of a fairness term and weak distance constraints, similar to what is known from least-squares meshes [45] or Laplacian mesh editing [46]. A comparison to alternative methods for surfacing from point clouds is discussed in Section 7.

The surface denoted by $\mathcal{S} = (V, E, F)$ has thus the same topology, i.e., $E = E^*$ and $F = F^*$. The only unknowns are the vertices $V = \{\mathbf{v}_i\}_{i=1}^n$ of $\mathcal{S}$. Finding the optimal surface geometry V is achieved by minimizing the following error functional

$$argmin_{V \in (R^3)^n} \sum_{l=1}^{m} \|\mathbf{d}_l\|^2 + \omega_S \sum_{i=1}^{n} \|\Delta_S \mathbf{v}_i\|^2 \tag{6}$$

where

$$\mathbf{d}_l = (\alpha_l \mathbf{v}_i + \beta_l \mathbf{v}_j + \gamma_l \mathbf{v}_k) - \mathbf{p}_l \tag{7}$$

is the distance term involving the barycentric coordinates (5) of the ribbon points $\mathbf{p}_l$ and

$$\Delta_S \mathbf{v}_i = \frac{1}{2a_i} \sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{v}_j - \mathbf{v}_i) \tag{8}$$

is the discrete Laplace-Beltrami operator for triangle meshes with cotan-weights [47, 48]. a_i is the Voronoï area around $\mathbf{v}_i$, α_{ij}, β_{ij} are the angles formed by the opposite vertices of the edge e_{ij}, and $N(i)$ is the 1-neighborhood of vertex $\mathbf{v}_i$. Solving this quadratic minimization problem results in a sparse linear system of equations. Details about efficient solving will be explained in the following section.

6. Implementation details

Triangulation of proxy plane. At each iteration the proxy plane is updated, including point projection, point location and

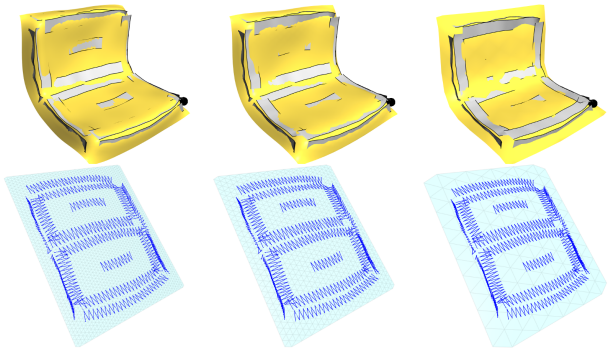

Fig. 7: Mesh resolution from left to right M=50,30,10. Top row: Surface with sketched ribbons. Bottom row: proxy plane with projected points.

triangulation. In order to speed up the updates, we choose a regular tiling of the proxy plane with almost equilateral triangles, so that the point location problem and projection for m points can be solved in $O(m)$ time, independent of the mesh resolution. The mesh resolution M is the number of triangles in the X or Y direction, depending on whether L or W is larger. The number of rows of equilateral triangles in the other direction is then automatically determined. If $L > W$ as in Fig. 5, the mesh consists of M triangles in X direction while the number of rows in Y direction is equal to $N = \lceil d \rceil$ where $d = 2LM/(\sqrt{3}W)$. In case, d is an integer, all triangles are exactly equilateral.
By specifying the maximum resolution in one direction only, we guarantee almost equilateral triangles. Notice, that the remeshing step is only necessary, when M or N changes.

Adaptive resolution. We generally keep the mesh resolution limited to $M = 30$, maximal 900 triangles, in order to guarantee interactive rates during a drawing session. At any moment, the user can display a high-resolution mesh if desired, see Figure 7. Alternatively the mesh resolution M can be varied dynamically by linking it to the frame rate.

Least-square problem-solving. The optimization problem (6) in matrix form writes

$$\|\Omega(A_p V - P)\|^2 + \omega_S \|LV\|^2 \;\rightarrow \min \tag{9}$$

which amounts to solve

$$A^T \Omega A V \;= A^T B, \tag{10}$$

where $V = [\mathbf{v}_1, \ldots, \mathbf{v}_n]^T$ and

$$A = \begin{pmatrix} A_p \\ L \end{pmatrix}_{(m+n)\times n} \quad B = \begin{pmatrix} P \\ 0 \end{pmatrix}_{(m+n)\times 3} \quad \Omega = \begin{pmatrix} I & 0 \\ 0 & \omega_S I \end{pmatrix}_{(m+n)\times(m+n)}$$

$$L_{ij} = \begin{cases} \frac{1}{2a_i}(\cot \alpha_{ij} + \cot \beta_{ij}) & \text{if } j \in N(i) \\ -\sum_{k \in N(i)} L_{ik} & \text{for } i = j \\ 0 & \text{otherwise.} \end{cases}$$

A_p is the sparse $m \times n$ barycentric coordinates matrix such that the l-th row of $A_p V$ is $(\alpha_l \mathbf{v}_i + \beta_l \mathbf{v}_j + \gamma_l \mathbf{v}_k)$, $P = [\mathbf{p}_1, \ldots, \mathbf{p}_m]^T$, and Ω is a weighted diagonal matrix.

