VRSurf: Surface Creation from Sparse, Unoriented 3D Strokes

Anandhu Sureshkumar, Amal Dev Parakkat, Georges-Pierre Bonneau, Stefanie Hahmann, Marie-Paule Cani

VRSurf: Surface Creation from Sparse, Unoriented 3D Strokes

Anandhu Sureshkumar[1], Amal Dev Parakkat[1], Georges-Pierre Bonneau[2], Stefanie Hahmann[2], Marie-Paule Cani[3]

[1]LTCI-Telecom Paris, Institut Polytechnique de Paris
[2] Univ. Grenoble Alpes, CNRS, INRIA, Grenoble INP, LJK
[3] LIX - Ecole Polytechnique / CNRS, Institut Polytechnique de Paris

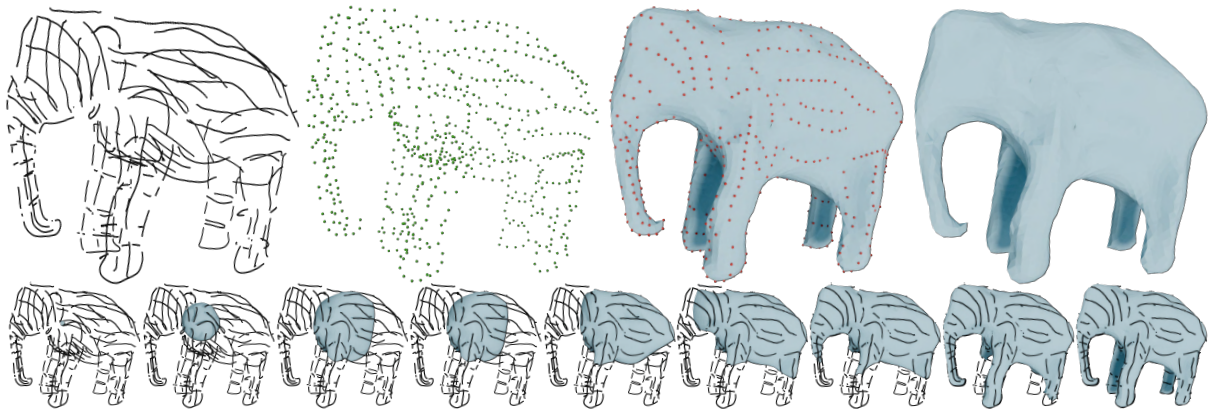


Figure 1: *Top row (left to right): A sparse set of unoriented 3D strokes, sampled version of the strokes used as input to our method, and VRSurf result, a closed manifold surface that interpolates through the samples. Second row: Different stages of VRSurf's balloon inflation process, starting from an initial user-given seed point.*

Abstract

Although intuitive, sketching a closed 3D shape directly in an immersive environment results in an unordered set of arbitrary strokes, which can be difficult to assemble into a closed surface. We tackle this challenge by introducing VRSurf, a surfacing method inspired by a balloon inflation metaphor: Seeded in the sparse scaffold formed by the strokes, a smooth, closed surface is inflated to progressively interpolate the input strokes, sampled into lists of points. These are treated in a divide-and-conquer manner, which allows for automatically triggering some additional balloon inflation followed by fusion if the current inflation stops due to a detected concavity. While the input strokes are intended to belong to the same smooth 3D shape, our method is robust to coarse VR input and does not require strokes to be aligned. We simply avoid intersecting strokes that might give an inconsistent surface position due to the roughness of the VR drawing. Moreover, no additional topological information is required, and all the user needs to do is specify the initial seeding location for the first balloon. The results show that VRsurf can efficiently generate smooth surfaces that interpolate sparse sets of unoriented strokes. Validation includes a side-by-side comparison with other reconstruction methods on the same input VR sketch. We also check that our solution matches the user's intent by applying it to strokes that were sketched on an existing 3D shape and comparing what we get to the original one.

CCS Concepts

• Computing methodologies → Shape modeling; • Human-centered computing → Virtual reality; • Applied computing → Arts and humanities;

1. Introduction

Although being the main building block of virtual environments used in many domains, from entertainment to medicine, architecture, or product design, 3D shapes are much more difficult to create than their 2D counterparts. In particular, they are often out of reach for novice users since 3D design requires not only modeling skills

but also expertise in handling complex interfaces. However, such complexity is difficult to avoid as long as designers use 2D interfaces (such as screens and sketching pads) to create 3D models.

Accompanying the recent developments of immersive video games and an increasing interest in the metaverse, virtual reality (VR) interfaces have become widespread among the general public. VR allows users to work and manipulate objects directly in a 3D environment. It is, therefore, a natural choice for 3D modelling, alleviating the restrictions of 3D modellers to work on 2D screens.

In VR, the user can easily draw 3D curves, although working directly in 3D using head-mounted displays and hand-held controllers is not as straightforward as one might think (see [AKA*17]). Despite the imprecision of 3D drawings, VR sketching tools such as Tilt brush or OpenBrush are spreading at the early stages of design, also known as the ideation phase [YAS*21], which facilitates the creation of shapes in 3D as if using a pencil in 2D. While generating accurate 3D shapes is not needed at this early stage, getting even a rough 3D model would enable users to better perceive the volume they are modeling, thanks to hidden parts. Therefore, complementing the ideation design stage with the generation of an approximate 3D model is a useful challenge to tackle. In line with previous work on the topic (see Sec. 2), our work introduces a novel method to generate a plausible 3D manifold surface mesh given a 3D sketch. Our method is robust to the sparsity and lack of connectivity found in VR sketches and requires minimal user intervention.

Our goal is to develop a surface generation (or "surfacing") method able to solve the challenging problem of inferring closed surface geometries from any sparse set of unconstrained 3D strokes, thus offering flexibility and freedom to users drafting 3D shapes. Our main contribution, inspired by a balloon inflation metaphor, is a surfacing method that compensates for the lack of topological information and converts sparse and unoriented 3D stroke drawings into closed manifold surfaces. We pose the problem as an interpolation problem over sampled points on the strokes. This is achieved thanks to an adaptive mesh, initialized as a maximal sphere inside the set of strokes, which then inflates as a balloon and iteratively interpolates through the sample points. We cast the deformation method as the resolution of a biharmonic Laplacian with linear constraints augmented by an intermediate step of virtual surface expansion while continuously remeshing the surface to ensure robustness and maintain an organic appearance. If needed, extra balloons are automatically seeded and inflated to robustly process folded surfaces. The main features of our method are:

- **Balloon-inflation.** The technical core is a new surfacing algorithm that mimics the progressive inflation of a balloon inside the sparse scaffold formed by the strokes.
- **Concavities.** We extend the balloon inflation method by seamlessly integrating it into a divide-and-conquer algorithm, allowing us to address surfaces folding back onto themselves automatically without any artefact.
- **Sparse set of free-form strokes.** Our method is designed to process mere 3D strokes without any topological information, such as knowledge about their connectivity or orientation. As hatches in a 2D sketch, the strokes might change curvature, zig-zag or loop over the surface.

2. Related work

We discuss surface reconstruction from different types of inputs, with an emphasis on 3D sketching and modeling in VR.

Shape reconstruction from point clouds. Surface reconstruction from point clouds is a well-studied topic in Geometry Processing. Starting from α-shapes [EM94], various algorithms were introduced to address this problem. While explicit solutions apply different filtering conditions on Delaunay triangulation [ACK01, DG03, BMR*99, POEM24], implicit solutions aim to fit an implicit function to the point cloud [KBH06, KH13]. The variational implicit method VIPSS [HCJ19] outperforms the previous ones in its ability to handle sparse and non-uniform point clouds. With the advances in Learning methods, neural solutions were introduced to generate surfaces from oriented dense or sparse point clouds [HGA*23, WGK*22] and extended towards the use of prior knowledge to aid in the reconstruction process [MLZH22, OB24].

Despite decades of research in this field, existing algorithms are still sensitive to various artefacts in the input data, including missing data and noisy samples, and fail when dealing with sparse 3D sketches. In contrast, our method was specifically designed to infer a manifold surface from 3D sketches with extremely sparse, non-uniform and non-oriented data, where large areas are empty, and sample points are only available along a few 3D strokes.

Shape reconstruction from curve networks. Finding patches in a curve network without any additional information is an inherently ambiguous problem, as some of the closed cycles in the network do not bound surface patches but cross sections. The classical methods [RSW*07, AJA12, AJG*13, ZJC13] in this direction usually search for an optimal surface using optimization and path-finding strategies. Adding information on normals helps to detect cycles [SHBSS17, YAS*21]. Similarly, computing a larger surface that interpolates a curve network was addressed in various ways [GJ12, BWSS12, SS14, PLS*15, SHBSS16].

However, all of these methods require structured input in the form of a connected curve network, plus additional information such as normals to compute the surface. They were not designed to handle sparse, disconnected sets of 3D strokes. It is worth noting that clean, connected curve networks are difficult to create using freehand sketching in VR. Indeed, while disconnected sketched curves can be snapped if they are close together, this requires frequent user intervention, which users have complained about [YAS*21] because it forces them to think of a surface in terms of patches or networks of curves, which hampers their creativity. In contrast, VRSurf frees them from such constraints, as strokes can be drawn in any order, from any angle (any normal), and the method does not require any connectivity information.

