CASSIE: Curve and Surface Sketching in Immersive Environments

Emilie Yu
Inria, Université Côte d'Azur
Sophia-Antopolis, France
emilie.yu@inria.fr

Rahul Arora
University of Toronto
Toronto, Ontario, Canada
arorar@dgp.toronto.edu

Tibor Stanko
Inria, Université Côte d'Azur
Sophia-Antopolis, France
tibor.stanko@inria.fr

J. Andreas Bærentzen
Technical University of Denmark
Kongens Lyngby, Denmark
janba@dtu.dk

Karan Singh
University of Toronto
Toronto, Ontario, Canada
karan@dgp.toronto.edu

Adrien Bousseau
Inria, Université Côte d'Azur
Sophia-Antopolis, France
adrien.bousseau@inria.fr

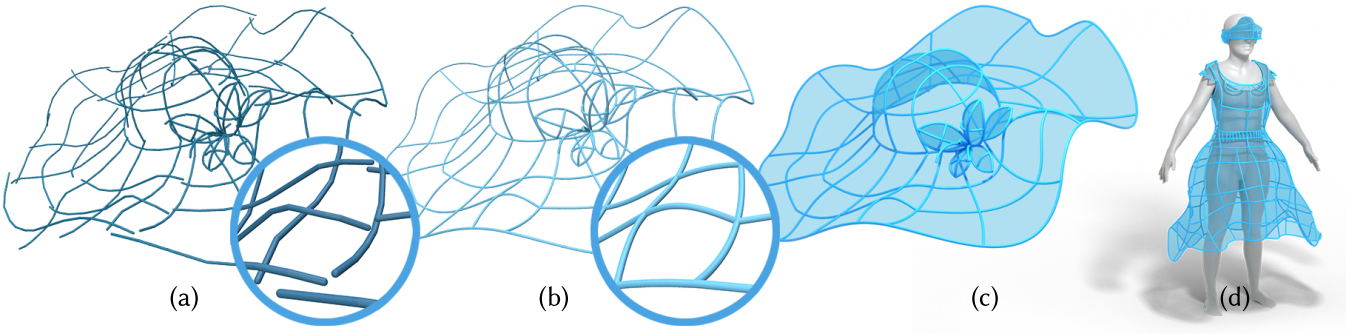

Figure 1: Freehand 3D sketching in AR/VR allows rapid conceptualization of design ideas (a). However, 3D inputs are prone to large inaccuracies (a, inset) and sketches cannot be utilized in downstream design pipelines. Our novel 3D sketching system, CASSIE, allows the creation of clean, well-connected 3D curve networks by performing automatic stroke neatening (b). These curve networks are augmented by our on-the-fly cycle detection and surfacing method (c) which improves shape perception by providing occlusion cues. We evaluated CASSIE with 12 users and utilized it for creating 3D concepts for a variety of application domains (d).

ABSTRACT

We present CASSIE, a conceptual modeling system in VR that leverages freehand mid-air sketching, and a novel 3D optimization framework to create connected curve network *armatures*, predictively surfaced using *patches* with C^0 continuity. Our system provides a judicious balance of interactivity and automation, providing a homogeneous 3D drawing interface for a mix of freehand curves, curve networks, and surface patches. Our system encourages and aids users in drawing consistent networks of curves, easing the transition from freehand ideation to concept modeling. A comprehensive user study with professional designers as well as amateurs (N=12), and a diverse gallery of 3D models, show our armature and patch functionality to offer a user experience and expressivity on par with freehand ideation, while creating sophisticated concept models for downstream applications.

CHI '21, May 8–13, 2021, Yokohama, Japan

CCS CONCEPTS

• **Computing methodologies** → **Virtual reality**; **Shape modeling**; • **Human-centered computing** → *Interactive systems and tools.*

KEYWORDS

curve networks, immersive design, ideation, concept modeling

ACM Reference Format:
Emilie Yu, Rahul Arora, Tibor Stanko, J. Andreas Bærentzen, Karan Singh, and Adrien Bousseau. 2021. CASSIE: Curve and Surface Sketching in Immersive Environments. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3411764.3445158

1 INTRODUCTION

The digital 3D realization of a mental design construct typically progresses from exploratory *ideation* to a *concept model* which is subsequently processed for presentation, structural analysis, or manufacturing [18, 44]. Freehand sketching, traditionally on pen and paper, dominates the ideation process. A number of digital 2D drawing interfaces support the creation [7, 16, 41] and exploration [2] of ideation sketches. These sketches serve as a visual

reference for concept 3D CAD modeling [6], though recent works like Analytic Scaffolding [50] and True2Form [60] have shown that 2D design sketches can be lifted into 3D curve networks, effectively bridging 2D interfaces for design sketching and 3D concept modeling.

In the immersive environments of Augmented and Virtual Realities (AR/VR), a sketch-pen becomes a magic wand, allowing users to step into their creations and draw directly in 3D. Popular VR applications like Tilt Brush [23] and Quill [20], that facilitate direct 3D creation and viewing of designs at diverse scales (even in-situ for AR [3]), suggest that sketching in immersive environments has the potential to completely disrupt 3D ideation. Our system *CASSIE* allows *freehand* 3D ideation, as well as automatic optimization of 3D strokes to create curve network *armatures* and predictively surfaces *patches*, thus enabling both ideation and concept modeling through a single interface.

Existing research and commercial software for 3D modeling in VR is based on disparate metaphors of free-form sculpting [20], CAD-like curve and surface creation [54], curve networks [59], swept surfaces [14], hybrid 2D/3D model creation [3], and even coarse-to-fine poly modeling [24]. In contrast, CASSIE is the first general ideation and concept modeling system that progressively transforms mid-air sketch strokes into patched 3D curve networks. Admittedly, the distinction between 3D ideation and concept models becomes subtle in VR when both are produced via a sketching tool such as CASSIE. In the subsequent text, we will use the following definitions:

– An *ideation* sketch serves as an externalization of a mentally evolving design. In our system such sketches consist of independent strokes from which a surface cannot trivially be inferred.
– A *concept model* expresses the design as a well-defined shape that designers use to communicate their idea to others. It loses part of the ambiguity present in ideation, thereby gaining structure which enables refinement and reuse. In our system, concept models consist of connected curves which form a network that aids us in computing meaningful surface patches.

Inspired by interactive sketch beautification [22, 31], we automatically neaten strokes as they are drawn to detect and enforce intersections with nearby curves and 3D grid points, tangent continuity, axis-alignment, and planarity—features that are common in man-made objects [60]. Uniformly applying multiple and potentially conflicting sketch constraints can cause neatened strokes to deviate from their design intent. We thus cast the selection of these criteria as selective 3D optimization, balancing geometric constraint satisfaction with fidelity to the sketched stroke. We further propose a real-time algorithm, inspired by Stanko et al. [55], to progressively construct surface patches across predicted cycles of network curves. The resulting surfaces serve a triple purpose: they incrementally bolster the 3D shape providing important depth and occlusion cues [5]; they serve as virtual canvases on which to draw or project additional detail curves; and they act as an ongoing incentive to draw consistent curve networks, naturally guiding the user from ambiguous ideation curves towards a well-defined 3D concept model.

While CASSIE embodies functionality to support both 3D ideation and conceptual modeling, we formally evaluate its salient features as three independent systems: *freehand* mid-air drawing with minimal and independent smoothing of 3D user strokes; *armature*, which enables overall stroke optimization to aid the creation of precise curve networks; and *patch* which further enables the automatic detection and surfacing of appropriate curve network cycles. We conducted a within-subjects study of the 3 systems with 12 users (design professionals or enthusiasts). The study validated the functional need for all aspects of CASSIE: *freehand* sketching for ideation, and *armature* and *patch* for concept modeling. Critically, the study showed that the complexity and constraints of our concept modeling functionality were achieved with an agency and overall user experience comparable to unconstrained ideation sketching. While our study data shows strong support for freehand ideative exploration (Fig. 10), it equally validates our hypothesis that intermingled curve network and surface creation is valuable and encourages the creation of consistent 3D conceptual models.

Some of our professional study participants were keen to continue experimenting with CASSIE after the study, and we show a gallery of compelling and diverse 3D concept models created by a variety of users (Fig. 12). We also demonstrate the suitability of these models for downstream sculpting, engineering, and fabrication applications (Fig. 14).

Our **contributions** are thus threefold: we present a homogeneous ideation and concept modeling system for design sketching in VR, based on a novel 3D curve and surface optimization framework; we report on a detailed comparative evaluation between ideation and conceptual modeling in VR, with 12 users; we provide a corpus of 3D stroke data for future design analysis and data-driven mid-air sketch processing.

2 RELATED WORK

We start by discussing related work on immersive sketching, before discussing methods for sketch beautification, sketch-based modeling, and surfacing of curve networks.

Immersive sketching. While our system targets modern head-mounted displays, it relates to early efforts on supporting 3D drawing and painting with a variety of virtual reality devices. The seminal HoloSketch allows the creation of 3D shapes with geometric primitives and sweep surfaces [14]. Observing that it can be challenging to position and scale 3D primitives accurately, the author included an option to snap them to a 3D grid, a solution that we also adopt. Surface Drawing [49] offers similar modeling capabilities by sweeping a strip of surface along the trajectory of the user's hand. Users of this system created 3D paintings by covering the intended surface with dense surface strips. A similar practice is often observed with modern systems like Tilt Brush [23], where users accumulate a large number of strokes to paint surfaces in VR. In contrast, we target the creation of lightweight sketches composed of a sparse network of curves, which convey 3D surfaces with an economy of means. These networks, along with their surfaced patches, can be readily exported to downstream 3D software rather than requiring dedicated post-processing as is the case for dense 3D paintings [46].