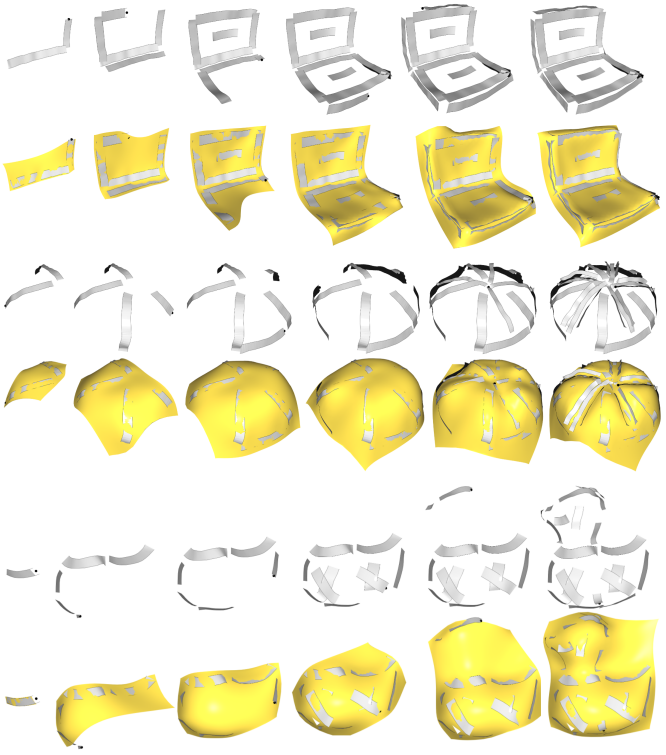

Fig. 8: Drawing session. Surface generation (yellow) starts simultaneously with ribbon drawing and evolves as long as ribbons are added.

We solve the linear system (10) using the Least-Squares Conjugate Gradient solver of Eigen with sparse matrices [49]. This iterative method requires an initial guess V_0. By default, we take the proxy plane mesh as initial guess, i.e. we set V_0 equal to the 3d-coordinates of the proxy plane mesh vertices $\mathbf{v}_i^*$. Solving the system can be significantly accelerated by taking the surface from the previous fitting step as an initial guess, but this is only possible if the mesh topology has not changed. The bath in Fig. 12 is such an example, where the outer border curves have been sketched first, and all following ribbons then always project to the same constant proxy mesh. The execution times for this example are thus much lower, see Fig. 9 or Table 1.

7. Evaluation and results

The system was tested with layman users for all results presented, except one result, for which our system has been made available to a qualified artist. Figure 8 focuses on the dynamic aspect of our algorithm by showing a few intermediate shapes (yellow) and ribbons (gray) drawn up to that point during the design session. The shape updates smoothly following hand gestures (see timings below). In practice, it is as if users were pulling and deforming the surface into the desired shape with their controller. This dynamic behavior is a key novelty of our method in comparison with related works.

Figure 12 shows a gallery of results. The three images per designed model shows the ribbons as drawn by the user

together with the surface rendered with transparency (left), the final surface as it appears at the end of a drawing session (middle) and a final surface "embellished" by the application of a texture and the trimming of a few borders if necessary (right). The number of strokes necessary to realize these models varies between 9 for hat3 and 25 for the bread. More model statistics are given in Table 1. Users typically start by drawing a few curvature lines, as noted already in [27], and add further ribbons in areas where the surface needs to be sculpted more expressively with more curvature variation. Looking at Figure 12, we see in (f) that two crossed ribbons have been added to the top of the hat to get a flat shape, in contrast to (h) where the top is round. In (g), a vertically oriented circular ribbon has been added to get the concave part of the top. It took about 5-10 min to draw each of the models. For all examples we show the original ribbons as captured by the system and displayed to the user without any smoothing. As shown in [34], it is extremely difficult to draw precise curves and ribbons in VR. These examples show that our method, which was developed specifically for such cases, works well with these disordered, very imprecise and particularly sparse ribbon drawings. In all cases, our results accurately reflect the shapes drawn by the user.