Surfacing unstructured 3D strokes. Thanks to recent developments in immersive environments using head-mounted displays and hand-held controllers, VR opens up new opportunities to create artistic 3D content. With the help of 3D sketching applications such as Cave-Painting, SculptrVR, GravitySketch or Tilt Brush, [KFM*01, Til16, Gra17, Scu19], anyone can easily sketch stylized curves/ribbons in 3D with simple hand movements. However, these applications can only create 3D sketches. The latter needs to be

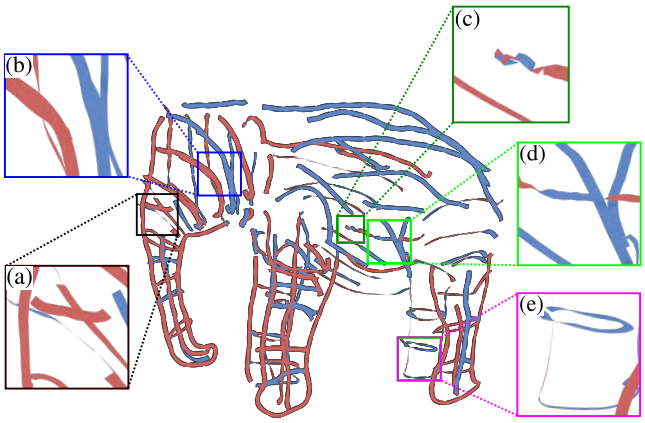


Figure 2: *Ribbon strokes captured in VR (with ribbon orientations shown in different colors) often contain commonly known issues: (a) Normal of the ribbon not aligning with the surface normal, (b) Arbitrary ribbon orientations, (c) Zig-zags at the stroke ends, (d) Inconsistent ribbon orientations along the stroke, (e) Sparse strokes. Thanks to VRSurf, we do not have to worry about these problems as it does not require any additional information.*

processed to create the desired surface. Different methods have tackled this problem so far. The first work in this direction, SurfaceBrush [RRS19], required users to nicely draw ribbon strokes side-by-side to generate a manifold surface. As this constraint is overly restrictive for VR artists, further work [YAB*22, BH24] relied on a user-given proxy surface to define the shape's topology. Closer to our work, [AOK12] approximate 3D sketches by gradually refining an initial coarse surface guided by a discrete vector field. However, their energy formulation is overly sensitive to stroke distribution and density. Other methods used deep learning to infer surfaces [BC22] but are still in the early stages and, therefore, cannot be used to create complex shapes. More recent works like [LCX*23, CDZ*24] proposed methods leveraging sparse VR sketches for learning-based 3D shape prototyping. However, the resulting shapes are limited to the category of trained shapes, e.g. chairs, cars, or planes. Another similar work worth noting is Stroke2Surface [RWSK24], which recovers a clean curve network from 4D data (unstructured strokes and time steps) together with additional meta-data applied to architectural design sketches. Lastly, a few techniques addressed direct bare-hand sculpting of 3D shapes in VR, rather than reconstructing them from strokes [SPS01, VSH19]. The latter is based on the sketch-based interface Fibermesh [NISA07], which offers sculpting operations such as pulling, extruding, or cutting a shape. However, curves implemented as positional constraints in surface optimization, like ours, act differently as control curves that the user can deform instead of drawing an entire surface in freehand.

Compared to these methods that tackled the same goal of surfacing 3D strokes, VRSurf offers greater freedom to the user since the orientation of the ribbon (or of the controller) is not taken into account, and no proxy surface is required.

3. Overview

General problem statement. The input of our method is the output of VR sketching systems, i.e. a raw set of unstructured 3D strokes. The latter may be sparsely distributed and spatially disordered (see Figure 2), and our goal is to infer a manifold surface from this sparse input, though it's important to note that this is an ill-posed problem. Indeed, given such a set of 3D curves, there is an infinite number of surfaces of different topological genus that interpolate them. While we wish to generate a solution that matches the user intent, not imposing any additional constraints on the stroke arrangement makes the problem quite challenging.

Insight: Smooth, closed shape assumption and Seed point. To make it easier to solve, we restrict the problem as follows: We first assume that we are looking for a single, smooth, closed 3D surface, interpolating all the strokes. As shown in the related work, inferring any surface requires establishing some sort of neighborhood relationship between the strokes. Rather than restricting their arrangement into rows or networks or requiring a user-designed proxy on which the strokes could be projected, we add an extra, minimal clue to the input: We ask the user to position a single seed point in space to indicate where the interior of the shape should be.

Inflating balloon metaphor. The seed point plus the smooth, closed surface assumption enable us to use a novel metaphor to solve the problem: Instead of trying to connect the strokes, we consider them as a scaffold in which a balloon-like surface inflates. The balloon (an adaptive mesh) is initialized at the seed point and is inflated until it reaches and progressively expands within the scaffold formed by the strokes treated as a sampled set of points.

Handling concave regions. Finally, the deformed balloon may not behave as expected in sparsely sampled concave regions, where the shape folds back onto itself. In such a case, it might self-intersect rather than spread properly through the junction. We detect such cases and automatically compute a new seed point from which a new balloon is inflated to interpolate the stroke samples on the other side of the fold. This process of splitting the set of points into several subsets captured by different balloons is repeated, if necessary, in a divide-and-conquer manner. It ends when all the sample points of all the input strokes are interpolated. The inflated balloons are then all merged into a single closed surface.

The next sections, respectively, detail the two main challenges to be solved: how to mimic an inflating balloon (Section 4) and how to detect and handle problematic concavities without any extra user intervention (Section 5).

4. Balloon inflation

Our goal is to mimic the inflation of a balloon initially centered at the user-specified seed point and inflate it inside the set of strokes, serving as a scaffold. Although our surface inflation process resembles a region growing, it is nevertheless very different from other region growing techniques used, e.g., for shape segmentation [LLL*22], surface reconstruction [KY05], mesh simplification [KT96] or sketch coloring [PCS21], as it combines surface

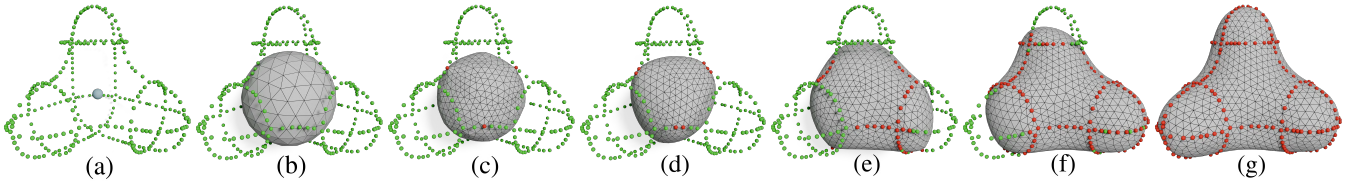


Figure 3: *Overview of the balloon inflation process. Uniformly sampled 3D strokes and user-selected seed point (a); Initial balloon, an empty sphere of maximal size which interpolates 4 of the input points (b); Resampled (c) and intermediate deformed shapes for the balloon's adaptive mesh after 25, 50, 150 iterations (d,e,f); Final shape (g).*

deformation with remeshing steps. Instead of using a computationally expensive physical simulation that might produce undesirable folded shapes around the strokes, we compute a sequence of smooth, adaptive close meshes that progressively grow along the strokes as well as *transversely* between them. Indeed, the key insight of our solution is to let this transversal growth discover a natural topological arrangement between them.

In the following, 3D strokes are considered as (but not restricted to) polylines whose vertices $\{\mathbf{p} \in I\!R^3\}$ are called *stroke points*. The successive balloon-like surfaces, growing throughout the inflation process, are represented each by a triangular surface mesh denoted $\mathcal{T} = \{V, E, F\}$ whose nodes $\mathbf{v} \in V$ will be called *vertices*. At each iteration, we denote by $\mathcal{P}$ the subset of stroke points that are interpolated by $\mathcal{T}$. Note that since these stroke points are interpolated by a mesh, we may also call them vertices of this mesh.

Figure 3 illustrates the processing pipeline described below. In this and the following figures, the already interpolated stroke points $\mathcal{P}$ are shown in red, those still to be processed in green, and the current mesh $\mathcal{T}$ in grey.

Background: Bi-harmonic surface interpolation. During both the initialization step and the inflation process, we will make use of biharmonic surface interpolation as the main deformation process successively applied to the balloon. It works as follows:

Let $\mathcal{T} = \{V, E, F\}$ be a triangular mesh, $\mathbf{v}_h$ a subset of its vertices, called *handles*, and $\mathbf{v}'_h$ new positions for these vertices. We denote $\mathcal{B}(\mathcal{T}, \mathbf{v}_h, \mathbf{v}'_h)$ the mesh with the same topology as $\mathcal{T}$, solution of the following biharmonic equation with interpolation conditions:

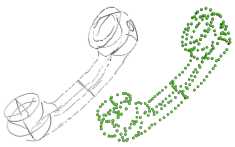
$$\Delta^2_{\mathcal{T}}\, \mathcal{B}(\mathcal{T}, \mathbf{v}_h, \mathbf{v}'_h) = 0$$
$$\text{subject to } \mathbf{v}_h = \mathbf{v}'_h\ ,$$

where $\Delta_{\mathcal{T}}$ is the discrete Laplace-Beltrami operator on $\mathcal{T}$ [BKP*10]. In other words, $\mathcal{B}(\mathcal{T}, \mathbf{v}_h, \mathbf{v}'_h)$ is the biharmonic mesh obtained by moving the handle vertices $\mathbf{v}_h$ of $\mathcal{T}$ to the positions $\mathbf{v}'_h$.