Sketching in VR brings new challenges, compared to traditional 2D sketching. Arora et al. [5] and Machuca et al. [39] observed poor mid-air sketching accuracy even when users sketched simple

strokes such as straight lines and circles. While the former studied drawing such strokes in isolation, the latter evaluated precision and aesthetic quality of shapes made of grid-aligned lines and circles. Compared to sketching on paper, the lack of a supporting surface and the additional degrees of freedom induce the need for finer motor control to properly position strokes in 3D space. The lack of depth cues in sparse sketches also contributes to the approximate positioning of strokes relative to each other [5]. This imprecision can be extremely frustrating, especially for artists transitioning from 2D sketching to AR/VR [28]. We address these challenges by automatically correcting imprecise strokes and by visualizing surfaces in-between strokes.

Our work is closer in spirit to FreeDrawer [59], where users first sketch a sparse network of feature curves, and then manually indicate curve cycles to form surface patches. Spacedesign [21] implements a similar concept but sketching is constrained to 2D curves projected onto virtual or physical planes. While we share the design philosophy of FreeDrawer and Spacedesign, we present a novel 3D optimization framework for curve network creation, as well as an algorithm for progressive, automatic surface patch creation, on current VR hardware. More importantly, we report on a comprehensive user study with design professionals and amateurs to evaluate the impact of these curve network and surface features, for ideation and concept modeling in AR/VR.

Several other solutions have been proposed to overcome the challenges of VR sketching. The 3-Draw system [47] decouples the act of drawing the curve from the act of positioning it with respect to other curves. Keefe et al. [34] use haptic feedback from a Phantom device and a two-hand interaction metaphor inspired by tape drawing [8] to separate drawing the curve from indicating its tangent directions. Other two-handed systems simulate deformable rods that users bend to model shape outlines and sharp surface features [58], or rely on a physical strip of sensors that provides bending and twisting controls on 3D curves [26]. Other methods avoid or limit the use of 3D freehand sketching in the creation process. Jackson and Keefe [32] use curves from a 2D sketch as a basis for VR creation, a feature also available in the commercial software Gravity Sketch [54] along with other curve editing operations based on control points. Strokes in Gravity Sketch however beautiful, do not form a connected curve network, and thus require dedicated user intervention to create surfaces (see Section 7.1 for a comparison). Arora et al. [3] and Drey et al. [17] propose to use a 2D tablet on which the artist sketches precise strokes that are then mapped to a proxy 3D surface defined by a few freehand 3D strokes or the extrusion of a stroke. In a similar spirit, Kim et al. [35] capture hand motion to describe 3D scaffolding surfaces on which 2D strokes are projected. In contrast, our system offers a direct sketching workflow where users draw free-form 3D curves with their dominant hand, and only use the other hand for navigation.

Finally, Machuca et al. [38] share our goal of beautifying freehand 3D strokes, which they achieve by automatically detecting potential geometric relationships with existing strokes. Their approach is limited however, to planar strokes snapped at stroke endpoints, resulting in regular shapes dominated by straight or circular strokes lying on perpendicular or parallel planes. Similarly, De Araùjo et al. [12] beautify planar strokes drawn in a semi-immersive environment by snapping endpoints to form closed contours and by enforcing parallelism, perpendicularity, and equal lengths detected with thresholds. The beautified strokes are then extruded in space to form regular 3D shapes. Our curve networks, in contrast, are generic, supporting highly non-planar curves with many potential intersections (Fig. 1).

In summary, CASSIE is the first integrated VR system which allows for both 3D ideation and concept modeling through a single, fluid sketch-based interface.

2D sketch regularization and sketch-based modeling. Our approach to 3D stroke neatening is inspired by sketch beautification and regularization frameworks originally developed to process 2D diagrams and drawings created with a computer mouse. Pavlidis and Van Wyk [43] introduced one of the first such systems, that takes a line drawing as input and outputs a drawing close to the input while satisfying geometric constraints. While this system was applied as a post-process on complete drawings, Igarashi et al. [31] proposed an interactive beautification method that treats each new stroke as it is sketched. This iterative approach can cope with more complex sketches and geometric relationships between strokes. The interactivity of the system also gives more control to the user who can choose between multiple possible beautified results. More recently, Fišer et al. [22] proposed an interactive method called ShipShape, that supports Bézier curves as input. We draw inspiration from their approach, although we focus on a small set of geometric constraints tailored to the most frequent sensorimotor errors we observed in 3D sketching, while they consider a larger variety of rules to beautify 2D drawings. In addition, while Fišer et al. [22] beautify each stroke by applying geometric rules in sequence, we cast stroke neatening and structuring as an energy minimization that balances the concurrent application of multiple geometric constraints with preservation of the input curve. This formulation is inspired by sketch-based modeling algorithms that seek to lift 2D strokes to 3D by enforcing geometric constraints, while making sure that the resulting curves re-project well on the input drawing [13, 50, 51, 60]. In particular, the interactive system by Schmidt et al. [50] lifts each new stroke by snapping it to the 3D curves inferred from previous strokes, which they achieve by balancing the satisfaction of snapping constraints with re-projection error.

Surfacing of 3D curve networks. Surfaced curve networks form a compact and descriptive representation of 3D shapes [25], which has motivated the development of algorithms to automatically generate surfaces from sparse, designer-drawn 3D networks. A first challenge is to identify, among all closed cycles in the curve network, which cycles bound surface patches rather than internal cross-sections. Treating this problem globally is difficult due to the large search space of all possible curve cycles in a network, and due to the inherent ambiguity of the task [1, 48, 61]. Other approaches rely on assumptions about the network topology; for example, Orbay and Kara [40] assume that the curves form a connected graph. We instead leverage the iterative nature of our workflow to surface the network progressively, only performing a local search for cycles around each newly-added stroke. Our interactive context also allows us to let users add or remove cycles that might have been misinterpreted by our automatic algorithm. A second challenge is to generate the surface geometry that interpolates the cycle boundaries. We adopt the method by Zou et al. [62] for this task,

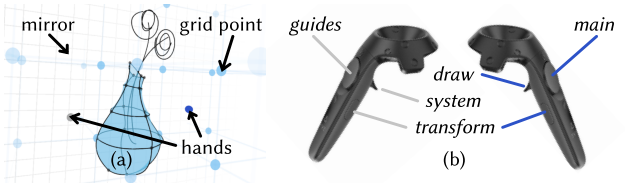

Figure 2: Our minimal interface is designed for simplicity, allowing the user to focus on creation (a). In a similar spirit, the controls are kept minimal and easily generalize to varied hardware (b). Here we show the controls mapped to an HTC Vive controller.

which applies a dynamic programming algorithm to generate a triangulation that satisfies various geometric criteria. Alternative solutions include the generation of a quad mesh aligned with the input curves [9]. Finally, various surface fairing algorithms have been proposed to improve the quality of the generated surfaces [42, 55], and could be applied to our results in a post-process.

3 USER INTERFACE AND WORKFLOW

We first present the drawing interface offered by our system and the workflow it enables. We describe the technical components necessary to achieve this workflow—stroke neatening and surfacing—in Sections 4 and 5 respectively.

3.1 Interface elements

We endeavoured to keep the interface minimal and user interactions simple, to keep the user focused on the creative task at hand. As a secondary objective, we only assumed standard interface features, common across modern VR setups, to ease remote evaluation on a variety of consumer VR hardware.

Fig. 2a illustrates our drawing interface as seen in the VR headset. In addition to the 3D shape being created, our interface visualizes the ambient workspace with an axis-aligned 3D grid for better depth perception. The workspace also includes an optional mirror plane, which greatly facilitates the design of symmetric objects. Users interact with the system using two 6-DoF (degree of freedom) controllers, each of which must provide three push-buttons. Fig. 2b shows all the controls on the HTC Vive controller, but the interactions can be similarly mapped to other common devices.

Users can draw strokes using the *draw* button. We represent each stroke as a smooth cubic poly-Bézier curve or a straight line segment, from which we remove unintended "hooks" at extremities as done by Liu et al. [37]. Double-clicking the *main* button on the dominant-hand controller is used to delete selected curves or surface patches. Selection is implicit, based on proximity to the curve/surface. Single-clicking the *main* button is used to manually create surface patches where automatic detection fails. Holding the *transform* button on the dominant or non-dominant controller allows uniformly scaling or rigidly transforming the workspace, respectively. Clicking or double-clicking the *guides* button on the non-dominant controller toggles the grid or the mirror plane, respectively. Finally, users can completely disable the automatic curve regularization and surfacing by clicking the *system* button, and use the same button to toggle the predictive features on again. Note that the functionalities attributed to the *draw* and *transform* buttons are very similar to commercial VR applications [19, 23], and therefore, users can quickly grasp their function.

Visible interface elements are similarly designed to be simple and unintrusive (see Fig. 2a). Both the controllers are rendered as small spheres, differentiated by colour. The grid is rendered as semi-transparent lines and spheres. Curves are rendered in a visually-dominant black, while nodes of the curve network (intersections) are shown as faint spheres. The latter serves as an important visual confirmation of a successful connection. Patches are shown as a faint blue to provide occlusion, thus improving depth perception [5, Figure 13], without obstructing important design elements or breaking the creative flow. Finally, implicit selections are indicated by a color change. We do not provide features to edit the strokes or surfaces once sketched, it is only possible to delete elements, once they are created.

3.2 User workflow

Fig. 3 illustrates a typical concept modeling session with our system. For every newly-drawn stroke, our system automatically identifies regularization criteria with respect to already-created 3D curves, as well as to the ambient 3D grid. In particular, the system aligns the new stroke, in position as well as tangent, to nearby curves to ease the creation of long curves (Fig. 3a–c) and well-connected curve networks (Fig. 3d–f). The system also snaps strokes to nearby grid nodes, facilitating the use of the grid as a precise scale reference.

For every new curve added to the network, our system searches for closed cycles in its vicinity to form surface patches or to modify existing ones (Fig. 3c–e). These patches greatly contribute to the perception of the 3D shape by occluding background curves. They can also serve as a sketching support, on which users can draw surface details or connect extruding parts (Fig. 3f). While the surface patches are found automatically, users can delete any patch to create holes, or add a patch where the automatic algorithm might have missed one due to ambiguity.