Model parameters. There are only two shape parameters, which have no effect on the runtime. The width w of the ribbons is modified with the hand-held controller. Each ribbon has its own width. The shape smoothing parameter ω_S is preset to value 1 and can be changed by the user at any time, since the shape is continuously re-computed. The mesh resolution is set by default to $M = 30$, but automatic adaptation to the frame rate can be switched-on if desired, as explained in Section 6.

Failure cases. The algorithm will converge and produce an output in any case, but will fail to generate a pleasant shape, when the input is not coherent with the underlying elevation surface model. This is also the case, when arbitrary ribbons are drawn in space without representing a surface shape. Then, it happens that multiple ribbon points project into the same proxy triangles, but their distances to the target 3D positions are contradictory. Something similar happens if the drawn surface cannot be projected onto the plane without overlapping, e.g. when drawing a sphere. However, we never observed non-converging least-squares solving, so that a surface is always displayed. Immediate visual feed-back indicates then, that it would be better to remove the last drawn ribbon.

Runtimes and parameters. We evaluated two aspects of our system. First, we evaluated the *dynamic behavior* by computing the time required to update the surface when 2 new ribbon points arrive, i.e. when the number of points increases from m to $m + 2$ (2 points for one new ruling). The result is shown in Figure 9. The absissa is the number m of points. The ordinate corresponds to the computing time of the update with m points including proxy plane fitting, point location and solving of the linear system (10). Overall, all surface update times for the 8 examples of our gallery in Figure 12 stay below

models	m $\lvert\mathcal{P}\rvert$	n $\lvert V\rvert$	$\lvert F\rvert$	strokes	proxy	proj	LS
chair	2128	913	1734	16	7.3ms	46.7ms	115ms
duck	1236	913	1734	20	4.8ms	27.5ms	234ms
bread roll	3266	976	1858	25	8.8ms	69.1ms	185ms
bath	2448	598	1114	20	4.8ms	53.6ms	96ms
umbrella	2808	976	1858	16	9.7ms	61.0ms	169ms
hat 1	2714	913	1734	11	8.0ms	58.9ms	165ms
hat 2	2778	913	1734	13	9.4ms	58.6ms	165ms
hat 3	1550	976	1858	9	5.5ms	33.4ms	155ms

Table 1: Model statistics (number of points, mesh vertices, strokes) and runtimes for each of the examples in Figure 12 using identical mesh resolution $M = 30$ (proxy plane computation, point projection and location, Least-Squares system solving).

0.1s, most of them are much lower, which we consider as "interactive frame rate". Users did not observe any lag when translating the controller with a speed that corresponds to the action of painting on a 3D surface. The plotted curves grow fast at the beginning, where the mesh of the proxy plane is often updated since it is growing. Later, most of the new ribbons serve to improve the design, they project inside the previous proxy plane, so that the number n of mesh vertices stays constant. The plots are almost linear as function of m. The last plotted point on each curve corresponds to the total time required to display the last final surface.
Second, we analyzed the *static behavior* by computing the time required for one individual surface from scratch. Here the linear system (10) has to be solved without a good initial guess, therefore the timings does not correspond the last surface drawn in the dynamic setting. As input, we take the whole set of ribbons and then compute a surface. The timings in Table 1 decompose into the main parts of our algorithm: proxy plane computation, point projection and location, linear system solving. All the timings presented in table 1 are computed for a constant value of $M = 30$. When computing from scratch, system solving always takes more than 50% of time. Proxy plane update is almost negligible. The absolute values in this table are *not* significant for a drawing session, because during sketching, we never compute a surface from scratch, but start the iterations of system solving from the previous solution. The real computation time for the final surface *while drawing* is therefore not the sum of the three entries in Table 1, but the last point plotted on the corresponding curve in Figure 9.

Post process of the final shape. The projected ribbons draw a pattern on the proxy plane. Triangles lying outside a connected region enclosing this pattern might not belong to the final intended shape. Looking for example at the duck in Fig. (12)(b) we see that parts of the resulting shape in bright yellow (middle) are trimmed in the final shape (right), in front and behind the neck especially. It is currently not possible to trim these parts of the mesh at each iteration. The main reason is, that the shape of the trimming curve is not well-defined. Indeed, we have a sparse set of disordered curves projected

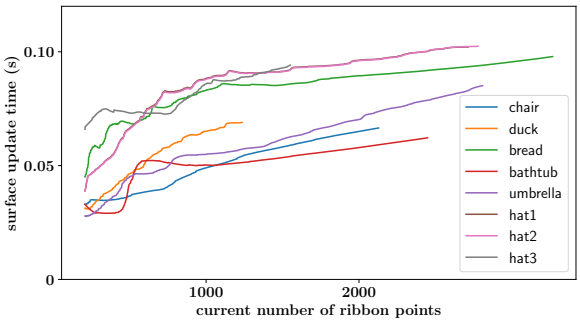

Fig. 9: Evolution of execution time for each iteration of our algorithm including proxy plane computation, triangle location and surface fitting with mesh resolution $M = 30$.