Although this deformation tends to minimize curvature between the handles, we consider it more appropriate than maintaining uniform curvatures as real balloons would do: in effect, it stretches the surface between the handles and will, therefore, prevent it from coming out of the scaffold formed by the strokes during the inflation process.

4.1. Initialization

Stroke points. As we impose no restriction on the input sketch, it can be composed of many small strokes, each defined as a polyline but with possibly quite different sampling rates. We decided to sub-sample all strokes on a regular basis, and its advantages are threefold: regularity will enable us to control mesh size, sampling density will enable us to control the speed of the inflation process, and equal sub-sampling for all strokes will not give more importance to one stroke over another. In a pre-processing step, we therefore sub-sample the input sketch into a set of regularly spaced *stroke points* $\mathbf{p}$, using voxel-based sub-sampling to eliminate any potential redundancy.



Using the **sampling rate** δ, the voxel size is set to δ·(bounding-box diagonal). In each voxel, we replace the polyline points with their centroid. The inset shows the result of sub-sampling on a 3D sketch from [HCJ19], which yields almost regularly sampled stroke points (green).

Initial Balloon. Once the user places the initial seed point locating the interior of the shape, we compute the first surface from which the inflation process will start. Inspired by [POEM24], we compute the 3D Delaunay triangulation of the stroke points, and $\mathcal{S}_{\text{start}}$ a surface triangulation of the circumsphere of the tetrahedron in which the seed point lies. Thanks to the empty-circle property of Delaunay triangulation, this sphere is empty of any other stroke point and thus fits inside the user-given sketch strokes. Let $\mathcal{P}_0$ be the set of points of the selected tetrahedron, already interpolated by $\mathcal{S}_{\text{start}}$. Our first biharmonic surface $\mathcal{T}_0$ is defined as a surface close to $\mathcal{S}_{\text{start}}$ but tighter on its handles, using;

$$\mathcal{T}_0 = \mathcal{B}(S_{\text{start}}, \mathcal{P}_0, \mathcal{P}_0)$$

Eventually, we remesh $\mathcal{T}_0$ following the procedure explained in Section 4.2. Fig. 3(b) shows $\mathcal{S}_{\text{start}}$, and Fig. 3(c) shows $\mathcal{T}_0$.

4.2. Inflation algorithm

While $\mathcal{T}_0$ only interpolates four stroke points, the aim of the inflation process is to progressively inflate it to interpolate one additional point at each iteration until all stroke points are processed or the expansion is blocked by some fold-over.

A naive solution would consist of simply letting the mesh grow through successive biharmonic interpolation of the additional

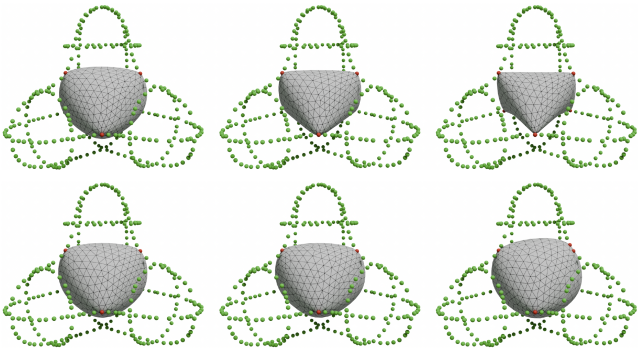


Figure 4: *Three balloon inflation instances without (top row) and with (bottom row) mesh expansion as detailed in Section 4.2.*

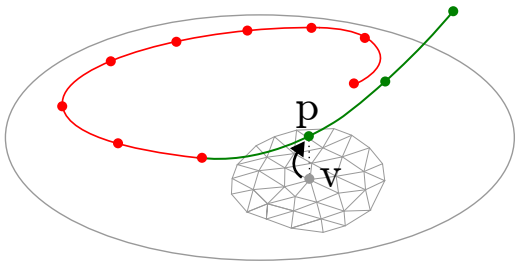


Figure 5: *Balloon inflation step. The current mesh (light grey) already interpolates a part of the input strokes (red). We select the non-interpolated stroke point* **p** *(green) closest to the mesh and the mesh vertex* **v** *closest to it. The balloon inflation step will deform the current mesh so that* **v** *moves towards* **p**.

stroke points. However, it turns out that this approach would lead to undesirable surfaces in very few steps due to increasing curvatures in the neighborhood of the interpolated handles, as shown in the top row of Fig. 4. We also observed undesirable shapes when experimenting with Laplacian operators of higher degrees (3 or 4). To overcome these issues, we perform an intermediate step before each biharmonic interpolation, which we call mesh expansion, and that will not only cancel the high curvature artefacts but also contribute to shape inflation. Additionally, as a last step, we remesh the biharmonic interpolated mesh.

The four steps of the inflation algorithm, namely next stroke point selection, mesh expansion, biharmonic interpolation, and remeshing are detailed below.

Selection of the next stroke point. To obtain a smooth final surface, it is essential to select the stroke points in an order that will deform the current closed mesh $\mathcal{T}$ as little as possible. Therefore, we select the next stroke point to be interpolated as closest to $\mathcal{T}$ (see Fig.5.):

$$\mathbf{p} = \underset{\substack{\mathbf{q}\ \text{stroke points}\\ \mathbf{q}\notin\mathcal{P}}}{\operatorname{argmin}} \operatorname{dist}(\mathbf{q},\mathcal{T}) \tag{1}$$

We also search for the vertex $\mathbf{v}$ of $\mathcal{T}\setminus\mathcal{P}$ - where $\mathcal{P}$ is the set of interpolated stroke points. The goal of the remaining steps is to pull the mesh vertex $\mathbf{v}$ towards the stroke point $\mathbf{p}$, as illustrated at the bottom of Fig. 5, the mesh vertex $\mathbf{v}$ acting as a *handle*.

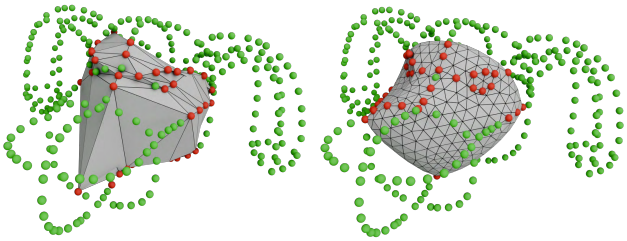


Figure 6: *Balloon inflation without (left) & with (right) remeshing.*

Mesh expansion. Before interpolating the newly selected stroke point $\mathbf{p}$, we first expand the current mesh $\mathcal{T}$ in the direction of its normal by the offset distance $\|\mathbf{p}-\mathbf{v}\|$. We denote $\overline{\mathcal{T}}$ this expanded mesh, and $\overline{\mathcal{P}},\overline{\mathbf{v}}$ the vertices on $\overline{\mathcal{T}}$ corresponding to $\mathcal{P}$, $\mathbf{v}$ in $\mathcal{T}$. Expanding the mesh locally decreases its curvature, which is important to prevent the appearance of the high-curvature artefacts mentioned above (see Fig. 4). Note that the expanded mesh no longer interpolates the stroke points.

Bi-harmonic interpolation: From the expanded mesh $\overline{\mathcal{T}}$, we compute the biharmonic interpolation of all the previously interpolated stroke points $\mathcal{P}$ as well as the newly selected point $\mathbf{p}$:

$$\mathcal{T} \longleftarrow \mathcal{B}(\,\overline{\mathcal{T}},\,\overline{\mathcal{P}}\cup\{\overline{\mathbf{v}}\},\,\mathcal{P}\cup\{\mathbf{p}\}\,) \tag{2}$$

Therefore, all previous interpolation properties are recovered while the new stroke point $\mathbf{p}$ is interpolated in a seamless manner. We may now update $\mathcal{P} \longleftarrow \mathcal{P}\cup\{\mathbf{p}\}$.

Remeshing. The iterative application of mesh expansion and biharmonic interpolation does not increase the number of triangles in the current mesh. If we only apply these steps during balloon inflation, there are two undesired effects occurring after a few iterations, illustrated in Figure 6, left: the triangles become very large, and not enough triangle vertices are available to handle selection. Inspired by the adaptive quasi-uniform mesh used in the Freestyle sculpting method [SCC11], we address these issues by applying the remeshing procedure introduced in [BK04] after each biharmonic deformation step.

Specifically, we choose the following target average edge length for the current mesh:

$$l_{\text{target}} = \min_{\substack{\mathbf{p},\mathbf{q}\ \text{stroke points}\\ \mathbf{p}\neq\mathbf{q}}} ||\mathbf{p}-\mathbf{q}|| \cdot m\,, \tag{3}$$

where m is a *remeshing parameter*, typically in the order of 1. Then we apply the remeshing procedure from [BK04] to get a new mesh whose edges have an approximate length of l_{target}, and whose vertices have valence close to 6. The only difference with [BK04] is that we fix the vertices $\mathcal{P}$ already pined to stroke points.