From the user perspective, the curve network and associated surface patches behave like a soap film, which the user shapes progressively by adding one curve at a time to form complete concept models (Fig. 3f). Note that strokes on which we detect no constraint to apply are left unchanged. Alternatively, users can also toggle all the predictive features off to fall back to a freehand sketching workflow more suitable to preliminary ideation.

4 CREATING CURVE NETWORKS

Designers often depict 3D shapes by sketching networks of curves running along surface boundaries and lines of curvature [18, 25]. These curves aid the creation of 3D meshes in downstream 3D modeling [56], and several algorithms exist to automatically surface 3D curve networks [9, 48, 61]. Unfortunately, unlike 2D sketching, 3D user strokes rarely intersect perfectly, making freehand drawing of 3D curve networks in VR difficult. We facilitate the creation of such curve networks by automatically detecting and enforcing proximal curve intersections. Complex drawings contain cluttered regions however, where enforcing all such candidate intersections for a single stroke, can distort the stroke dramatically. Inspired by related

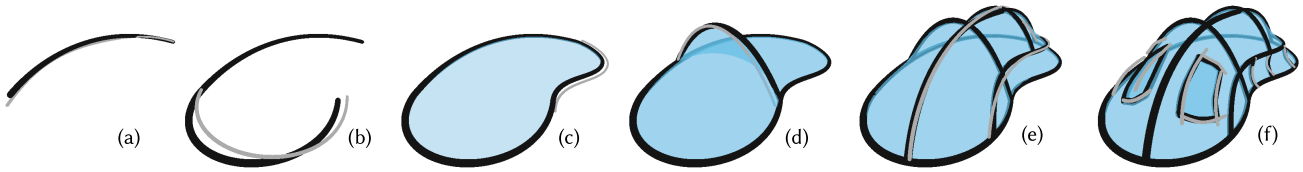

Figure 3: Typical workflow to create a 3D concept model with CASSIE. User-sketched strokes are shown in gray, and neatened strokes in black. Our system automatically connects user strokes as they are drawn to form long curves (a–c) and curve networks (d–e). We detect closed cycles in this network to form surface patches (c–e), which improves depth perception, supports the creation of surface details (f), and results in models ready for downstream applications. See also a video of 3 full sketching sessions in Supplemental Material.

work on 2D sketch beautification [22, 31] and sketch-based modeling [50, 60], we propose a mechanism to connect each new 3D stroke to as many nearby 3D curves as possible (discrete hard constraints) while staying close to its original trajectory, and compensating for drawing inaccuracy by favoring geometric features: planarity of curves and tangent continuity (continuous soft constraints). Solving such a problem is a mix of a potentially exponential search for the optimal subset of discrete hard constraints to satisfy, and function minimization for the continuous soft constraints.

4.1 Algorithm overview

Our solution has three steps. Given a sketched 3D stroke, we first find all candidate hard intersection constraints using the distance between points on the stroke and existing 3D curves. We then find an approximately optimal subset of discrete constraints using a greedy linear search, and formulate a least squares minimization for the continuous constraints. Our overall algorithm is thus efficient, robust, and works well in practice without any noticeable lag.

We next describe our algorithm in a bottom-up fashion. We first detail our 3D curve optimization that minimizes soft constraints and remains as close as possible to the input stroke, while satisfying a prescribed subset of hard constraints (Section 4.2). We then describe our strategy to select this optimal subset (Section 4.3), and finally detail how the various terms of the optimization are formulated (Section 4.4). We explain how we treat the simpler case of straight lines in supplemental material.

4.2 Enforcing intersections via continuous optimization

Let us first assume that we have selected a set of intersections $\mathcal{I}$ between the user stroke and existing curves in the 3D sketch. We associate each intersection with a constraint $c_{i\in\mathcal{I}}$ that needs to be satisfied for the user stroke to intersect the corresponding curve exactly. However, the input stroke needs to be deformed to satisfy these constraints. We measure the amount of deformation with an energy that we denote E_{fidelity}. In addition, we also seek to encourage regularization criteria such as curve planarity and tangential alignment to intersected curves, which we also express as energy terms E_{planar} and E_{tangent}. We express these criteria as soft energy terms rather than hard constraints because they correspond to aesthetic properties that are desirable but that can be balanced against other desiderata. In contrast, we need intersections to be satisfied exactly to form a well-connected curve network. Finally, we also ensure that the deformed stroke preserves the original G^1 continuity between successive curve segments, which we express as a set of constraints g_k. Given an input stroke S, we compute the deformed stroke S' that satisfies all constraints while minimizing deformation and best satisfying other criteria by solving the *continuous* optimization

$$\begin{aligned} \min_{S'} \quad & E_{\text{fidelity}}(S,S') + E_{\text{planar}}(S') + \sum_{i\in\mathcal{I}} E_{\text{tangent}}(S',i), \\ \text{s.t.} \quad & c_{i\in\mathcal{I}}(S') = 0 \\ \text{and} \quad & g_k(S') = 0. \end{aligned} \tag{1}$$

Recall that our strokes are represented using cubic Bézier segments. We formulate the soft constraints as quadratic energy functions of the Bézier control points, and hard constraints as linear functions, allowing us to solve this constrained least squares problem as a single linear solve as detailed in supplemental material.

4.3 Selecting intersections via discrete optimization

New strokes often run near multiple curves, especially on complex drawings with high stroke density. Our next challenge is to identify which of these curves should intersect the input stroke. On the one hand, we would like the curve network to be as connected as possible. On the other hand, we don't want to deviate too much from the original user intent. We cast these competing objectives as two terms in a *discrete* optimization, where we model the activation of each constraint c_i with a binary variable b_i set to 1 if the intersection is selected and to 0 otherwise

$$\min_{b_{i\in\mathcal{I}}} \lambda E_{\text{fidelity}}\big(S,S'(b)\big) + (1-\lambda)E_{\text{connectivity}}(b) \tag{2}$$

where the deformed stroke $S'(b)$ is computed using Equation 1 and the selected intersections, E_{fidelity} measures the deviation of S' from the input stroke, $E_{\text{connectivity}}$ is an energy term that decreases as more intersections are selected, and $\lambda = 0.6$ is a parameter that balances the two objectives.

Unfortunately, finding the optimal subset would require solving Equation 1 for all $2^{|\mathcal{I}|}$ combinations of binary variables b_i, which is impractical. Yet, we observed that the detected intersections are most often valid, and only a few superfluous ones need to be rejected. This observation motivated us to minimize Equation 2 using a greedy strategy, where we first select all intersections, and then test all subsets obtained by removing one intersection. If the

best of these subsets yields a lower energy than the full set, we keep it and repeat the process by removing another intersection, until the energy doesn't decrease anymore. Fig. 4 illustrates the result of this selection mechanism on a 2D curve, and on typical 3D curve configurations. The number of subsets that must be tested for each stroke is determined by the total number of detected constraints $|\mathcal{I}|$, and the number of rejected constraints $|\mathcal{I}_r|$ as: $1 + (|\mathcal{I}_r| + 1)(2|\mathcal{I}| - |\mathcal{I}_r|)/2$. The efficiency of our greedy strategy was validated during a subsequent user study (see section 6), where we counted an average of 5 subsets tested for each stroke, with a median of only 2 subsets. In practice, even this count is an upper bound as we further prune the number of subsets tested when some constraints are deemed incompatible, such as when a point is constrained to multiple positions.

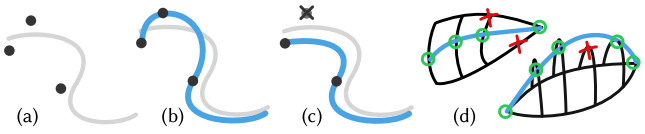

Figure 4: An input stroke with all detected potential intersections (a). Enforcing all intersections distorts the curve (b). Enforcing the optimal subset keeps the curve close to the input (c). Typical configurations where too many intersections are detected (d): in areas with high stroke density (top) or when the input stroke is smooth but the candidate intersections do not form a smooth path (bottom). Our algorithm selects a subset of intersections (circled) and rejects others (crossed out) to preserve the shape of the original curve (blue).

4.4 Implementation details

We now describe how we detect and express the constraints and energy terms that form Equation 1 and Equation 2. We refer the interested reader to supplemental material for additional implementation details.

Interactive neatening. Since we target an interactive sketching application, our method optimizes each stroke as soon as it is sketched, in the context of previously-drawn curves. We do not revisit the strokes that have already been optimized, as this would turn neatening into a global optimization problem that would become increasingly costly as more strokes are added.

Distance and angular thresholds. Many of our regularization constraints and objectives depend on a notion of proximity between curves, in space and/or angle. In addition, we wish to make the sensitivity of our beautification process adaptive to the drawing scale, such that the beautification is less aggressive when users zoom-in to draw details. We achieve this goal by defining a spatial proximity threshold $\delta(s) = \frac{\delta^1}{s}$, where s denotes the scale of the drawing space and $\delta^1 = 4$cm was fixed experimentally for appropriate beautification. In the initial zoomed out state, $s = 1$.

We set the angular threshold to $\theta = \frac{\pi}{6}$, under which we consider that two lines or curves are parallel.

Intersections. We detect intersections between the input stroke and any nearby curve, grid nodes or the mirror plane by testing if the brush tip comes within a radius $r_{\text{proximity}} = \delta(s)$ of such elements while sketching. We also check if the two endpoints of the stroke are within a distance $r_{\text{proximity}}$ of each other to form closed loops. We enforce an intersection by inserting a control point at the closest point of the new curve and constraining it to be on the other curve or grid node.

Tangent alignment. At each intersection, we compare the tangents between the two curves. If the angle between both tangents is under the angular threshold θ, we encourage the tangent of the new curve to align with the other one using an energy term E_{tangent} that measures the norm of their cross product. Otherwise, we set E_{tangent} to zero.