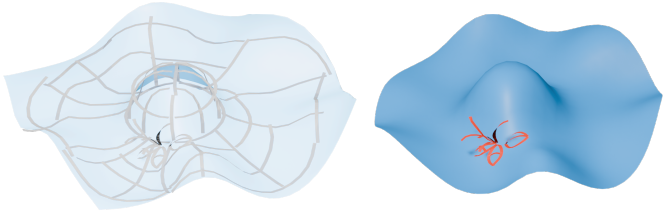

Fig. 10: Cassie hat [6]. Application of our method to raw input data from Cassie, our final surface (left), trimmed surface (right).

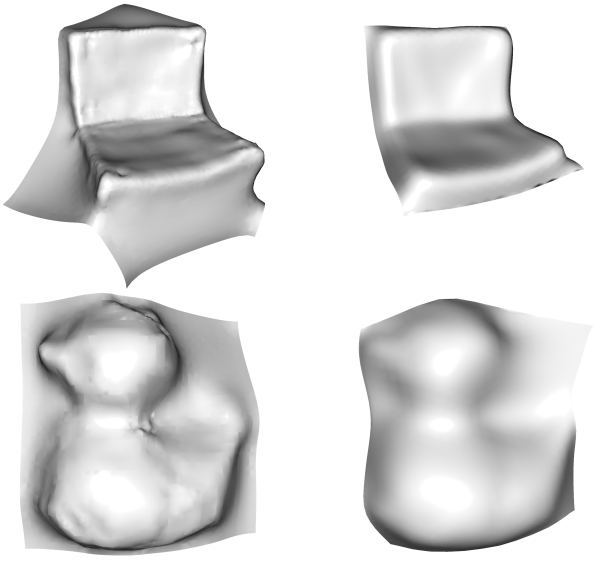

Fig. 11: Comparison to Screened Poisson reconstruction [53], left, ~0.5s reconstruction time for both examples, our solution, ~0.05s average update time while drawing for both examples.

onto the plane, that do not build a closed region. A bounding curve has to be computed, even though what is "inside" and "outside" of this region is ill-defined, and might change when adding new ribbons. One might use a Delaunay-based algorithm, such as α-shapes [50], or an image-based tool, such as the intelligent Lasso tool [51]. The result is a coarse polygonal curve filling more or less naturally the gaps between the ribbons. It is not obvious, if such an automatic trimming corresponds to the user's idea of the final shape. As a future work, an in-depth study is needed on the best way to calculate these trim curves automatically so that they do not affect the running time of our method and meet the users' expectations in terms of style and aesthetics. A user-guided cutting step directly in the immersive environment could also be studied as future work. For the examples shown in Figure 12, we used a manual post process by cutting the mesh using Blender [52].

Comparisons to previous works. None of the existing literature take into account the same type of data as ours. We do neither consider curve networks, patch boundary curves, B-Reps, Nurbs control points. Although we process ribbon strokes as [5] does, we do not require them to be aligned close together in order to cover the whole shape. We process unstructured input as [9] does, but our algorithm is interactive. The type of shapes synthesized in [5, 9] are impressive. They are much more general than ours, since they enable to sketch closed surfaces for example. This versatility requires complex algorithms and finally is overall very time-consuming. None of these methods is suited to interactive sketching with simultaneous surfacing. Our goal from the beginning was to keep in interactive rates, a goal that we obtained by the detriment of the generality of our method. We proposed a new 3D sketching framework as an attempt to free the user from curve drawing constraints. However, we propose the following two comparisons. In Figure 10 we used the same raw input strokes from the teaser example of Cassie [6], that do not form proper curve networks. It shows the resulting shape as output of our method (left) and the "embellished" shape after trimming (right), comparable to the Cassie result. Note however that Cassie does not allow to see the surface evolve in real-time while drawing the ribbon strokes.

In Figure 11 we compare our resulting surface (right) with the Screened Poisson reconstruction method [53] (left). As expected, the result on the left is less smooth. Indeed, surfacing methods from point clouds are generally not well suited for very sparse input from ribbon strokes. The user is referred to [5] where an extensive evaluation of state-of-the-art reconstruction methods applied to ribbon strokes is exposed. The recent VIPSS method [54] better manages sparse input. But the mandatory requirement of interactive frame rates excludes the use of these methods for the surfacing part in our setting and thus justifies our use of a fast, energy minimization approach.