Improving robustness. The method we just presented might fail in some rare cases, where the tetrahedron containing the seed point

(see Section 4.1) may be close to degenerate, which prevents the balloon from inflating as expected. To address this issue, we modify the biharmonic interpolation process in Eq. (2) for the first n iterations to start from a sphere. More precisely, we replace $\overline{\mathcal{T}}$ by a spherical mesh co-centered with $\mathcal{S}_{\text{start}}$, with a radius equal to the sum of the offset distances used in the mesh expansion steps. In our implementation, n was set to ten percentage of the number of iterations (equal to the number of stroke points).

5. Handling surface fold-overs

While the mesh expansion step helps to avoid excessive positive curvature, it may introduce high negative curvature as it decreases the curvature even in concave regions of the mesh. It can then happen that the surface distorts, bulges or self-intersects, which in turn are undesired deformations. This effect appears especially when there are not enough stroke points around the concave region. Figure 8 illustrates a case where in the concave region, namely at the junction between the ears and the head for this specific dog-head example, growing a single balloon results in undesirable shapes such as the surface being attracted outside its local scaffold and possibly extending, self-intersecting, and gradually merging two adjacent sub-parts of the shape.

Instead of asking users to draw more strokes to better constrain the shape in such regions, we use an automatic divide-and-conquer approach to solve the problem. The idea is to automatically detect problematic deformations that may soon cause a fold-over of the inflating balloon, stop its growth, and start the growth of a new balloon from a new seed point, using the algorithm in Section 4. This process is repeated until all stroke points are interpolated. The different closed meshes resulting from balloon inflation are then merged to form a single, closed manifold surface.

5.1. Detection criterion

The probable occurrence of a fold-over deformation is detected using the Gauss curvature K, defined as the product between maximal and minimal curvature. It allows to classify a surface point as being elliptical ($K > 0$), hyperbolic ($K < 0$) and parabolic or flat ($K = 0$ in both cases). In the discrete setting, [MDSB03] shows that the Gauss curvature at a mesh vertex $\mathbf{v}$ with f faces around it is:

$$K(\mathbf{v}) = (2\pi - \sum_{i=1}^{f} \theta_i)/\mathcal{A},$$

where θ_i is the angle of the triangle at $\mathbf{v}$, and $\mathcal{A}$ is an area associated to $\mathbf{v}$. Negative Gaussian curvature characterizes concave regions.

Typically, a fold-over deformation happens several iterations *after* a concave region has emerged in the balloon, and has the effect of suddenly changing the negative Gauss curvature at a previously interpolated stroke point in this concave region. To detect such changes, we keep track of the Gauss curvature at each interpolated stroke point and launch the seeding and growth of a new balloon whenever there exists a previously interpolated stroke point $\mathbf{p}$ (not necessarily the last one) in a region of negative curvature (4) with an abruptly changing curvature at the current iteration (5)

$$K_{\text{before}}(\mathbf{p}) < 0 \tag{4}$$

$$|K_{\text{before}}(\mathbf{p}) - K_{\text{after}}(\mathbf{p})| > c. \tag{5}$$

c is a threshold value with the effect of having a smaller value for 'c' will place new spheres for every small curvature change, and a higher value will place the new sphere only after the undesired deformation happens. The parameter serves as a threshold value that determines when new spheres are placed. The smaller the value of c, the more often new spheres are placed for all small changes in curvature. In contrast, a larger value of c delays the placement of new spheres until significant deformation or curvature changes occur.

5.2. Automatic seeding of new balloons

As we just explained the mesh vertex with a negative curvature suddenly changing, where the problem was discovered, is among those that already interpolate a stroke point $\mathbf{p}_c$. We first revert back the deformation to the state of the mesh *just before interpolating* $\mathbf{p}_c$, and pause the inflation of the current balloon.

Let $\mathbf{v}_c$ denote the mesh vertex that was closest to $\mathbf{p}_c$ when it was interpolated; see Figure 5. We start a new balloon inflation using $\mathbf{v}_c$ as the seed point and let it run until either all stroke points are interpolated or the distance between a selected stroke point $\mathbf{p}$ and its closest mesh vertex $\mathbf{v}$ (see Section 4.2) is larger than a threshold value. In our experiments, we set the distance threshold to $9 \cdot \delta$. This prevents the inflation from incorporating stroke points in a region where they should not be interpolated by the new balloon. Growth is applied recursively - which means that when the current balloon can no longer grow, we resume the growth of the previous balloon, left in pause mode.

Note that the choice of seeding the new balloon at the detected fold peak $\mathbf{p}_c$ and not elsewhere has two reasons. First, it ensures that no gaps between neighboring balloon meshes occur, which would make the subsequent merging impossible. Second, it segments the balloon into pieces without strong concavities.

5.3. Seamless merging process

The algorithm stops when all stroke points have been interpolated, possibly resulting in a series of several interpenetrating closed meshes, each generated by a different inflation process. In this case, a final shape is obtained by combining all these balloons using a boolean merge operation.

Figure 7 shows the full processing of a dog head, where the two concave regions at the ears led to two interruptions of the growth of the initial surface. First, in Figure 7 (b,c), the right ear is seeded by a new growing balloon, and then the same occurs at the left ear (e,f). The final surface (g) is the result of the merge of the three balloon meshes (b,d,f).

6. Results and Evaluation

Throughout this paper, results of VRSurf and other methods are presented using flat shading to increase the saliency of potential smoothness problems in the output meshes.

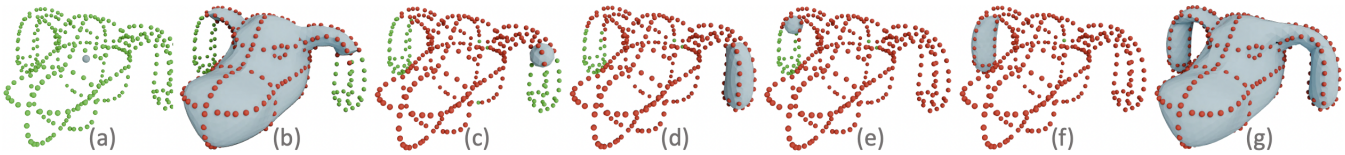


Figure 7: *Divide and conquer approach for handling fold-overs: From the input stroke and seed points (a), a first balloon inflates until a fold-over deformation is detected by the left ear of the dog (b). The inflation is paused, and a new balloon is inflated from the fold-over peak (c,d). The remaining stroke points being too distant, the inflation of the first balloon resumes. A second fold-over is detected and processed (e,f). Finally, all the stroke points being interpolated, the three balloons are merged into a single, manifold closed mesh (g).*

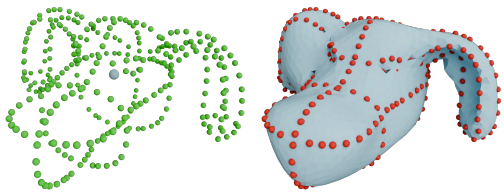


Figure 8: *If the first balloon was fully inflated, undesirable deformation would occur near concave regions due to fold-overs of the inflating mesh (left ear glued to the head, right ear with a bulge).*

6.1. Categories of input

Our algorithm can be used to model a 3D shape either from a clean set of curves or from a coarser, disordered VR input.

Wireframe curves and clean strokes points. We first tested our method on 3D wireframe curves that were supposed to lie exactly on some (unknown) 3D shape. We used the curve data from [HCJ19], which resulted in the trebol (Fig. 3), the dog (Fig. 7) and the spiral (Fig. 9-left). The resulting shapes are smooth and look faithful to the input. The dog was a highly challenging case since it not only has prominent features but also fold-overs where different parts of the shape come close together. It could not have been processed without our method for robustly handling surface fold-overs.

VR-sketched curves. Most of the results shown in this paper were produced using OpenBrush [Ope20], a 3D sketching tool in VR. Indeed, the main motivation for our method was to provide a surface tool for sparse and disorganized 3D sketches, where the user creates a shape in 3D as if using a pencil in 2D. As shown in Figures 1 and 10, we achieve the generation of the intended rough shape from such sparse sets of strokes. Furthermore, in Fig. 11 we used sketches from the 3d-sketches-curated-dataset [Yu23], originally taken from other interactive systems such as GravitySketch, True2Form, FlowRep and CurveFusion [Gra17, XCS*14, GSV*17, LCC*18]. And finally, we applied our method to freehand sketches of the phone and the walrus borrowed from [HCJ19], which are available in the form of a point set, see Figure 9, middle and right.

It is well known that 3D stroke drawings in VR are not smooth, they are generally not closed, and cannot intersect precisely [AKA*17]. In practice, we asked the user to avoid the intersection case by only drawing short strokes to avoid possible unnecessary bumps that might arise near such intersection regions. In return, users who have done the VR sketches in Fig. 10 noted that it was easy to create with a head-mounted VR system because there were no restrictions on handheld-controller orientation. Sketching shapes in this way felt like painting objects with short brushstrokes. While avoiding intersections with shorter lines was an imposed limitation, it was still seen as easier and more manageable than having to generate perfectly intersecting curves.

Although these results clearly suggest familiar shapes, the inaccuracy of the input sketches does not allow for the generation of a very smooth surface because of the interpolation of the input strokes. However, let us stress that our approach of mimicking the inflation of a balloon is resilient to the typical inaccuracy of 3D VR sketches and solves the topology problem inherent to disordered 3D drawings so that, in all examples, it allows the imagined shape to be recovered. The elephant example (see Fig. 1) worked surprisingly well without requiring any additional balloons. Although we assume that a sketch has been pre-drawn, a surface can be progressively enhanced, as illustrated in Fig. 12. In this case, when new strokes are added, the previously generated surface serves as the initial surface, and the old strokes act as position constraints. If a stroke is deleted, we can rerun the biharmonic deformation while releasing the position constraints imposed by the deleted stroke sample.