Planarity. We first compute the best-fit plane to the control points by least squares. If the distance from the plane to the farthest control point is below the threshold $r_{\text{proximity}}$, we encourage all control points to lie in the plane using an energy E_{planar} that minimizes the dot product between the plane normal and the vectors formed by pairs of successive control points. We also test whether the plane normal is within an angle θ to one of the three orthogonal grid directions, in which case we snap the plane to be orthogonal to that direction. For non-planar curves, we set E_{planar} to zero.

Fidelity to input. Our fidelity energy E_{fidelity} measures deviation between the beautified curve and the input one. Following Xu et al. [60], we measure both the deviation in absolute position of the control points and the deviation in the slope of pairs of successive control points, the latter term penalizing variations in the overall shape of the curve.

Connectivity. The goal of $E_{\text{connectivity}}$ is to favor the selection of as many intersections as possible, as long as they do not overly deform the input. We model it with an exponential that increases as fewer constraints are selected

$$E_{\text{connectivity}}(b) = e^{-\left(\frac{\sum_{i\in\mathcal{I}} b_i w_i}{\sum_{i\in\mathcal{I}} w_i}\right)^2}, \tag{3}$$

where w_i denotes the weight associated to intersection i, and b_i equals 1 if the intersection is selected, 0 otherwise. Following Schmidt et al. [50], we vary this weight depending on the nature of the intersection, being equal to 1 if the curve intersects a grid node or the mirror plane along its trajectory, 1.25 if the curve intersects a grid node at its endpoint, 1.5 if the curve intersects another curve, and 2 if the curve passes through an existing intersection.

Sharp features. We support strokes with sharp features by representing them with poly-Bézier curves without G^1 continuity at the junctions between successive Bézier segments. For such junctions, we do not enforce the hard constraint g_k from Equation 1. We empirically determine that an input stroke intends to represent a sharp junction if we detect a strong variation of tangent directions ($> 45°$, as in [46]).

5 SURFACING

Our stroke neatening and structuring algorithm greatly facilitates the creation of well-connected curve networks (Fig. 5), which in turn enables the automatic detection of curve cycles (Fig. 5b) bounding

surface patches (Fig. 5a). We first describe how we identify these cycles before detailing how we generate the surface they bound and how we exploit these surface patches for drawing on-surface details.

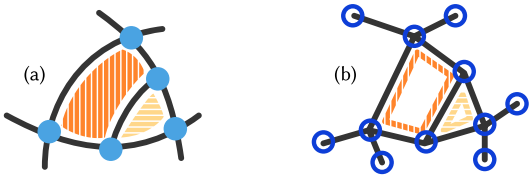

Figure 5: After enforcing intersections (section 4), we obtain a result such as in (a). The sketch is composed of curves that meet at intersections. The sparse strokes represent surface patches (bright and light orange) that we create automatically. To do so, we reason on a curve network representation of the sketch (b) where each curve is represented by one or multiple segments connected at nodes. On this curve network we search for cycles corresponding to the patches.

5.1 Cycle detection

A number of existing cycle detection algorithms perform a global optimization over the complete curve network [1, 61]. Unfortunately, the space of cycles of a given curve network is very large, making a global search computationally demanding and unnecessarily repetitive in an interactive system. Instead, we exploit the fact that after the user has sketched a new stroke, any new patches should only be created for cycles in parts of the network affected by the stroke. We thus apply a local search strategy, which is more suitable for real-time, incremental construction of the sketch. Our approach is inspired by the algorithm of Stanko et al. [55], which relies on the surface normal at each intersection to sort all segments incident to that intersection, such that cycles can be formed by walking around the curve network in a fixed order as illustrated in Fig. 6.

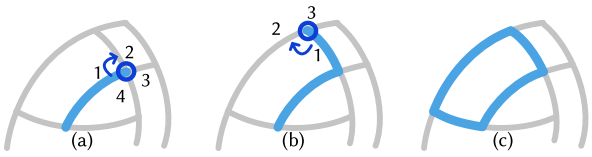

Figure 6: Local cycle detection. We sort all segments incident to an intersection according to the surface normal at that intersection (a). We walk around the cycle by selecting at each node the next segment in clock-wise order (b), until we obtain a closed cycle (c).

However, while Stanko et al. [55] were reconstructing real-world surfaces for which the normal was measured, we need to infer the surface normal from the input curves. We obtain this normal estimate at each intersection by assuming that the intersecting curves lie on a smooth surface, whose tangent plane is spanned by the curve tangents. Yet, this tangent plane only defines the normal direction, not its orientation. We remove this sign ambiguity at the scale of a cycle by comparing normal orientations between successive nodes. Along a curve, we can compute a family of vectors by parallel-transporting the normal from one of its nodes [27]. These vectors are a good approximation of the normals of the underlying surface if the curve lies on this surface without significant geodesic torsion, which is the case for curvature lines that form a large part of designer-drawn curves [30, 42, 51]. Based on this observation, parallel-transporting the normal $\vec{N}_A$ from node A to node B gives a vector $\vec{N}_{A\|B}$ that should closely match the direction of normal $\vec{N}_B$. If the two vectors point in opposite directions, we flip $\vec{N}_B$ before proceeding with the next node along the cycle, as illustrated in Fig. 7.

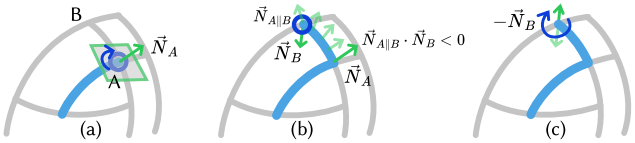

Figure 7: Local normal orientation. We compute the surface normal $\vec{N}_A$ at node A by fitting a plane on the tangents of incident curve segments (a). We parallel-transport this normal to node B to obtain $\vec{N}_{A\|B}$, which we compare to $\vec{N}_B$ (b). Since the two vectors point to opposite directions, we flip $\vec{N}_B$ before proceeding (c).

We apply this cycle detection on both sides of each new curve segment, taking care of removing any existing cycle when a curve is drawn across it. We also run the cycle detection on every segment that intersects user-deleted curves to update the surface after such edits.

While simple and efficient, the above algorithm is limited to smooth surfaces for which the tangent plane is well-defined at each curve intersection. If a node lies on a sharp surface feature, such as the corner of a cube, then several surface normals exist at that node (Fig. 8). We detect such cases by checking whether some curve tangents at this node lie far from the best fit tangent plane, by assessing the dot products of the tangents $\{\vec{t}_i\}_i$ with the plane normal $\vec{n}_P$. We consider that a node is on a sharp feature if: $\max_i \vec{n}_P \cdot \vec{t}_i > \cos(\frac{\pi}{2} - \theta)$, with θ the angular threshold defined in Section 4.4. On encountering such a node, we rely on the last well-defined normal along the cycle, which we transport to the node to sort its segments. We also take care of excluding the segments that do not lie close to the plane defined by this normal. In the rare case where the starting node of a cycle is itself sharp, we first resort to the normal of the best-fit plane to sort the segments incident to that node, and then for each successive pair of segments we look for a cycle with starting normal defined as the cross product between these two segments.

Nevertheless, our automatic algorithm sometimes fails to detect cycles, especially on sparse curve networks as in Fig. 9a. Furthermore, our assumption that the curves exhibit little geodesic torsion does not always hold. While adding new curves to form a denser network often resolves these issues, we also allow users to explicitly trigger the creation of a surface patch by clicking near the center of the patch they envision (Fig. 9b). We then detect the closest stroke

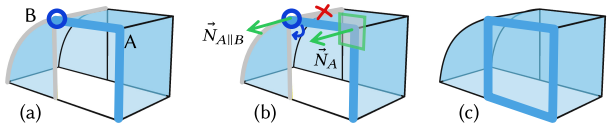

Figure 8: Dealing with sharp surface features. The surface normal is ill-defined at node B, as one of the incident segments does not lie in the plane formed by the other ones (a). We exclude the segment that is not in the plane defined by the parallel-transported normal $\vec{N_{A\|B}}$ (b, crossed out) and select the next segment in clockwise order around $\vec{N_{A\|B}}$ among the remaining options to form the cycle (c). In cases where the starting node A is also ill-defined, we compute the surface normal $\vec{N_A}$ as the cross-product of two successive segments (b, blue) among the segments at A sorted according to the normal of the best-fit tangent plane.

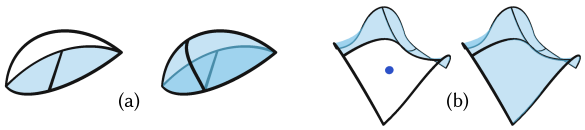

Figure 9: The automatic cycle detection might fail. Adding strokes resolves the issue by creating additional intersections (a). It is also possible to manually add a patch by clicking (b).

segment to the click and walk along the curve network by always selecting the segment that is closest to the click, as measured by comparing distances from the click to the closest point on each segment. We also allow users to delete unwanted surface patches, for instance to create holes.

5.2 Surface geometry

Once a cycle is detected, we generate a triangle mesh over it with the method by Zou et al. [62], using their public implementation. We then refine the generated surface by applying isotropic re-meshing followed by curvature-flow smoothing [15, 33] to generate minimal (soap-film) surfaces. Both steps are performed using CGAL [57].

5.3 Sketching on surfaces

The surface patches generated by our method not only enhance the perception of the created shape, they can also serve the role of a canvas on which users can draw additional details. We consider that a stroke should lie on a surface patch if a) all of its control points are within a distance $r_{\text{proximity}}$ from that patch; and b) the curve does not split the cycle bounding the patch. When these criteria are fulfilled, we project all control points of the curve onto the surface patch so that the user stroke appears to stick to the object.