Report on a user experience. We did not carry out a user study about the easiness and intuitiveness of using our system in VR. This was out of the scope of our work. Our main contribution is the modeling system itself, allowing to sketch a surface from ribbon strokes, which constantly evolves in real-time while adding new 3D strokes. We did not innovate on the VR interface for our modeling approach but rather build on a home-made system.

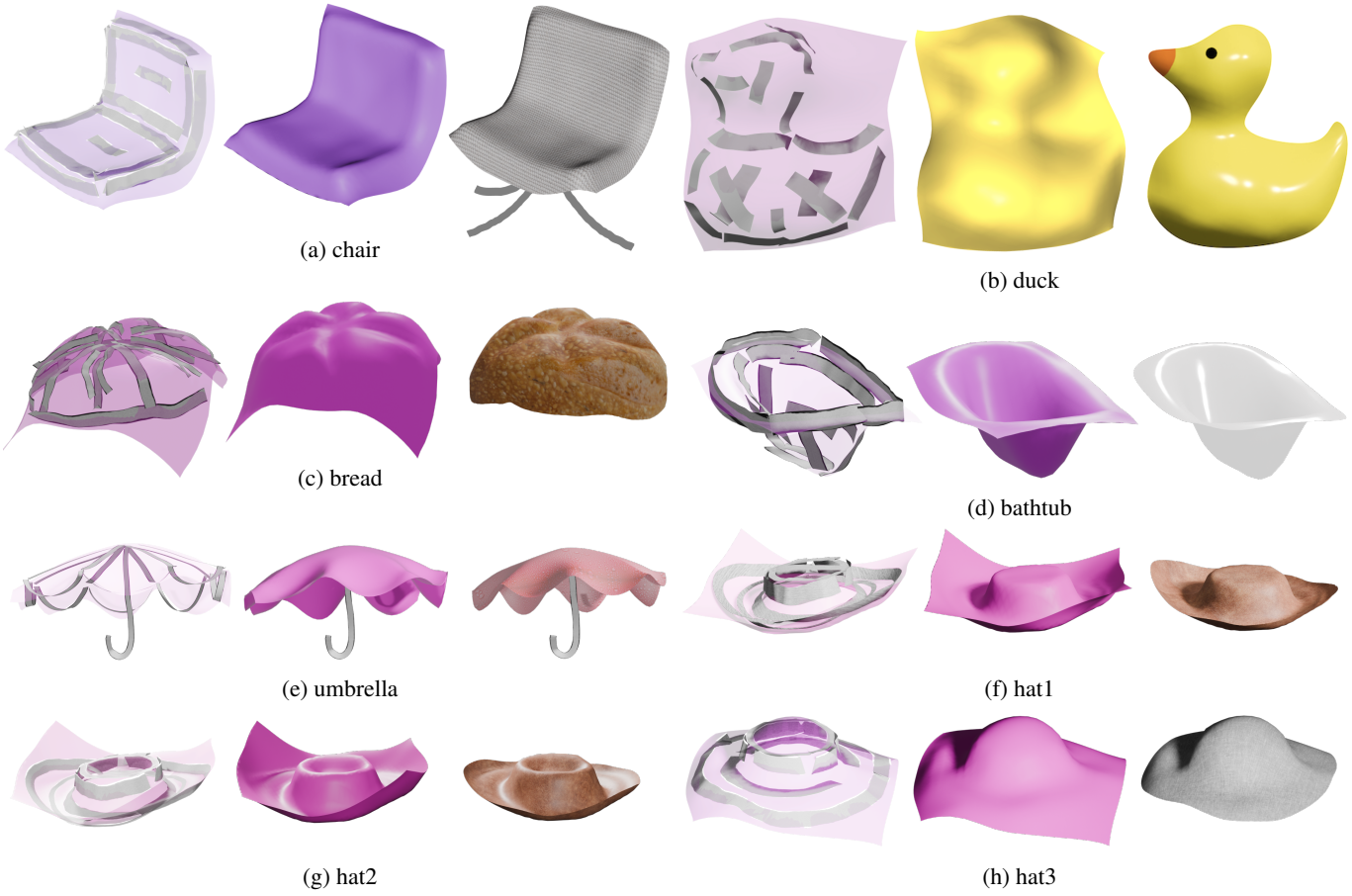

Fig. 12: Designs created with our system. Sparse, disordered ribbon strokes (gray) (left), resulting surface (middle), "beautified" designs using Blender (right).

However, we are proud to report that our surfaces have been put to practical use by visual artist Pauline de Chalendar[1] for several of her recent works in collaboration with the PIRVI platform[2] at the University of Lille. Figure 13 shows as an example two 3D images of the project "L'Ébauchoir" [55], which was presented to a public audience in museums with a tangible 3D exploration experience of a reinterpretation of Jean-Baptiste Carpeaux's sculpture "La Danse" - a fresco created in 1865 for the façade of the Garnier Opera House in Paris (see inset).