6.2. Evaluation of the method

To precisely evaluate VRSurf, we discuss parameter choices and performances, compare the results with ground truth, and finally compare against recent surface reconstruction methods.

Influence of parameter values. Let us first discuss the **seed point** that defines the position of the initial balloon. It must lie "inside" the imagined shape, but since 3D strokes are not oriented, "inside" is not clearly defined. Consequently, positioning the seed point (depicted as a small blue ball in all our examples) is the only user intervention that we require. The final result, however, depends on its location: Our experience has shown that starting with a large balloon is best, as it needs less inflation than a small one, resulting in a smoother final surface. It is, therefore, recommended that the seed point be positioned in the middle of the thickest part of the imaginary shape volume.

Sampling parameters first include the **stroke sampling resolution** δ. The latter controls the voxel size used to downsample the input strokes with nearly equidistant stroke points. Its influence on

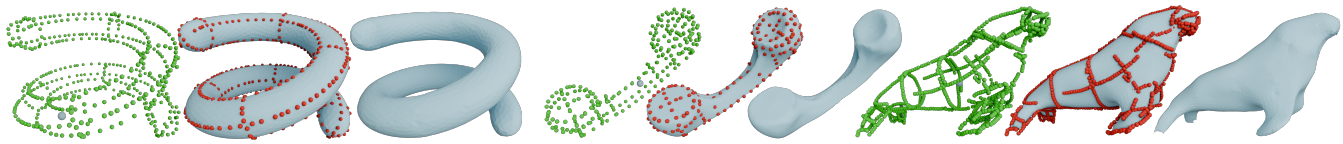


Figure 9: *VRSurf results on clean wire-frame curves (spiral) and freehand 3D sketches borrowed from [HCJ19]. Left to right: Sampled 3D strokes, result after balloon inflation, and final result.*

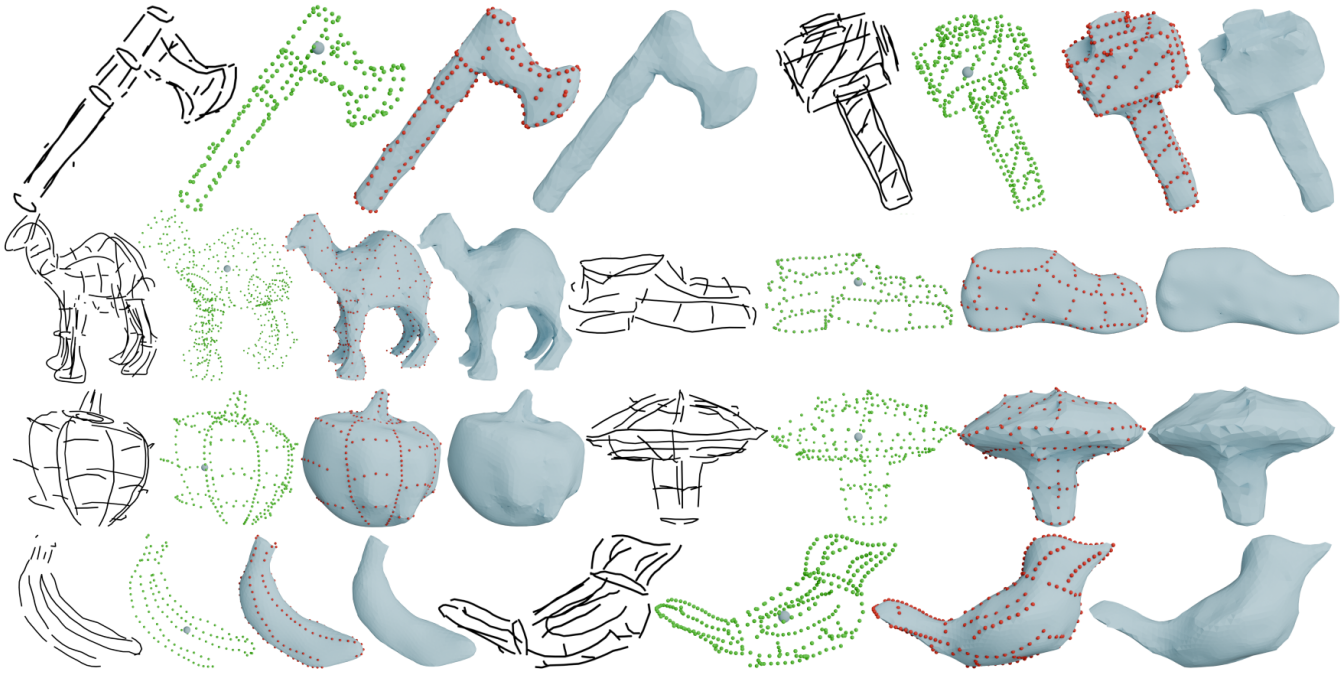


Figure 10: *VRSurf results on 3D sketches made with a VR drawing system of varying complexity. Note that a novice user took 2-5 minutes to create each of these sketches.*

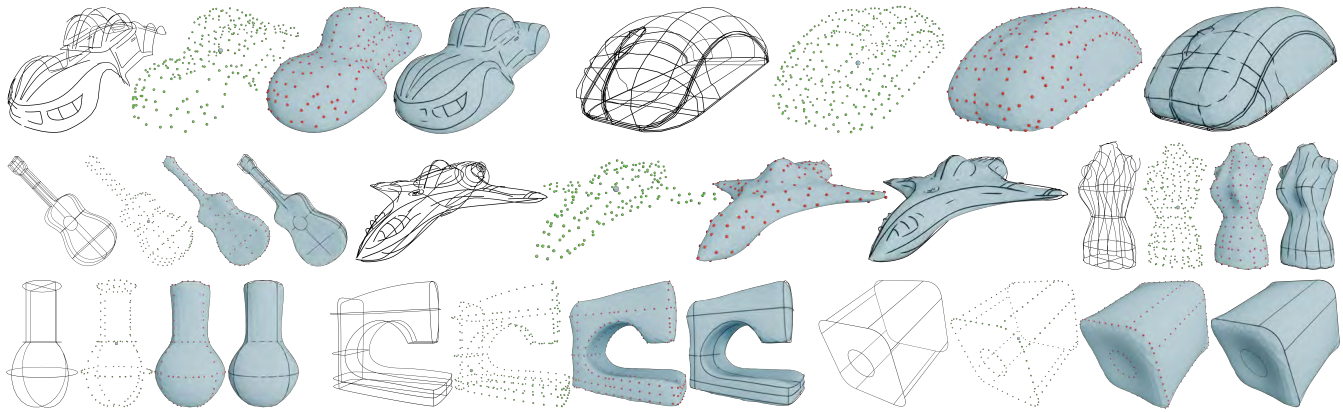


Figure 11: *VRSurf results on 3D sketches taken from the 3D-sketches-curated-dataset [Yu23] after removing redundant strokes. From left to right (for each model): input 3D sketch, sampled stroke points, final results with interpolated stroke points, the final result with 3D sketch.*

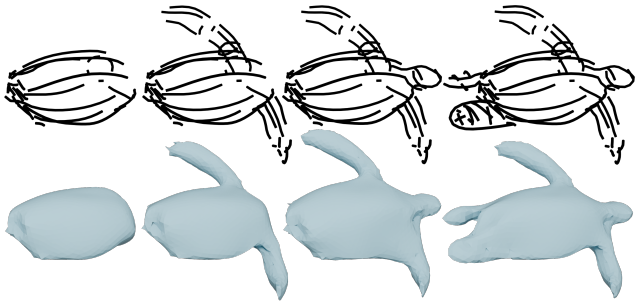


Figure 12: *A surface can be improved step by step by adding new features. In this case, the previously created surface serves as the starting surface and the old strokes as positional constraints.*

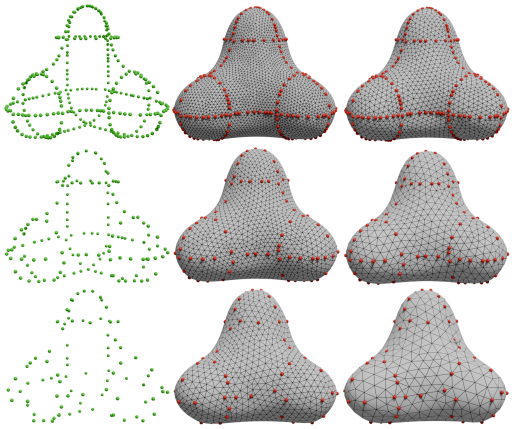


Figure 13: *Results from the same input sketch using sampling parameter values* $\delta = 0.03/0.06/0.09$ *for the strokes (lines) and remeshing parameter values* $m = 1.0/1.5$ *(columns), from the initial mesh resolution proportional to one of the strokes. The number of stroke points from top to bottom is resp. 289/141/87.*

the shape is illustrated in Figure 13, where the first column shows the stroke points of increasing voxel sizes. In the second column, we show the **mesh resolution**, which is usually automatically set from the stroke resolution using $m = 1$ for the remeshing parameter value in equation (3). When the balloon is inflated, the remeshing step automatically increases the number of triangles, if necessary, to maintain a constant average length of edges. It is also possible to custom-tune the mesh resolution. The third column of Figure 13 shows the resulting surfaces computed at a lower mesh resolution.