6 USER EXPERIENCE STUDY

We formally evaluated the novel features of CASSIE (*armature*, *patch*), using *freehand* as a baseline, by re-configuring CASSIE as three isolated sub-systems with a common interface. Our study goals were twofold. One, evaluating whether the user experience achieved with stroke neatening and surfacing is comparable to free-form sketching, making CASSIE a homogeneous sketching interface capable of supporting ideation as well as more refined concept modeling. Two, evaluating the effectiveness of our algorithms in neatening the user strokes and creating well-connected models. Specifically, our study participants tried and evaluated three versions of the tool: a baseline *freehand* version which only contained mid-air sketching features, an *armature* version which implemented our curve neatening and structuring featureset, and a *patch* version which also included the automatic and manual surfacing features, and the ability to draw on surface patches. We performed this study with expert and novice users who are all familiar with VR interfaces in general.

In addition to this formal study, in Section 7.1 we position CASSIE relative to commercial 3D VR modeling software, and further report on an informal comparison of CASSIE vis-à-vis Gravity Sketch [54].

6.1 Participants

We recruited 12 participants (2 female) aged 25–50 for the study. Six of the participants were professional artists or designers, while the others had at least basic experience in CAD modelling or creating design sketches as part of their jobs in graphics, HCI, or related fields and as hobbyists. All but two had used VR for over a year, and most (9/12) had utilized VR for creative tasks, using Tilt Brush, Medium, Masterpiece VR, Blocks, AnimVR, or research prototypes. Participants were paid approximately US $22 for their time, converted to their currency of choice.

6.2 Apparatus

Owing to the restrictions on in-person user studies due to the COVID-19 pandemic, the study sessions were conducted remotely, using participants' own VR setups. As a result, participants used a variety of VR devices, including Oculus Rift and Rift S, HTC Vive, Vive Pro, and Vive Pro Eye. During the study session, an experimenter was available via video-conferencing for answering participant questions.

6.3 Procedure

Participants first used the three systems to draw a computer mouse. These untimed sessions were utilized as tutorial sessions, intended to let the participants familiarize themselves with the interface. Then, the participants used the three systems in the same order as the tutorial for timed sessions, which were later utilized for qualitative and quantitative analyses. For each system, participants were given 5 minutes to design a desk lamp followed by another 5 minutes to create a running shoe. To reduce the influence of learning effects, the order of the systems was counterbalanced between participants using a Latin square design. It is important to note that participants were not constrained to draw the same shape with all the three systems. Instead, we asked them to work with the capabilities and constraints of each system to design the optimal object.

Each participant was sent a set of instructions a day before their study slot (see supplemental), informing them of the nature of the research, the objects they were expected to draw, and the

Table 1: Creative Support Index (CSI) [10] scores (out of 20, higher is better) for each of the 3 systems (mean and standard deviation). Statistically significant differences are marked with asterisks (*). These scores put *armature* and *patch* on par with the *freehand* system, showing that the additional mental load imposed by these two systems did not cause a noticeable drop in users' creative potential.

	Freehand	Armature	Patch
Enjoyment	16.8 (2.7)	15.5 (2.4)	16.2 (2.7)
Exploration	15.0* (4.3)	13.1* (4.4)	13.5 (4.3)
Expressiveness	16.8 (2.9)	14.6 (3.2)	14.6 (3.6)
Immersion	16.6 (2.4)	15.5 (2.3)	15.2 (3.5)
Results Worth Effort	15.1 (3.5)	16.1 (1.9)	15.6 (3.0)
CSI	16.1	15.0	15.0

expected drawing style of a sparse curve network. We hoped that this preparatory material helped participants think ahead and plan their designs in advance, thus focusing on concept modeling in addition to pure ideation. However, participants were not required to do so.

The study session started with participants signing an informed consent form, followed by them watching a detailed instructions video (11 min, see supplemental material), and filling out a demographics questionnaire. After evaluating each system, participants took a break and filled a short questionnaire for that system. Finally, participants filled a more detailed questionnaire comparing the three systems.

6.4 Results

Fig. 10 shows the 3D sketches and models created by the six participants who declared to be professional artists. We provide the creations of the six other participants as supplemental material.

Most participants created shapes of similar complexity with all three systems (65 strokes on average for *freehand*, 44 for *armature*, 42 for *patch*, with no statistically significant difference between the three), although some participants took advantage of the *freehand* system to achieve a more sketchy style with significant over-sketching (P5). Under close inspection, most *freehand* sketches exhibit over-shooting and gaps at stroke ends, which are mostly absent in the drawings created with the *armature* and *patch* systems. Finally, several of the models created with the *patch* system contain spurious holes in the 3D surface, which are either due to disconnected curves in the sketched network, or to failure of the automatic cycle detection algorithm.

Fig. 11 shows two drawings created in the *armature* system next to the raw strokes before snapping, which illustrates the approximate user inputs and the net effect of stroke neatening, structuring, and snapping.

We next discuss participants' feedback about their experience with the three systems, and then perform quantitative analysis of the curve networks they created.

6.4.1 User feedback. Users evaluated the degree of creative support provided by each system via the Creative Support Index (CSI) questionnaire [10]. We skipped the collaboration part of the questionnaire, as we did not build any collaborative tools in our systems. The summary results are shown in Table 1. The three systems fare almost evenly, most differences being statistically insignificant ($p > .05$ on a repeated-measures ANOVA), except for *Exploration* where the *freehand* system outperforms the two others.

Participants commented that each system offered support for a slightly different portion of the *ideation–concept design continuum*: "Each has its own uses. Freehand is good for quickly sketching up an idea as you are not restricted by the system. Armature is good for refining a design..." (P11) , "Armature seems [to be] a way to express volumes for more technical goals, but sketching [freehand] expresses an idea." (P1) Some wanted to be able to quickly switch back-and-forth between the systems, a feature we implemented after the study: "[*Armature*] handles noise coming from the input motion and signal and creates a much nicer looking stroke. It would be nice to be able to turn off snapping however, to allow strokes to be drawn closer together." (P3) In particular, automatic curve neatening reaches its limits on detailed sketches: "...predictive neatening was useful to help me quickly block out the initial shape, and snapping to form continuous lines was helpful since I was able to break up a complex stroke into smaller ones. But when the sketch became more complex and I had to draw finer/smaller strokes, it often "over-neatened" my strokes." (P7)

Still, to compare the three systems, participants were asked to rate the value of the *armature* features compared to *freehand*, and the value of the *patch* system compared to *armature* on 5-point Likert scales (5 is highest). Most participants agreed that *armature* features added to the *freehand* system (6×5, 6×3, **median 4**) and the *patch* features were a valuable addition over *armature* (5×5, 4×4, 2×3, 1×2, **median 4**). Participants felt that the curve network *armatures* encouraged them to think more about the object's use in downstream processes: "...with armature you can see the beautified sketch closer to a final draft." (P8) , "In the freehand method, I find I'm more inclined to do a messy sketch model rather than a more thought out one when using the armature system." (P4)

The surface *patches* further bolstered this idea, motivating the creation of solid, usable, real-world shapes: "The patch system let [sic] you understand the geometry to create solid object necessary for many projects in an organic way. As for armature you can't see this right away taking an extra step in the process." (P8) , "The patch system visually assisted my understanding of the model, [it] seemed to give me a better sense of its volume." (P9) Similarly, P2 commented that *patch* "allows users to define [a] surface without many lines which can clutter a design and possibly de-emphasise important features that need line defininition [sic]."

However, the incentive to create well-connected curve networks in the *patch* system was also experienced as a constraint by some: "I found that using the patch system actually distracted me with trying to get the patching correct over the act of just sketching out what I was thinking about." (P4) , "I felt that I had to change the way I worked to accommodate the system. For example, I would have to add more strokes to get the patches to show up, which meant that I deviate from my initial intent, or create a messier/busier looking drawing than I would have liked." (P7) Nevertheless, all the participants felt that they could derive value by incorporating the *armature* and *patch* functionalities in their typical workflow.