8. Conclusion and discussion

In this paper, we have proposed a new interactive technique for modeling shapes. We have shown that dynamic surface modeling in symbiosis with 3D sketches is possible and provides an alternative paradigm for modeling shapes compared to previous work. The algorithm is fast and provides immediate feedback to the user. The user is free to sketch a surface, like a painter using a brush. Our system can deal with arbitrary disordered curves or ribbons input as demonstrated by our results. The most limiting factor is currently the projection of the strokes to a planar proxy surface. Accounting for more complex proxy shapes would enable to extend the present approach to closed surfaces, even though it is still an open question how to decide then what is inside/outside when ribbon strokes are supposed to sketch closed surfaces. Modeling surfaces from completely disordered 3D strokes is an ill-defined problem that requires simplifying assumptions. We therefore believe, that our choice of a proxy plane is a reasonable limitation, since it enabled us to demonstrate that shape modeling from disordered sparse 3D sketching can be solved. Extensions to more complex shapes are to be done next.
In addition, it might be worthwhile to offer more sophisticated post-processing tools in VR, for example an interactive cutting tool like the one we used offline. Combining 3D sketches for coarse-grained ideation with 3D sculpting tools such as Freestyle [56], which can be used to model fine-grained details of the form, is also a possibility for future work.

Acknowledgements

We would like to thank Maxime Isnel, Valentin Guiziou, Hugo Kraft and Paul-Elian Tabarant for their help in implementing the algorithms in VR environment and Samuel Degrande, head of the PIRVI platform of the University of Lille, for integrating our surface algorithm into their VAirDraw system[3] used by artist Pauline de Chalendar.

References

[1] Shapelab. 2021. https://shapelabvr.com.
[2] Oculus Medium. 2019. https://www.oculus.com/experiences/rift/1336762299669605/.
[3] Tilt Brush by Google. 2016. https://www.tiltbrush.com.
[4] Tilt Brush 3D models on Sketchfab. 2019. https://sketchfab.com/tags/tiltbrush.
[5] Rosales, E, Rodriguez, J, Sheffer, A. Surfacebrush: From virtual reality drawings to manifold surfaces. ACM Transaction on Graphics 2019;38(4). doi:https://doi.org/10.1145/3306346.3322970.
[6] Yu, E, Arora, R, Stanko, T, Bærentzen, JA, Singh, K, Bousseau, A. Cassie: Curve and surface sketching in immersive environments. In: ACM Conference on Human Factors in Computing Systems (CHI). 2021,doi:10.1145/3411764.3445158.
[7] Perkunder, H, Israel, JH, Alexa, M. Shape modeling with sketched feature lines in immersive 3d environments. In: Sketch-Based Interfaces and Modeling Symposium. SBIM'10; 2010, p. 127–134.
[8] Jackson, B, Keefe, DF. Lift-off: Using reference imagery and freehand sketching to create 3d models in vr. IEEE Transactions on Visualization and Computer Graphics 2016;22(4):1442–1451. doi:10.1109/TVCG.2016.2518099.
[9] Yu, E, Arora, R, Bærentzen, JA, Singh, K, Bousseau, A. Piecewise-smooth surface fitting onto unstructured 3d sketches. ACM Trans Graph 2022;41(4). doi:10.1145/3528223.3530100.
[10] Nealen, A, Igarashi, T, Sorkine, O, Alexa, M. Fibermesh: Designing freeform surfaces with 3d curves. ACM Trans Graph 2007;26(3):41–es. doi:10.1145/1276377.1276429.
[11] Verhoeven, F, Sorkine-Hornung, O. RodMesh: Two-handed 3D Surface Modeling in Virtual Reality. In: Schulz, HJ, Teschner, M, Wimmer, M, editors. Vision, Modeling and Visualization. The Eurographics Association; 2019,doi:10.2312/vmv.20191312.
[12] Keefe, DF, Feliz, DA, Moscovich, T, Laidlaw, DH, LaViola, JJ. Cavepainting: a fully immersive 3d artistic medium and interactive experience. In: Symposium on Interactive 3D Graphics. I3D'01; 2001, p. 85–93. doi:10.1145/364338.364370.
[13] Jerald, J, Mlyniec, P, Yoganandan, A, Rubin, A, Paullus, D, Solotko, S. Makevr: A 3d world-building interface. In: 2013 IEEE Symposium on 3D User Interfaces (3DUI). 2013, p. 197–198. doi:10.1109/3DUI.2013.6550246.
[14] Blocks by Google. 2017. https://store.steampowered.com/app/533970.
[15] Adobe Medium. 2020. https://www.adobe.com/de/products/medium.html.
[16] Grimm, C, Joshi, P. Just drawit: A 3d sketching system. SBIM '12; Eurographics Association; 2012, p. 121–130.
[17] GravitySketch. 2017. https://www.gravitysketch.com/.
[18] Wesche, G, Seidel, HP. Freedrawer: A free-form sketching system on the responsive workbench. VRST '01; 2001, p. 167–174. doi:10.1145/505008.505041.
[19] Schkolne, S, Pruett, M, Schröder, P. Surface drawing: Creating organic 3d shapes with the hand and tangible tools. In: Conference on Human Factors in Computing Systems. CHI'01; 2001, p. 261–268. doi:10.1145/365024.365114.
[20] Zen, J. Painting in air. SIGGRAPH Comput Graph 2004;38(3):7–9. doi:10.1145/1015999.1016012.
[21] Bonnici, A, Camilleri, KP, editors. Interactive Sketch-based Interfaces and Modelling for Design. River Publishers, Series in Document Engineering; 2023.
[22] Schmidt, R, Khan, A, Singh, K, Kurtenbach, G. Analytic drawing of 3d scaffolds. ACM Trans Graph 2009;28(5):1–10. doi:10.1145/1618452.1618495.
[23] Cordier, F, Seo, H, Melkemi, M, Sapidis, NS. Inferring mirror symmetric 3d shapes from sketches. Computer-Aided Design 2013;45(2):301–311. doi:https://doi.org/10.1016/j.cad.2012.10.013; solid and Physical Modeling 2012.
[24] Shao, C, Bousseau, A, Sheffer, A, Singh, K. Crossshade: Shading concept sketches using cross-section curves. ACM Trans Graph 2012;31(4). doi:10.1145/2185520.2185541.
[25] Zou, M, Holloway, M, Carr, N, Ju, T. Topology-constrained surface reconstruction from cross-sections. ACM Trans Graph 2015;34(4). doi:10.1145/2766976.
[26] Xu, B, Chang, W, Sheffer, A, Bousseau, A, McCrae, J, Singh, K. True2Form: 3D curve networks from 2D sketches via selective regularization. ACM Transactions on Graphics 2014;33(4). doi:10.1145/2601097.2601128.
[27] Iarussi, E, Bommes, D, Bousseau, A. Bendfields: Regularized curvature fields from rough concept sketches. ACM Trans Graph 2015;34(3). doi:10.1145/2710026.