Statistics and performances. We implemented our method in CPP using the LibIGL library on an Apple Macbook M3 Pro using CPU only. Runtimes vary from seconds to minutes, depending on the number of input stroke points, which determine the number of bi-harmonic interpolation and remeshing steps, see Table 1. The bottleneck is the remeshing step, whose computational cost is much higher than that for bi-harmonic interpolation. In addition, to maintain a constant triangle size, the number of mesh vertices increases during inflation from simple to double, which in turn increases calculation time. As an example, the computation times for the two surfaces in the middle row of Figure 13 take resp. 15 min, 5 min. For the banana with much fewer mesh triangles, the runtime is only 43s. Users can manually reduce mesh size and thus reduce calculation time, but overly coarse meshes are often not smooth enough.

models	$\#strokes$	$\lvert\mathcal{P}\rvert$	$\lvert V_0\rvert$	$\lvert V_{final}\rvert$	(R)	
trebol(VIPSS)		140	1398	2551	705s	Fig. 3
dog(VIPSS)		303	509	1676	744s	Fig. 7
banana(author)	19	120	42	367	43s	Fig. 10
axe(author)	38	212	120	1656	459s	Fig. 10
hammer(author)	44	363	324	2101	18min	Fig. 10
elephant(author)	119	661	520	3765	63min	Fig. 1

Table 1: *We provide model statistics for a few representative models, which gives orders of magnitude valid for all our models: number of strokes, number stroke points* $|\mathcal{P}|$*, number of mesh vertices of initial balloon* $|V_0|$*, final surface* $|V_{final}|$*, and total runtimes (R).*

Validation against ground truth. Figure 15 shows the results of our method using input strokes which lie exactly on a known surface, enabling us to evaluate the accuracy of reconstruction. For the duck example, the root-mean-square (RMS) error is 0.0143 for a bounding-box diagonal of 7.98. The fish example has an RMS error of 0.0467 for a diagonal of 12.15. We also applied our method to an academic model, a half-torus, see Figures 17, 18, 19. Herein, we tested the robustness by varying the sampling density of stroke points (Fig. 17), the position of seed points (Fig. 18), and the number of strokes (Fig. 19). For all examples, we measured the Hausdorff distance from the ground truth, knowing that the strokes were drawn exactly on the given surfaces.

Comparisons against surface reconstruction from point clouds. Figure 14 gathers comparison results with three previous reconstruction methods from point clouds: We show the input followed by our results, then, the well-known Screened Poisson [KH13] reconstruction (third row), the implicit surface reconstruction method VIPSS [HCJ19] (fourth row), and the recent Ball Merge [POEM24] (bottom). While VIPSS is known for its ability to handle sparse, non-uniform point clouds while not requiring any normal information, it struggles to handle our sparse sampled strokes and, like the other methods, often fails to infer the intended shape.

6.3. Limitations

While an effective method for surfacing arbitrary wire-frame curves or for conveying 3D shapes from a VR sketch, our surfacing method suffers several limitations.

The first one is the lack of interactivity, which currently prevents using the method in real-time during incremental sketching in VR. The main reason for high computational times lies in the approach of considering only a single handle point per inflation step. Considering multiple support points simultaneously could speed up the process by reducing the number of iterations of the bi-harmonic interpolation and remeshing steps. However, this would

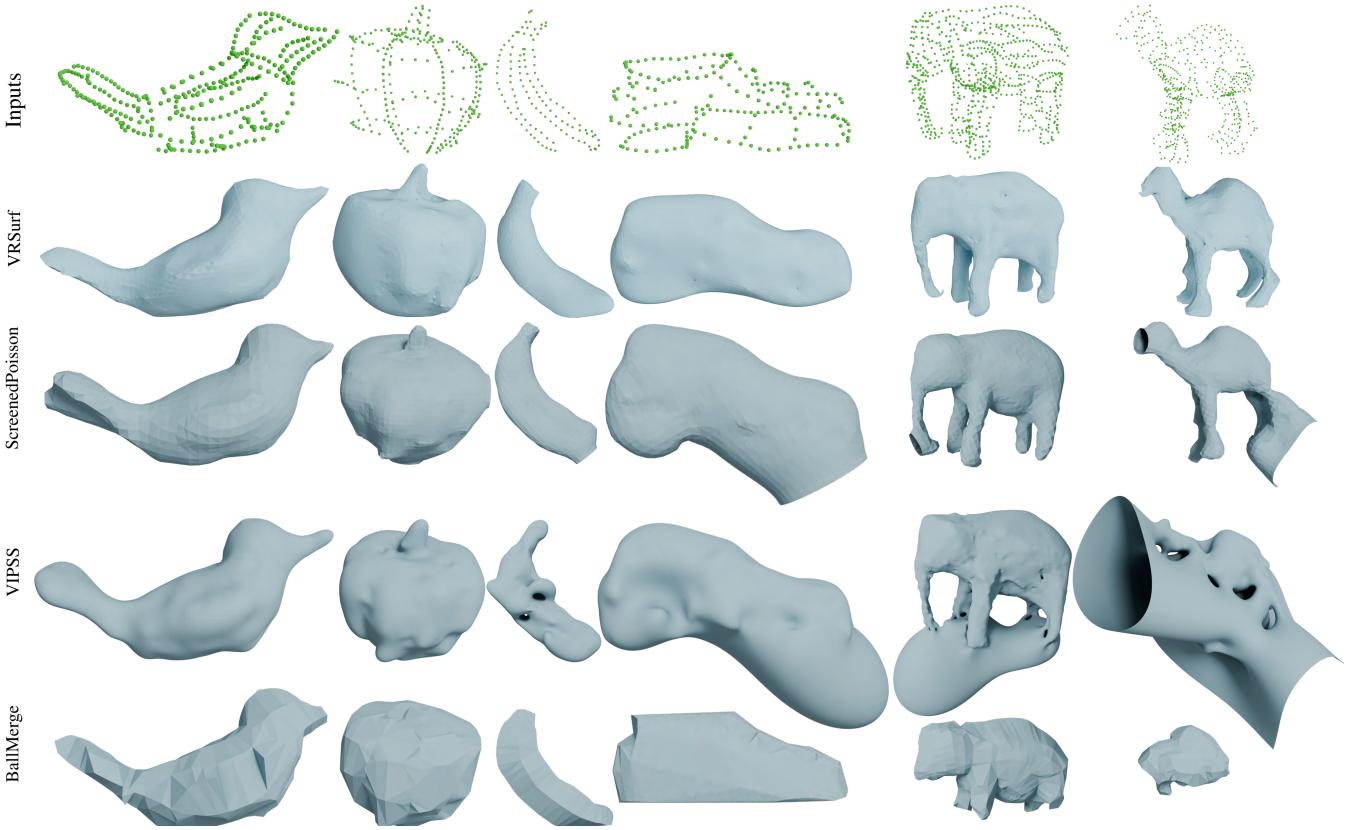


Figure 14: *Comparison against state-of-the-art surface reconstruction methods. Stroke points from sketches (row 1) are shown in increasing order of complexity in terms of both shape and stroke density. We compare VRSurf (row 2) with the implicit reconstruction method Screened Poisson [KH13] (row 3), the sparse sample reconstruction method VIPSS [HCJ19] (row 4), and the point cloud reconstruction method BallMerge [POEM24] (bottom row).*

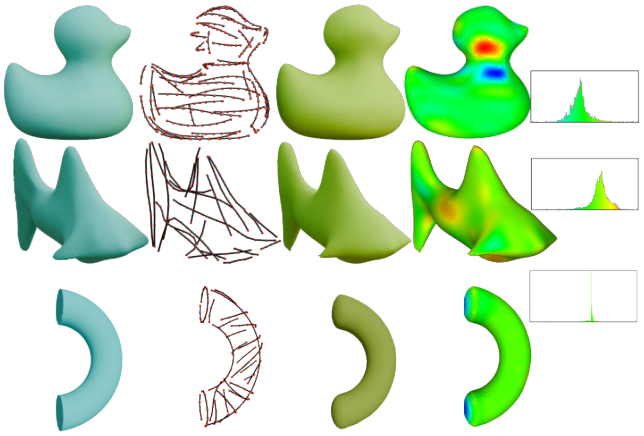


Figure 15: *Comparison to ground truth. Left to right: reference surface, 3D strokes lying on the reference surface, our result, the signed distance between the reference and resulting meshes, and the histograms (see text for error measures).*

likely complicate fold-over detection, a challenge that remains to be addressed in future work. Additional computational savings might be achieved by reducing the frequency of remeshing operations, for instance, by implementing a local remeshing strategy or performing remeshing only every k-th step. However, this approach carries the risk of insufficient samples in areas with multiple deformations, potentially compromising the smoothness and quality of the resulting surface. A more effective solution could involve developing an intelligent mechanism to determine when remeshing is truly necessary, accelerating the remeshing process itself, or locally refining the surface where needed.