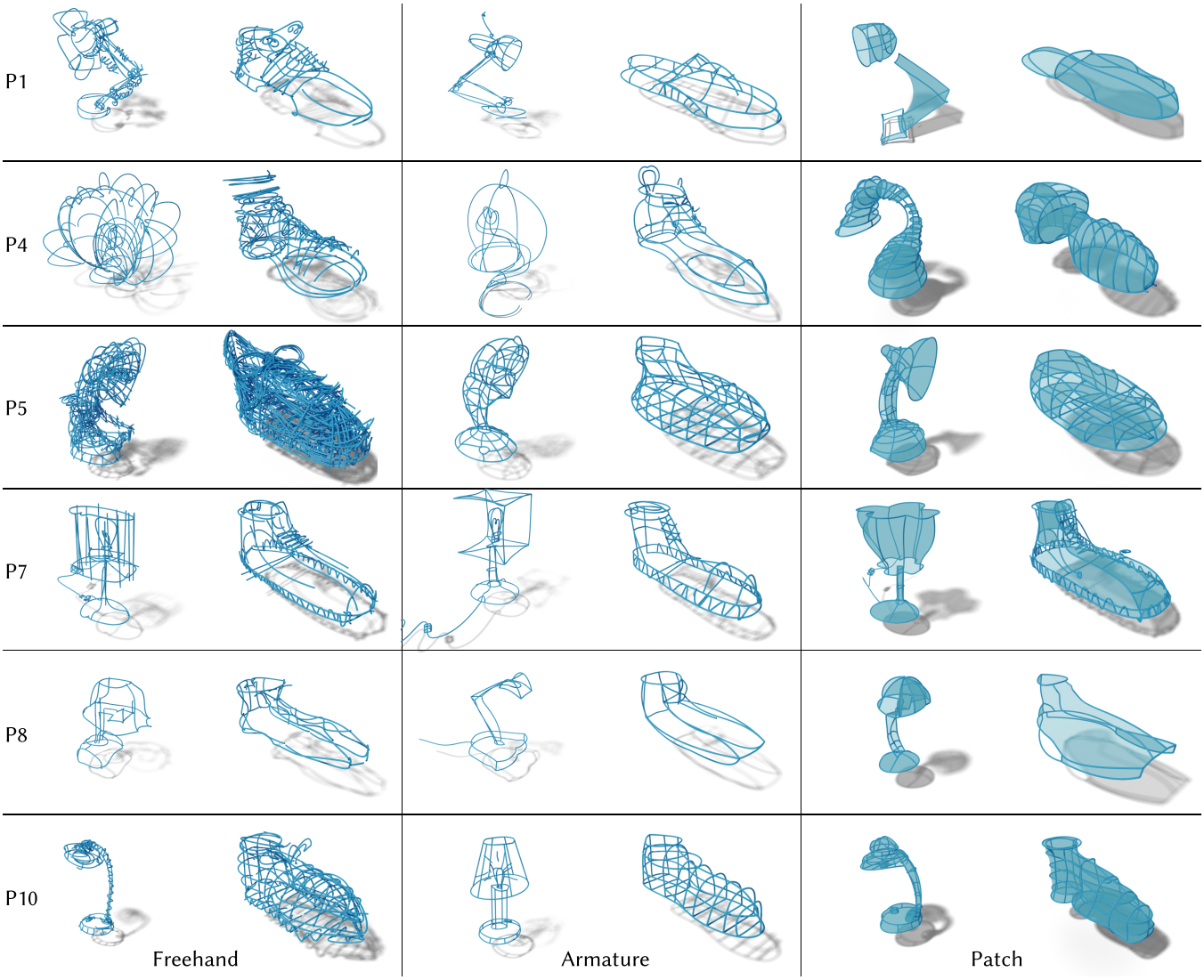

Figure 10: Designs created by the six professional participants in the study using all three systems. We provide the results created by amateurs as supplemental material, although we didn't observe strong differences between the two groups.

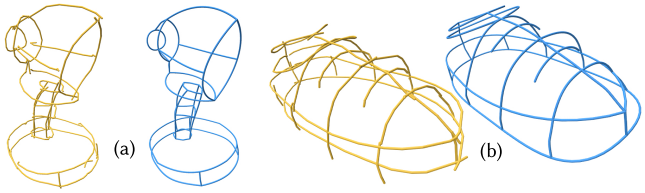

Figure 11: Input strokes (yellow) and beautified result (blue) for sketches by P12 (a) and P9 (b). Notice how our beautification creates well-connected curve networks while deviating minimally from the original user strokes. Images rendered using Polyscope [52].

For instance, P10 remarked that the systems were ideal for conceptualization: "Could be a great way to define rough shapes before bringing them into a desktop app to refine."

6.4.2 Quantitative analysis. Our goal when designing CASSIE was to support users in creating well-connected curve networks that can be surfaced for downstream applications. While visual comparison between *freehand* sketches and *armatures* show that the latter are much more connected, we hypothesize that the *patch* system provides additional incentive to create connected networks by rewarding participants when they do so. We evaluate this hypothesis by computing, for both *armature* and *patch* curve networks, the ratio of endpoint (dangling) nodes out of all nodes. We find out that the curve networks created with *patch* contain fewer endpoint nodes (μ = 9.8%, σ = 6.8%) compared to armature (μ = 14.9%, σ = 11.4%),

meaning that they are better connected. The difference is statistically significant ($t = 2.50, p = .020$ on a paired t-test). The variance of this metric is also lower with the *patch* system, suggesting that more participants tended to sketch well-connected networks with *patch*.

While the *armature* and *patch* systems attempt to detect curve intersections automatically, they might sometimes miss intended intersections, or create unintended ones. When this is the case, users typically delete the stroke and redraw a new version that better reflects their goal. In contrast, users of the *freehand* system might correct for erroneous strokes simply by over-sketching duplicate strokes. We quantify these different strategies by counting the ratio of deleted strokes for each system: *freehand* ($\mu = 27.2\%,\ \sigma = 14.1\%$), *armature* ($\mu = 36.3\%,\ \sigma = 12.5\%$), *patch* ($\mu = 39.5\%,\ \sigma = 13.3\%$). A repeated-measures ANOVA reveals that users deleted significantly less strokes in the *freehand* system ($F_{2,22} = 6.93, p = .005$).

7 RESULTS AND APPLICATIONS

Fig. 12 provides a gallery of models created with CASSIE by several trained users, including a professional study participant (P1) and 3 different authors. The sketches cover diverse application domains such as product design (a,b,e,g), architecture (c,d), and character design (f). Note how VR sketching facilitates design in context, as illustrated with the VR headsets drawn using a 3D head as an underlay to achieve accurate proportions and contacts. Most sketches were done in short sessions of 5–20 minutes, while the 1:1 scale car design (3.2m in length) took an hour and 40 minutes.

7.1 Comparisons to commercial VR modeling software and 2D sketch-based modeling

A focused user study comparing new research prototypes like CASSIE, to commercial software like Gravity Sketch [54], or CAD tools (Sketchup, Blender) is difficult for multiple reasons: CAD modeling is fundamentally different from CASSIE in workflow and interface; Gravity Sketch also uses disparate workflows for curve ideation and concept surface creation; mature commercial systems have many additional controls like stroke type, color, and polished renders that can confound participants' perception of creative support; and several of our participants already had extensive experience with commercial systems, while they were all entirely unfamiliar with CASSIE. As Cherry and Latulipe, the creators of the Creative Support Index [10], point out, "the key is to ensure that only one of the factors (tool, task, and expertise of participant group) varies at a time; otherwise, comparison of the CSI score between two different treatments will be difficult to decipher."

We did however, qualitatively compare our system to Gravity Sketch [54], a commercial software that allows the creation of 3D curves and surfaces in VR. A major difference between this system and ours is that Gravity Sketch provides completely independent facilities for curve and surface creation that are switched between using a modal interface. As a result, users typically first lay down the salient feature curves of the model before draping these curves with surface patches. Since the curves are drawn without explicit connectivity, users need to explicitly define every single surface patch to be created, as illustrated in Fig. 13 that shows a sequence of modeling steps extracted from an online tutorial (https://youtu.be/ymCe5C_IlF4). In contrast, users of CASSIE can quickly create a similar model by seamlessly drawing the feature curves over the inferred surface patches.

We qualitatively extend this comparison to desktop-based CAD modeling tools by looking at the perceived advantages of CASSIE from participants with varied experience with those tools. P1 particularly liked how CASSIE's fluid interface contrasted with their prior experience with Gravity Sketch and desktop CAD tools, remarking verbally: "I need the most direct tool possible in order to maintain a state of creative flow. The tool shouldn't obstruct that, which is what I find to be an issue in desktop CAD tools and Gravity Sketch where the interface presents many different modes and tools to me. In contrast, [CASSIE's] sketching interface helps me express this creative momentum." P3, who rated themselves as a beginner with CAD modeling (2/5 for prior experience in CAD modeling), appreciated the fact that CASSIE allows for "quick generation of sketches in VR without cumbersome tooling and commands." Similarly, participants with extensive CAD experience remarked that CASSIE is well suited to "quickly express a 3D concept to colleagues" (P4) or for "fast modeling prototyping" (P8) . P5, who experimented with a mixed workflow with ZBrush (Fig. 14c), appreciated the use of CASSIE to build a rough shape as it enabled them to "step into the model and create the curves and shapes I was thinking." Still, Gravity Sketch and other mature modeling tools provide a wide range of precise editing features which were missed in CASSIE by some participants (P1, P10), thus pointing out an interesting avenue of future work in combining both manual and assisted creation paradigms.

Another comparison can be drawn to sketch-based modeling tools that *lift* 2D sketches into 3D. While CASSIE adopts a fluid, natural interface, such tools typically require designers to either follow a strict creation style [50] or provide additional manual annotations [60]. Moreover, layered or geometrically complex 3D objects—such as the hat (Fig. 1c), buildings (Fig. 12c–d), or HMDs (Fig. 12e)—that do not have a single descriptive 2D viewpoint [18] can be extremely cumbersome or impossible to design using purely 2D sketch-based modeling tools.

7.2 Downstream processing

3D models created in CASSIE are readily usable in downstream software, as illustrated in Fig. 14.the following proofs-of-concept. Using existing remeshing tools [11, 29], CASSIE patches can be merged to form a single manifold surface, which can then be 3D printed (Fig. 14a). Our surfaced models can also be used for engineering applications; for instance, for creating volumetric trusses (Fig. 14b). Finally, CASSIE models can be refined in commercial sculpting applications (Fig. 14c) or colored and shaded for presentation (Fig. 14d).

8 CONCLUSION AND FUTURE WORK

Ideation and concept modeling are the foundation of 3D design. In current practice, 2D sketches dominate ideation, and disparate CAD-like tools produce 3D concept models. Our system CASSIE showcases the ability of AR/VR to effectively support both ideation

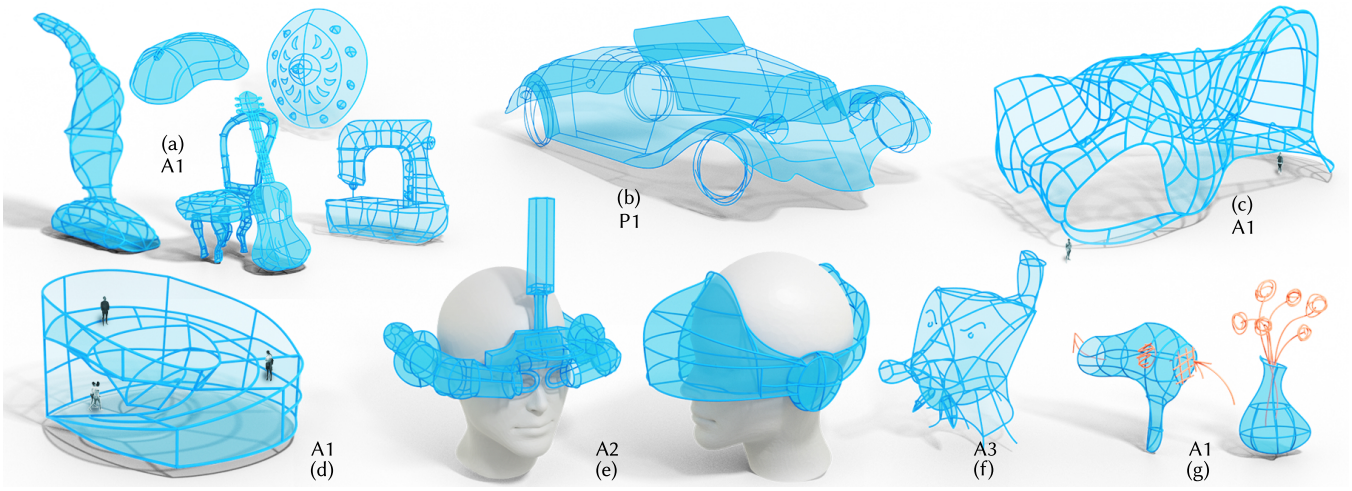

Figure 12: A gallery of 3D sketches created with CASSIE, labelled with the creator (A: author, P: participant). While most designs made use of the *armature* and *patch* features, some also included freehand strokes (orange) to suggest less definite parts (h).