[1] https://www.facebook.com/paulinedechalendar/
[2] https://pirvi.univ-lille.fr
[3] https://pirvi.univ-lille.fr/projects/VAirDraw/

Fig. 13: 3D image realized using virtual reality by Pauline de Chalendar for the artwork "L'Ébauchoir". It offers a tangible 3D exploration experience of a reinterpretation of Jean-Baptiste Carpeaux's sculpture "La Danse" - a fresco created in 1865 for the façade of the Garnier Opera House in Paris (see inset). The shaded surfaces (blue, transparent green and pale yellow) are designed with our algorithm.

[28] Fondevilla, A, Bousseau, A, Rohmer, D, Hahmann, S, Cani, MP. Patterns from photograph: Reverse-engineering developable products. Computers & Graphics 2017;66:4–13. doi:https://doi.org/10.1016/j.cag.2017.05.017; shape Modeling International 2017.

[29] Fondevilla, A, Rohmer, D, Hahmann, S, Bousseau, A, Cani, M. Fashion Transfer: Dressing 3D Characters from Stylized Fashion Sketches. Computer Graphics Forum 2021;40(6):466–483. doi:https://doi.org/10.1111/cgf.14390.

[30] Ding, C, Liu, L. A survey of sketch based modeling systems. Front Comput Sci 2016;10(6):985–999. doi:10.1007/s11704-016-5422-9.

[31] Kazmi, IK, You, L, Zhang, JJ. A survey of sketch based modeling systems. In: Int. Conference on Computer Graphics, Imaging and Visualization. 2014, p. 27–36. doi:10.1109/CGiV.2014.27.

[32] Bae, SH, Balakrishnan, R, Singh, K. Ilovesketch: As-natural-as-possible sketching system for creating 3d curve models. In: Symposium on User Interface Software and Technology. UIST '08; 2008, p. 151–160. doi:10.1145/1449715.1449740.

[33] De Paoli, C, Singh, K. Secondskin: Sketch-based construction of layered 3d models. ACM Trans Graph 2015;34(4). doi:10.1145/2766948.