Second, VRSurf always generates a single, manifold closed surface of genus 0 since it starts with a balloon shape and progressively inflates it. Although additional balloons may be created and finally merged, our solution does not allow them to intersect by covering twice the same data point. Therefore, an input of a non-zero topological genus such as a torus or a mug will need user intervention to close the open loops in shape by merging neighboring parts, see Fig. 16(left). To make such merging automatic, we could inspire from the sculpting system [SCC11], where all shape parts that come at a distance smaller than a given threshold distance automatically merge together. The other failure cases include situations

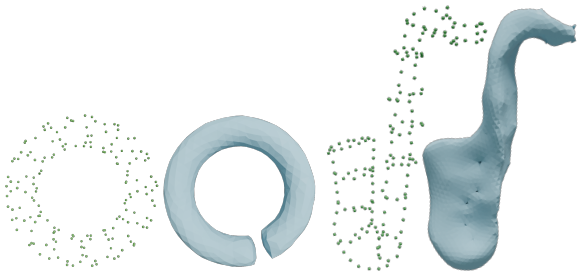


Figure 16: *Limitations. VRSurf outputs shapes of genus 0. Indeed, the inflation stops without covering twice the same point, so some user intervention would be needed to merge any remaining open loop, as in this torus example (left). VRSurf also fails if the input sketch is too sparse, especially with strong surface fold-backs, as in this example where the user's intent was a saxophone (right).*

where the user defined sketch is definitely too sparse, even for our method, see Fig. 16(right).

Lastly, we assumed that the sampled strokes were to be interpolated. Therefore, any crossing between input strokes should exactly go through the same point, with the same tangent plane, which is extremely difficult to achieve in VR. We turned around the problem by asking users to only sketch short, non-intersecting strokes (see Fig. 10). Moreover, the small gaps that need to be left between the strokes may result in surface imperfections.

7. Conclusion

VRSurf was designed to help convey 3D shapes in VR. Complementary to existing 3D stroke drawing tools, it relies on a balloon inflation metaphor to transform any sparse sketch into a closed manifold surface, which in our experiments appears to match quite well the user's intent. The advantage over previous methods is their ease of use, since no specific arrangement or normal information is required for the input strokes.

In complement to the strokes, the user needs to define a seed point. Although this never was a problem in our experiments, investigating methods to automatize this would be an interesting addition, although it could be challenging for shapes with highly concave parts.

Automatically handling shapes of any topological genus would be a great extension of our method. The mesh merging tools, which are readily available and trigger the decision whether close parts need to be merged or not, could be inspired by the adaptive, quasi-uniform mesh introduced in Freestyle [SCC11], which can seamlessly switch topological genus during an interactive sculpting session.

In future work, it might also be worthwhile to design an automatic method to pre-process VR coarse input in order to make our method robust to approximately crossing curves (allowing more diverse sketching styles), or alternatively, consider an approximation of the stroke sample points instead of interpolation.

Lastly, our surfacing method was built on a smooth shape assumption, which made our choice of an inflating balloon metaphor quite consistent. Extending 3D shape design in VR to shapes with sharp features, where the intended sharp edges could be automatically guessed, would be a challenging but useful extension.

Acknowledgements

The authors thank all anonymous reviewers for their constructive feedback. The work is partially funded by the ANR JCJC project SketchMAD (ANR-23-CE33-0009) and generous support from Adobe.

References

[ACK01] AMENTA N., CHOI S., KOLLURI R. K.: The power crust. In *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications* (2001), SMA '01, ACM, p. 249–266. 2

[AJA12] ABBASINEJAD F., JOSHI P., AMENTA N.: Surface patches from unorganized space curves. In *Proceedings of the Twenty-Eighth Annual Symposium on Computational Geometry* (2012), SoCG '12, ACM, p. 417–418. 2

[AJG*13] ABBASINEJAD F., JOSHI P., GRIMM C., AMENTA N., SIMONS L.: Surface patches for 3D sketching. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (2013), SBIM'13, ACM, p. 53–60. 2

[AKA*17] ARORA R., KAZI R. H., ANDERSON F., GROSSMAN T., SINGH K., FITZMAURICE G.: Experimental evaluation of sketching on surfaces in VR. In *Conference on Human Factors in Computing Systems* (2017), CHI'17, p. 5643–5654. 2, 7

[AOK12] ARISOY E. B., ORBAY G., KARA L. B.: Free form surface skinning of 3D curve clouds for conceptual shape design. *J. Comput. Inf. Sci. Eng. 12* (2012). 3

[BC22] BHATTACHARJEE S., CHAUDHURI P.: Deep interactive surface creation from 3D sketch strokes. In *International Joint Conference on Artificial Intelligence* (2022). 3

[BH24] BONNEAU G.-P., HAHMANN S.: 3D sketching in immersive environments: Shape from disordered ribbon strokes. *Computers and Graphics 123* (Oct. 2024), 103978. 3

[BK04] BOTSCH M., KOBBELT L.: A remeshing approach to multiresolution modeling. In *Symposium on Geometry Processing* (2004), Scopigno R., Zorin D., (Eds.), The Eurographics Association. 5

[BKP*10] BOTSCH M., KOBBELT L., PAULY M., ALLIEZ P., LÉVY B.: *Polygon Mesh Processing*. A K Peters, 2010. 4

[BMR*99] BERNARDINI F., MITTLEMAN J., RUSHMEIER H., SILVA C., TAUBIN G.: The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics 5*, 4 (1999), 349–359. 2

[BWSS12] BESSMELTSEV M., WANG C., SHEFFER A., SINGH K.: Design-driven quadrangulation of closed 3D curves. *ACM Trans. Graph. 31*, 6 (Nov. 2012). 2

[CDZ*24] CHEN T., DING C., ZHANG S., YU C., ZANG Y., LI Z., PENG S., SUN L.: Rapid 3D model generation with intuitive 3D input. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2024), pp. 12554–12564. 3

[DG03] DEY T. K., GOSWAMI S.: Tight cocone: a water-tight surface reconstructor. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications* (2003), SMA'03, ACM, p. 127–134. 2

[EM94] EDELSBRUNNER H., MÜCKE E. P.: Three-dimensional alpha shapes. *ACM Transactions On Graphics (TOG) 13*, 1 (1994), 43–72. 2

[GJ12] GRIMM C., JOSHI P.: Just drawit: a 3D sketching system. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (2012), SBIM '12, Eurographics Association, p. 121–130. 2

[Gra17] GravitySketch, 2017. https://www.gravitysketch.com/. 2, 7

[GSV*17] GORI G., SHEFFER A., VINING N., ROSALES E., CARR N., JU T.: FlowRep: descriptive curve networks for free-form design shapes. *ACM Trans. Graph. 36*, 4 (jul 2017). 7

[HCJ19] HUANG Z., CARR N., JU T.: Variational implicit point set surfaces. *ACM Trans. Graph. 38*, 4 (jul 2019). 2, 4, 7, 8, 9, 10

[HGA*23] HUANG J., GOJCIC Z., ATZMON M., LITANY O., FIDLER S., WILLIAMS F.: Neural kernel surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 4369–4379. 2

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (2006), SGP '06, Eurographics Association, p. 61–70. 2

[KFM*01] KEEFE D. F., FELIZ D. A., MOSCOVICH T., LAIDLAW D. H., LAVIOLA J. J.: CavePainting: a fully immersive 3D artistic medium and interactive experience. In *Symposium on Interactive 3D Graphics* (2001), I3D'01, p. 85–93. 2

[KH13] KAZHDAN M., HOPPE H.: Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG) 32*, 3 (2013), 1–13. 2, 9, 10

[KT96] KALVIN A. D., TAYLOR R. H.: Superfaces: Polygonal mesh simplification with bounded error. *IEEE Comput. Graph. Appl. 16*, 3 (1996), 64–77. 3

[KY05] KUO C.-C., YAU H.-T.: A delaunay-based region-growing approach to surface reconstruction from unorganized points. *Computer-Aided Design 37*, 8 (2005), 825–835. 3

[LCC*18] LIU L., CHEN N., CEYLAN D., THEOBALT C., WANG W., MITRA N. J.: CurveFusion: reconstructing thin structures from RGBD sequences. *ACM Trans. Graph. 37*, 6 (Dec. 2018). 7

[LCX*23] LUO L., CHOWDHURY P. N., XIANG T., SONG Y.-Z., GRYADITSKAYA Y.: 3D VR sketch guided 3D shape prototyping and exploration. In *IEEE/CVF International Conference on Computer Vision (ICCV)* (2023), pp. 9233–9242. 3

[LLL*22] LIN C., LIU L., LI C., KOBBELT L., WANG B., XIN S., WANG W.: SEG-MAT: 3D shape segmentation using medial axis transform. *IEEE Transactions on Visualization and Computer Graphics 28*, 6 (2022), 2430–2444. 3

[MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III* (2003), Hege H.-C., Polthier K., (Eds.), Springer Berlin Heidelberg, pp. 35–57. 6

[MLZH22] MA B., LIU Y.-S., ZWICKER M., HAN Z.: Surface reconstruction from point clouds by learning predictive context priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 6326–6337. 2

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: FiberMesh: designing freeform surfaces with 3D curves. *ACM Trans. Graph. 26*, 3 (2007), 41–es. 3

[OB24] OUASFI A., BOUKHAYMA A.: Robustifying generalizable implicit shape networks with a tunable non-parametric model. *Advances in Neural Information Processing Systems 36* (2024). 2