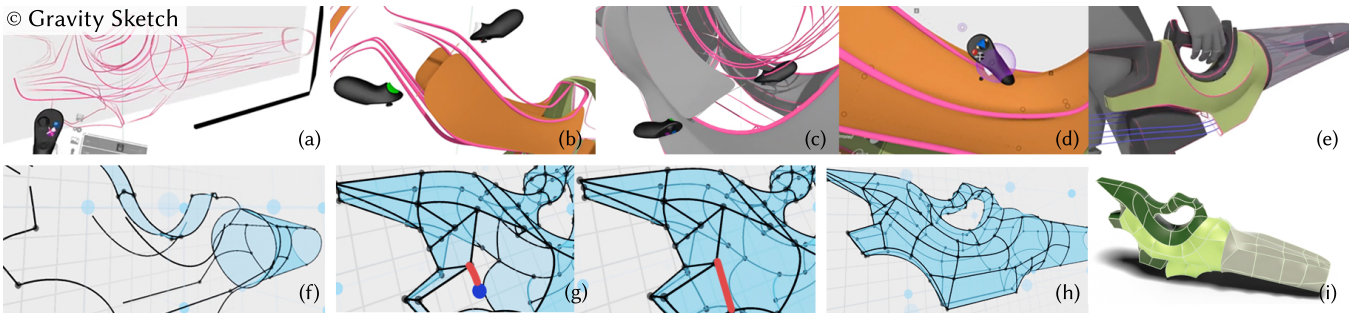

Figure 13: Comparing design workflows in Gravity Sketch [54] (top) and CASSIE (bottom). In Gravity Sketch, the user starts with a loose sketch composed of strokes with no explicit connectivity (a). Adding surfaces comes at a later stage, either by bridging nearby curves (b) or by roughly aligning the surface to the strokes (c). The user can then refine the surfaces via control points. With CASSIE, the user simultaneously sketches well-connected curve networks (f) and surfaces automatically appear as a cycle is closed (g) by a new stroke (red). The user can refine the surfaces by adding strokes that directly control the shape of the surface (h). The final result is a leaf blower (e,i), which we colored and rendered in Blender. A video version of this comparison is provided in the supplementals.

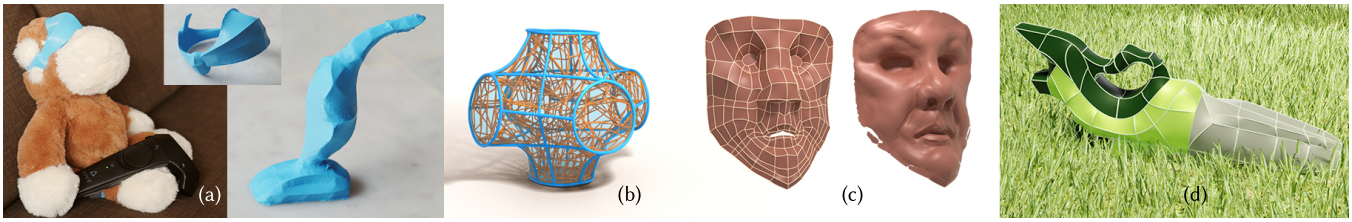

Figure 14: Applications: (a) 3D printed HMD from Fig. 12f and vacuum cleaner from Fig. 12a, (b) structural analysis and truss generation using Arora et al. [4], (c) detailed face model sculpted by P5 by sketching in CASSIE (left) and then refining patches in ZBrush [45] (right), (d) colored and shaded render to use as communication material.

and concept modeling in a shared immersive space. CASSIE combines the fluidity of freehand sketching with geometric and aesthetic constraints of 3D modeling, to predictively neaten, structure, and surface user strokes into 3D models of sufficient quality for downstream applications. Importantly, our study reveals that such progressive structuring and surfacing is not detrimental to user agency, is seen by some users as an aid in creating and perceiving a solid 3D shape, and acts as an incentive to draw well-connected 3D concept models.

Limitations. While automatic neatening greatly eases the creation of connected curve networks, our current implementation based on a single proximity threshold was judged too intrusive by some participants. Several strategies could be explored to provide finer control on neatening, such as adapting the threshold to stroke speed or pressure, or learning a user-specific threshold by analyzing a few annotated freehand drawings of that user. Additionally, exploring interactive solutions to edit and manually tune the neatening in ambiguous cases, as well as combining manual deformation and edition of strokes [7, 40, 54, 58] with automatic neatening are promising avenues of research to enhance user agency in the neatening process. Surface generation could also be improved, for instance by better reproducing the curvature depicted by the armature [42], or by proposing surface editing interactions based on the sketched strokes [36, 53]. In order to bootstrap these efforts, we openly release the interaction data from the 72 sketches in our study as well as the 16 other sketches depicted in the paper, including raw user strokes as well as the neatened curves[1]. We also release the CASSIE source code for academic research[2].

Future Work. While our current prototype allows users to switch between creative exploration and more precise modeling simply by enabling neatening and surfacing, we see several avenues to achieve a continuum between these two forms of sketching. On the one hand, freehand strokes could be used as underlays to trace more definite armatures, and as extra guidance in our optimization to best position the neatened curves and surface patches. On the other hand, sparse armatures could be used as structured spatial deformers to warp denser freehand strokes, allowing designers to explore design variations while retaining the original look of their ideation sketches. The AR/VR setting could also be leveraged for sketching in situ, as suggested with the headsets sketched over a head in Fig. 12. In this usage scenario, the contextual 3D models could be used as extra cues for stroke beautification and surface generation.

We believe that mid-air drawing in immersive environments has the potential to significantly disrupt the interactive 3D design process, and CASSIE is a positive step in that direction.

ACKNOWLEDGMENTS

We thank our study participants for their time and effort, and the anonymous reviewers for their helpful comments. This work was supported by ERC Starting Grant D3 (ERC-2016-STG 714221), NSERC Discovery Grant 480538, and by software and research donations from Adobe.

[1] https://gitlab.inria.fr/D3/cassie-data
[2] https://gitlab.inria.fr/D3/cassie

REFERENCES

[1] Fatemeh Abbasinejad, Pushkar Joshi, and Nina Amenta. 2011. Surface patches from unorganized space curves. *Computer Graphics Forum* 30, 5 (2011), 1379–1387. https://doi.org/10.1111/j.1467-8659.2011.02012.x
[2] Rahul Arora, Ishan Darolia, Vinay P Namboodiri, Karan Singh, and Adrien Bousseau. 2017. Sketchsoup: Exploratory Ideation using Design Sketches. *Computer Graphics Forum* 36, 8 (2017), 302–312.
[3] Rahul Arora, Rubaiat Habib Kazi, Tovi Grossman, George Fitzmaurice, and Karan Singh. 2018. Symbiosissketch: Combining 2d & 3d sketching for designing detailed 3d objects in situ. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–15.
[4] Rahul Arora, Alec Jacobson, Timothy R. Langlois, Yijiang Huang, Caitlin Mueller, Wojciech Matusik, Ariel Shamir, Karan Singh, and David I.W. Levin. 2019. Volumetric Michell Trusses for Parametric Design & Fabrication. In *Proceedings of the 3rd ACM Symposium on Computation Fabrication* (Pittsburgh, PA, USA) *(SCF '19)*. ACM, New York, NY, USA, 13.
[5] Rahul Arora, Rubaiat Habib Kazi, Fraser Anderson, Tovi Grossman, Karan Singh, and George W Fitzmaurice. 2017. Experimental Evaluation of Sketching on Surfaces in VR.. In *CHI*, Vol. 17. ACM, New York, NY, USA, 5643–5654.
[6] Robert McNeel & Associates. 2018. Rhinoceros 3D. https://www.rhino3d.com/.
[7] Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. ACM, New York, NY, USA, 151–160.
[8] Ravin Balakrishnan, George Fitzmaurice, Gordon Kurtenbach, and William Buxton. 1999. Digital tape drawing. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*. ACM, New York, NY, USA, 161–169.
[9] Mikhail Bessmeltsev, Caoyu Wang, Alla Sheffer, and Karan Singh. 2012. Design-driven quadrangulation of closed 3D curves. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–11.
[10] Erin Cherry and Celine Latulipe. 2014. Quantifying the Creativity Support of Digital Tools through the Creativity Support Index. *ACM Trans. Comput.-Hum. Interact.* 21, 4, Article 21 (June 2014), 25 pages. https://doi.org/10.1145/2617588
[11] Paolo Cignoni, Massimiliano Corsini, and Guido Ranzuglia. 2008. MeshLab: an Open-Source 3D Mesh Processing System. *ERCIM News* 2008, 73 (2008), 47–48. http://dblp.uni-trier.de/db/journals/ercim/ercim2008.html#CignoniCR08
[12] Bruno R De Araùjo, Géry Casiez, and Joaquim A Jorge. 2012. Mockup builder: direct 3D modeling on and above the surface in a continuous interaction space. In *Proceedings of Graphics Interface 2012*. Canadian Information Processing Society, Mississauga, ON, Canada, 173–180.
[13] Chris De Paoli and Karan Singh. 2015. SecondSkin: sketch-based construction of layered 3D models. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10.
[14] Michael F Deering. 1995. HoloSketch: a virtual reality sketching/animation tool. *ACM Transactions on Computer-Human Interaction (TOCHI)* 2, 3 (1995), 220–238.
[15] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, 317–324.
[16] Julie Dorsey, Songhua Xu, Gabe Smedresman, Holly Rushmeier, and Leonard McMillan. 2007. The Mental Canvas: A Tool for Conceptual Architectural Design and Analysis. In *Proc. IEEE Pacific Conference on Computer Graphics and Applications*. IEEE, New York, NY, USA, 201–210.
[17] Tobias Drey, Jan Gugenheimer, Julian Karlbauer, Maximilian Milo, and Enrico Rukzio. 2020. VRSketchIn: Exploring the Design Space of Pen and Tablet Interaction for 3D Sketching in Virtual Reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–14.
[18] K Eissen and R Steur. 2009. *Sketching: Drawing techniques for product designers* (11 ed.). BIS Publishers, AMsterdam, The Netherlands.
[19] Facebook. 2015. Oculus Medium. https://www.oculus.com/medium/.
[20] Facebook. 2016. Oculus Quill. https://www.oculus.com/experiences/rift/1118609381580656/.
[21] Michele Fiorentino, Raffaele de Amicis, Giuseppe Monno, and Andre Stork. 2002. Spacedesign: A mixed reality workspace for aesthetic industrial design. In *Proceedings. International Symposium on Mixed and Augmented Reality*. IEEE, New York, NY, USA, 86–318. https://doi.org/10.1109/ISMAR.2002.1115077
[22] Jakub Fišer, Paul Asente, and Daniel Sỳkora. 2015. ShipShape: a drawing beautification assistant. In *Proceedings of the workshop on Sketch-Based Interfaces and Modeling*. Eurographics Association, Geneve, Switzerland, 49–57.
[23] Google. 2016. Tilt Brush. https://www.tiltbrush.com.
[24] Google. 2017. Google Blocks. https://arvr.google.com/blocks/.
[25] Giorgio Gori, Alla Sheffer, Nicholas Vining, Enrique Rosales, Nathan Carr, and Tao Ju. 2017. Flowrep: Descriptive curve networks for free-form design shapes. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–14.
[26] Tovi Grossman, Ravin Balakrishnan, and Karan Singh. 2003. An Interface for Creating and Manipulating Curves Using a High Degree-of-Freedom Curve Input Device. In *Proc. of the ACM SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 185–192.