[34] Arora, R, Kazi, RH, Anderson, F, Grossman, T, Singh, K, Fitzmaurice, G. Experimental evaluation of sketching on surfaces in vr. In: Conference on Human Factors in Computing Systems. CHI'17; 2017, p. 5643–5654. doi:10.1145/3025453.3025474.

[35] Rosales, E, Araújo, C, Rodriguez, J, Vining, N, Yoon, D, Sheffer, A. Adaptibrush: Adaptive general and predictable vr ribbon brush. ACM Trans Graph 2021;40(6). doi:10.1145/3478513.3480511.

[36] Wacker, P, Wagner, A, Voelker, S, Borchers, J. Physical guides: An analysis of 3d sketching performance on physical objects in augmented reality. In: Symposium on Spatial User Interaction. SUI'18; 2018, p. 25–35. doi:10.1145/3267782.3267788.

[37] Kwan, KC, Fu, H. Mobi3dsketch: 3d sketching in mobile ar. In: Conference on Human Factors in Computing Systems. CHI '19; Association for Computing Machinery; 2019, p. 1–11. doi:10.1145/3290605.3300406.

[38] Stanko, T, Hahmann, S, Bonneau, GP, Saguin-Sprynski, N. Surfacing curve networks with normal control. Computers & Graphics 2016;60:1 – 8. doi:https://doi.org/10.1016/j.cag.2016.07.001.

[39] Arora, R, Singh, K. Mid-air drawing of curves on 3d surfaces in virtual reality. ACM Trans Graph 2021;40(3). doi:10.1145/3459090.

[40] Keefe, D, Zeleznik, R, Laidlaw, D. Drawing on air: Input techniques for controlled 3d line illustration. IEEE transactions on visualization and computer graphics 2007;13:1067–80. doi:10.1109/TVCG.2007.1038.

[41] OpenVR. 2024. https://github.com/ValveSoftware/openvr.

[42] Panero, J, Zelnik, M. Human dimension & interior space : a source book of design reference standards. Whitney Library of Design; 1979.

[43] Vergne, R, Barla, P, Bonneau, GP, Fleming, R. Flow-Guided Warping for Image-Based Shape Manipulation. ACM Transactions on Graphics 2016;34(4):Article No. 93. doi:10.1145/2897824.2925937.

[44] Hughes, JF, Moller, T. Building an orthonormal basis from a unit vector. Journal of Graphics Tools 1999;4(4):33–35. doi:10.1080/10867651.1999.10487513.

[45] Sorkine, O, Cohen-Or, D. Least-squares meshes. In: Proceedings Shape Modeling Applications. 2004, p. 191–199. doi:10.1109/SMI.2004.1314506.

[46] Botsch, M, Kobbelt, L. An intuitive framework for real-time freeform modeling. ACM Trans Graph 2004;23(3):630–634. doi:10.1145/1186562.1015772.

[47] Pinkall, U, Polthier, K. Computing discrete minimal surfaces and their conjugates. Experimental Mathematics 1993;2:15–36. doi:10.1080/10586458.1993.10504266.

[48] Desbrun, M, Meyer, M, Schröder, P, Barr, AH. Implicit fairing of irregular meshes using diffusion and curvature flow. SIGGRAPH'99; 1999, p. 317–324. doi:10.1145/311535.311576.

[49] Guennebaud, G, Jacob, B, et al. Eigen. http://eigen.tuxfamily.org; 2010.

[50] Edelsbrunner, H, Kirkpatrick, D, Seidel, R. On the shape of a set of points in the plane. IEEE Transactions on Information Theory 1983;29(4):551–559. doi:10.1109/TIT.1983.1056714.

[51] Mortensen, EN, Barrett, WA. Intelligent scissors for image composition. In: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '95; New York, NY, USA: Association for Computing Machinery. ISBN 0897917014; 1995, p. 191–198. URL: https://doi.org/10.1145/218380.218442. doi:10.1145/218380.218442.

[52] Blender. 2020. https://www.blender.org.

[53] Kazhdan, M, Hoppe, H. Screened poisson surface reconstruction. ACM Trans Graph 2013;32(3). doi:10.1145/2487228.2487237.

[54] Huang, Z, Carr, N, Ju, T. Variational implicit point set surfaces. ACM Trans Graph 2019;38(4). doi:10.1145/3306346.3322994.

[55] de Chalendar, P. L'Ébauchoir. 2020. https://www.atelier-arts-sciences.eu/lebauchoir/.
[56] Stanculescu, L, Chaine, R, Cani, MP. Freestyle: Sculpting meshes with self-adaptive topology. Computers & Graphics 2011;35(3):614–622. doi:https://doi.org/10.1016/j.cag.2011.03.033; shape Modeling International (SMI).