[Ope20] OpenBrush, 2020. https://github.com/icosa-foundation/openbrush. 7

[PCS21] PARAKKAT A. D., CANI M.-P. R., SINGH K.: Color by numbers: Interactive structuring and vectorization of sketch imagery. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (2021), CHI '21, ACM. 3

[PLS*15] PAN H., LIU Y., SHEFFER A., VINING N., LI C., WANG W.: Flow aligned surfacing of curve networks. *ACM Trans. Graph. (SIGGRAPH) 34*, 4 (2015). 2

[POEM24] PARAKKAT A. D., OHRHALLINGER S., EISEMANN E., MEMARI P.: Ballmerge: High-quality fast surface reconstruction via voronoi balls. *Computer Graphics Forum 43*, 2 (2024). 2, 4, 9, 10

[RRS19] ROSALES E., RODRIGUEZ J., SHEFFER A.: SurfaceBrush: from virtual reality drawings to manifold surfaces. *ACM Trans. Graph. 38*, 4 (2019). 3

[RSW*07] ROSE K., SHEFFER A., WITHER J., CANI M.-P., THIBERT B.: Developable Surfaces from Arbitrary Sketched Boundaries. In *Geometry Processing* (2007), Belyaev A., Garland M., (Eds.), The Eurographics Association. 2

[RWSK24] RASOULZADEH S., WIMMER M., STAUSS P., KOVACIC I.: Strokes2Surface: Recovering curve networks from 4D architectural design sketches. *Computer Graphics Forum 43*, 2 (2024), e15054. 3

[SCC11] STANCULESCU L., CHAINE R., CANI M.-P.: Freestyle: Sculpting meshes with self-adaptive topology. *Computers & Graphics 35*, 3 (2011), 614–622. 5, 10, 11

[Scu19] SculptrVR, 2019. https://www.sculptrvr.com/. 2

[SHBSS16] STANKO T., HAHMANN S., BONNEAU G.-P., SAGUIN-SPRYNSKI N.: Surfacing curve networks with normal control. *Computers & Graphics 60* (2016), 1–8. 2

[SHBSS17] STANKO T., HAHMANN S., BONNEAU G.-P., SAGUIN-SPRYNSKI N.: Shape from sensors: Curve networks on surfaces from 3D orientations. *Computers & Graphics 66* (2017), 74–84. Shape Modeling International 2017. 2

[SPS01] SCHKOLNE S., PRUETT M., SCHRÖDER P.: Surface drawing: creating organic 3D shapes with the hand and tangible tools. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2001), CHI '01, ACM, p. 261–268. 3

[SS14] SADRI B., SINGH K.: Flow-complex-based shape reconstruction from 3D curves. *ACM Trans. Graph. 33*, 2 (2014). 2

[Til16] Tilt Brush by Google, 2016. https://www.tiltbrush.com. 2

[VSH19] VERHOEVEN F., SORKINE-HORNUNG O.: RodMesh: Two-handed 3D Surface Modeling in Virtual Reality. In *Vision, Modeling and Visualization* (2019), Schulz H.-J., Teschner M., Wimmer M., (Eds.), The Eurographics Association. 3

[WGK*22] WILLIAMS F., GOJCIC Z., KHAMIS S., ZORIN D., BRUNA J., FIDLER S., LITANY O.: Neural fields as learnable kernels for 3D reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 18479–18489. 2

[XCS*14] XU B., CHANG W., SHEFFER A., BOUSSEAU A., MCCRAE J., SINGH K.: True2Form: 3D curve networks from 2D sketches via selective regularization. *ACM Trans. Graph. 33*, 4 (2014). 7

[YAB*22] YU E., ARORA R., BÆRENTZEN J. A., SINGH K., BOUSSEAU A.: Piecewise-smooth surface fitting onto unstructured 3D sketches. *ACM Trans. Graph. 41*, 4 (jul 2022). 3

[YAS*21] YU E., ARORA R., STANKO T., BÆRENTZEN J. A., SINGH K., BOUSSEAU A.: Cassie: Curve and surface sketching in immersive environments. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (2021), CHI'21, ACM. 2

[Yu23] YU E.: 3D sketches curated dataset, 2023. https://gitlab.inria.fr/D3/3d-sketches-curated-dataset. 7, 8

[ZJC13] ZOU M., JU T., CARR N.: An algorithm for triangulating multiple 3D polygons. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing* (2013), SGP'13, Eurographics Association, p. 157–166. 2

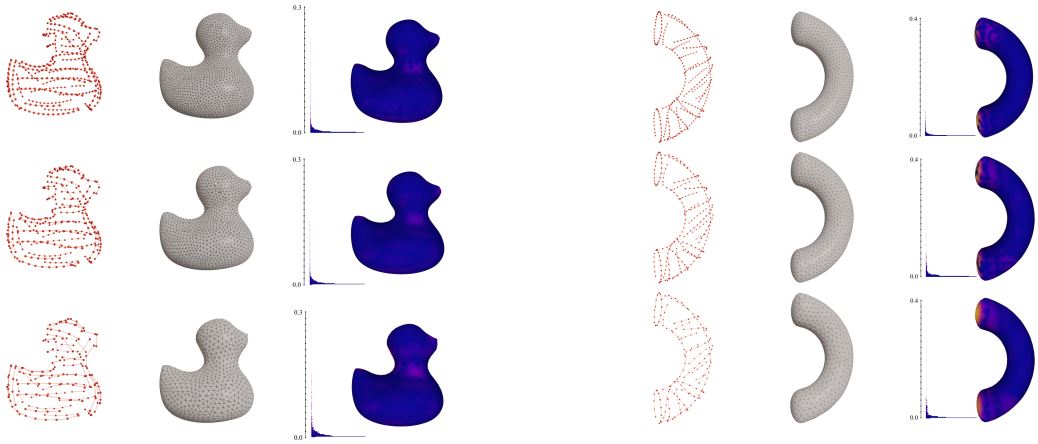


Figure 17: *Varying stroke sampling density. The number of stroke points used for the duck are 371,284,165 (from top to bottom), and for the half-torus, 334,254,165. For each model, the input sketch with sub-sampled stroke points lying exactly on a given surface (left), our result (middle), and comparison to ground truth using Hausdorff-distance with a distance histogram (right).*

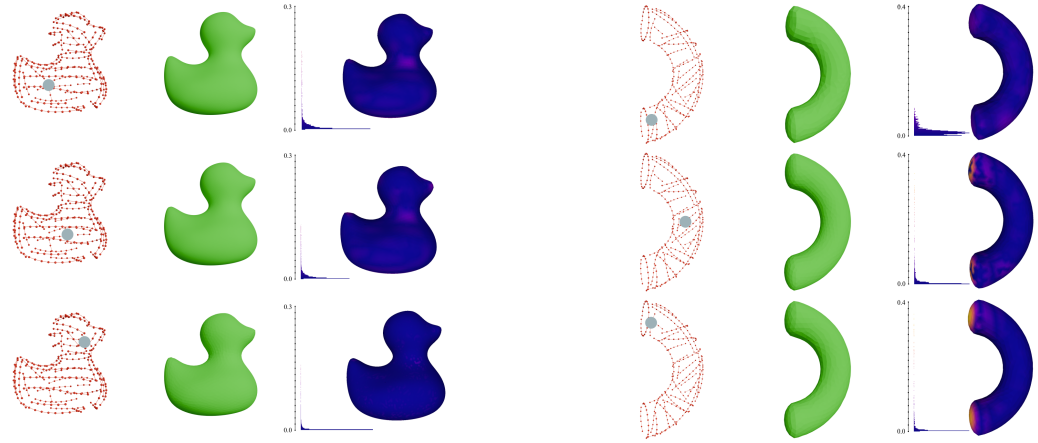


Figure 18: *Changing the position of the seed points (blue dot inside the sketch, left), our results (middle), and Hausdorff-distance to ground truth (right). Our results (green surface) appear robust against the choice of seed points.*

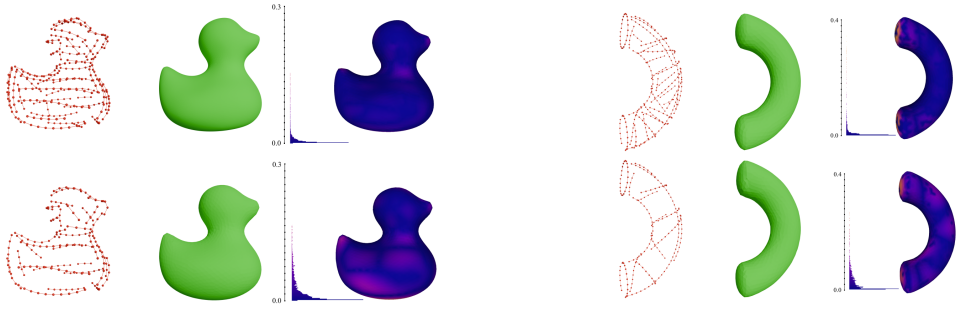


Figure 19: *Changing the number of input strokes: the duck was computed from 58 and 28 strokes, the half-torus using 26 and 15 strokes. Hausdorff-distance to ground truth. Even with as few as only 15 strokes, the RMS error on the reconstructed half-torus shape, for example, is less than 0.06 for a bounding-box diagonal of 11.45.*