[27] Andrew J Hanson and Hui Ma. 1995. Parallel transport approach to curve framing. *Indiana University, Techreports-TR425* 11 (1995), 3–7.

[28] Laura M. Herman and Stefanie Hutka. 2019. Virtual Artistry: Virtual Reality Translations of Two-Dimensional Creativity. In *Proceedings of the 2019 on Creativity and Cognition* (San Diego, CA, USA) *(C&C '19)*. Association for Computing Machinery, New York, NY, USA, 612–618. https://doi.org/10.1145/3325480.3326579

[29] Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 60 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201353

[30] Emmanuel Iarussi, David Bommes, and Adrien Bousseau. 2015. BendFields: Regularized Curvature Fields from Rough Concept Sketches. *ACM Trans. Graph.* 34, 3, Article 24 (May 2015), 16 pages. https://doi.org/10.1145/2710026

[31] Takeo Igarashi, Satoshi Matsuoka, Sachiko Kawachiya, and Hidehiko Tanaka. 2007. Interactive beautification: a technique for rapid geometric design. In *ACM SIGGRAPH 2007 courses*. ACM, New York, NY, USA, 18–es.

[32] Bret Jackson and Daniel F Keefe. 2016. Lift-off: Using reference imagery and freehand sketching to create 3d models in VR. *IEEE transactions on visualization and computer graphics* 22, 4 (2016), 1442–1451.

[33] Michael Kazhdan, Jake Solomon, and Mirela Ben-Chen. 2012. Can mean-curvature flow be modified to be non-singular? *Computer Graphics Forum* 31, 5 (2012), 1745–1754.

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

[35] Yongkwan Kim, Sang-Gyun An, Joon Hyub Lee, and Seok-Hyung Bae. 2018. Agile 3D sketching with air scaffolding. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–12.

[36] Jung-hoon Kwon, Han-wool Choi, Jeong-in Lee, and Young-Ho Chai. 2005. Free-Hand stroke based NURBS surface for sketching and deforming 3d contents. In *Pacific-Rim Conference on Multimedia*. Springer-Verlag, Heidelberg, Germany, 315–326. https://doi.org/10.1007/11581772_28

[37] Chenxi Liu, Enrique Rosales, and Alla Sheffer. 2018. Strokeaggregator: Consolidating raw sketches into artist-intended curve drawings. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–15.

[38] Mayra D Barrera Machuca, Paul Asente, Wolfgang Stuerzlinger, Jingwan Lu, and Byungmoon Kim. 2018. Multiplanes: Assisted freehand VR Sketching. In *Proceedings of the Symposium on Spatial User Interaction*. ACM, New York, NY, USA, 36–47.

[39] Mayra Donaji Barrera Machuca, Wolfgang Stuerzlinger, and Paul Asente. 2019. The Effect of Spatial Ability on Immersive 3D Drawing. In *Proceedings of the ACM Conference on Creativity & Cognition (C&C'19)*. ACM, New York, NY, USA, 173–186.

[40] GüNay Orbay and Levent Burak Kara. 2012. Sketch-based surface design using malleable curve networks. *Computers & Graphics* 36, 8 (2012), 916–929.

[41] Patrick Paczkowski, Min H. Kim, Yann Morvan, Julie Dorsey, Holly Rushmeier, and Carol O'Sullivan. 2011. Insitu: Sketching Architectural Designs in Context. *ACM Trans. Graph.* 30, 6 (Dec. 2011), 1–10. https://doi.org/10.1145/2070781.2024216

[42] Hao Pan, Yang Liu, Alla Sheffer, Nicholas Vining, Chang-Jian Li, and Wenping Wang. 2015. Flow aligned surfacing of curve networks. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10.

[43] Theo Pavlidis and Christopher J Van Wyk. 1985. An automatic beautifier for drawings and illustrations. *ACM SIGGRAPH Computer Graphics* 19, 3 (1985), 225–234.

[44] A. Pipes. 2007. *Drawing for Designers*. Laurence King Publishing, London, UK. https://books.google.co.nz/books?id=phgfAQAAIAAJ

[45] Pixologic. 2016. ZBrush. http://pixologic.com/features/about-zbrush.php.

[46] Enrique Rosales, Jafet Rodriguez, and Alla Sheffer. 2019. SurfaceBrush: From Virtual Reality Drawings to Manifold Surfaces. *ACM Transaction on Graphics* 38, 4, Article 96 (2019), 15 pages. https://doi.org/10.1145/3306346.3322970

[47] Emanuel Sachs, Andrew Roberts, and David Stoops. 1991. 3-Draw: A tool for designing 3D shapes. *IEEE Computer Graphics and Applications* 11, 6 (1991), 18–26.

[48] Bardia Sadri and Karan Singh. 2014. Flow-Complex-Based Shape Reconstruction from 3D Curves. *ACM Transactions on Graphics* 33, 2, Article 20 (2014), 15 pages.

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

[50] Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. 2009. Analytic drawing of 3D scaffolds. *ACM transactions on graphics (TOG)* 28, 5 (2009), 1–10.

[51] Cloud Shao, Adrien Bousseau, Alla Sheffer, and Karan Singh. 2012. CrossShade: shading concept sketches using cross-section curves. *ACM Transactions on Graphics (TOG)* 31, 4, Article 45 (2012), 11 pages.

[52] Nicholas Sharp et al. 2019. Polyscope. www.polyscope.run.

[53] Karan Singh and Eugene Fiume. 1998. Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, 405–414.

[54] Gravity Sketch. 2017. Gravity Sketch. https://www.gravitysketch.com/.

[55] Tibor Stanko, Stefanie Hahmann, Georges-Pierre Bonneau, and Nathalie Saguin-Sprynski. 2016. Smooth interpolation of curve networks with surface normals. In *EG 2016 - Short Papers*. The Eurographics Association, Geneve, Switzerland, 21–24.

[56] Kenshi Takayama, Daniele Panozzo, Alexander Sorkine-Hornung, and Olga Sorkine-Hornung. 2013. Sketch-Based Generation and Editing of Quad Meshes. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)* 32, 4 (2013), 97:1–97:8.

[57] The CGAL Project. 2020. *CGAL User and Reference Manual* (5.0.2 ed.). CGAL Editorial Board. https://doc.cgal.org/5.0.2/Manual/packages.html

[58] Floor Verhoeven and Olga Sorkine-Hornung. 2019. RodMesh: Two-handed 3D Surface Modeling in Virtual Reality. In *Proceedings of the Symposium on Vision, Modeling and Visualization (VMV)*. The Eurographics Association, Geneve, Switzerland, 10.

[59] Gerold Wesche and Hans-Peter Seidel. 2001. FreeDrawer: a free-form sketching system on the responsive workbench. In *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, New York, NY, USA, 167–174.

[60] Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. 2014. True2Form: 3D curve networks from 2D sketches via selective regularization. *ACM Transactions on Graphics* 33, 4, Article 131 (2014), 13 pages.

[61] Yixin Zhuang, Ming Zou, Nathan Carr, and Tao Ju. 2013. A general and efficient method for finding cycles in 3D curve networks. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–10.

[62] Ming Zou, Tao Ju, and Nathan Carr. 2013. An algorithm for triangulating multiple 3D polygons. *Computer Graphics Forum* 32, 5 (2013), 157–166.