ScaffoldSketch: Accurate Industrial Design Drawing in VR

Xue Yu
George Mason University
Fairfax, USA
xyu21@gmu.edu

Stephen DiVerdi
Adobe Research
San Francisco, USA
diverdi@adobe.com

Akshay Sharma
Virginia Tech
Blacksburg, USA
Iowa State University
Ames, USA
akshays@iastate.edu

Yotam Gingold
George Mason University
Fairfax, USA
ygingold@gmu.edu

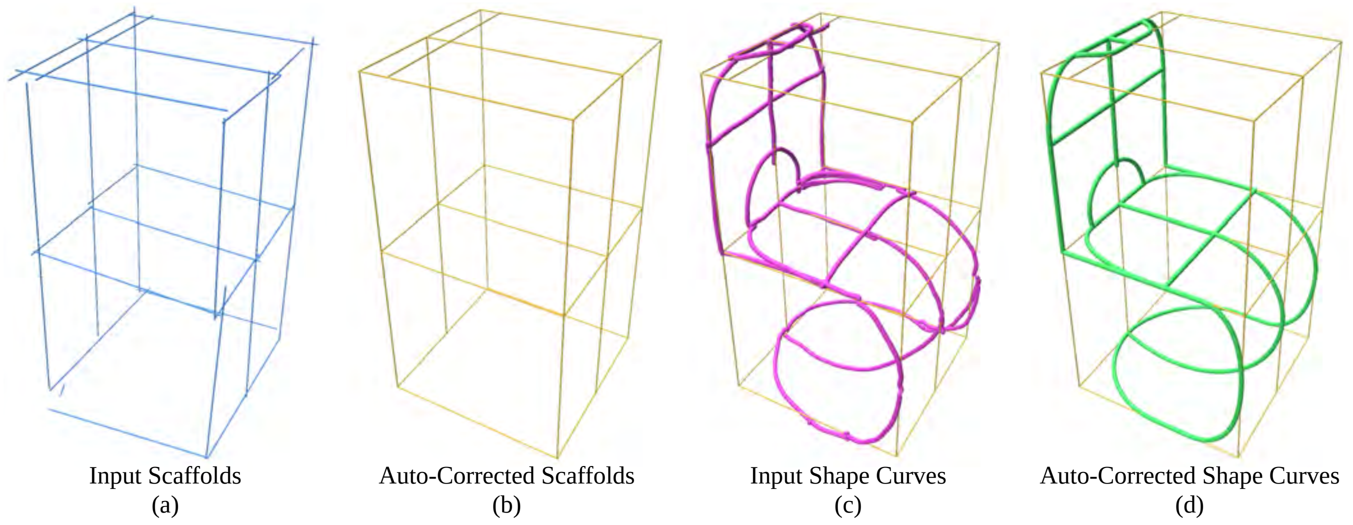

Figure 1: ScaffoldSketch user inputs and system outputs. (a) The scaffold strokes a user drew in VR. (b) The auto-corrected scaffold from the user's strokes. (c) The shape strokes the user drew in VR. (d) The auto-corrected shape from the user's strokes and the scaffold.

ABSTRACT

We present an approach to in-air design drawing based on the two-stage approach common in 2D design drawing practice. The primary challenge to 3D drawing in-air is the accuracy of users' strokes. Beautifying or auto-correcting an arbitrary drawing in 2D or 3D is challenging due to ambiguities stemming from many possible interpretations of a stroke. A similar challenge appears when drawing freehand on paper in the real world. 2D design drawing practice (as taught in industrial design school) addresses this by decomposing the process of creating realistic 2D projections of 3D shapes. Designers first create scaffold or construction lines. When drawing shape or structure curves, designers are guided by the scaffolds. Our key insight is that accurate industrial design drawing in 3D becomes tractable when decomposed into auto-correcting scaffold strokes, which have simple relationships with one another, followed by auto-correcting shape strokes with respect to the scaffold strokes. We demonstrate our approach's effectiveness with an expert study involving industrial designers.

CCS CONCEPTS

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

KEYWORDS

industrial design, virtual reality, sketching, auto-correct

ACM Reference Format:
Xue Yu, Stephen DiVerdi, Akshay Sharma, and Yotam Gingold. 2021. ScaffoldSketch: Accurate Industrial Design Drawing in VR. In *The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21), October 10–14, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3472749.3474756

1 INTRODUCTION

Industrial designers invest years learning and practicing a particular approach to drawing perspective views of 3D shapes [14, 20, 21, 26, 41, 44]. These *design drawings* are lossy projections of an often *de novo* 3D form in their mind. Designers learn to draw scaffolds to help them create accurate and aesthetic projections of 3D shapes (Figure 2). A set of construction lines serves as a scaffold for the descriptive lines that depict the intended shape. Scaffolds are straight lines that can be easily drawn accurately. Scaffolds help designers draw aesthetic shape strokes by providing designers with accurate visual references. Shape strokes pass through key points such as intersections between scaffold lines. Without scaffolds, creating globally consistent and accurate design drawings would be much more difficult.

While 2D projections can communicate approximate ideas quickly, 3D shapes are needed when communicating with stakeholders such as structural or fabrication engineers, to measure or manipulate the shape, or anytime additional viewpoints are needed. To create 3D shapes, designers learn an entirely new skillset unrelated to sketching: polygonal modeling in CAD software. This discards the knowledge and experience industrial designers spend years mastering.

We introduce **ScaffoldSketch**, an approach that allows users to create accurate sketches in 3D (using a VR headset and controllers) by leveraging a particular technique from 2D industrial design drawing (Figure 1), effectively granting industrial designers new powers they already know how to use. The challenge with existing approaches to 3D sketching is the difficulty humans have accurately drawing in space. Accuracy is significantly reduced compared to 2D sketching [3, 7, 48, 50]. This may be due to depth perception or the additional complexity in the body's motor plan, which must consider the extra free dimension not constrained by contact with a drawing surface. For example, it is easy to draw two intersecting straight lines in 2D but exceptionally difficult in 3D. Previous 3D sketching approaches either target "artistic" drawing with rough and disconnected strokes (Quill, TiltBrush, Gravity Sketch, [28, 32, 43]) or focus on new tools and algorithms for *beautifying*, *neatening*, or *auto-correcting* strokes [2, 6, 11, 12, 17, 18, 25, 27, 31, 53]. To use these approaches, designers must learn new ways to draw or accept globally inconsistent 3D sketches. Design drawings (in 2D or 3D) are too complex for general approaches to algorithmic beautification or auto-correction (hereafter auto-correction) [23, 35, 37], which must determine which strokes are intentionally straight or curved; parallel, perpendicular, or otherwise; the same length; and intersect tangentially or otherwise.

The key insight underpinning ScaffoldSketch is that scaffolds and shape strokes decompose design drawing into simpler problems, each amenable to algorithmic global auto-correct. Designers learn this decomposition as one of many techniques from industrial design education. This decomposition was used for 2D-to-3D inference by Schmidt et al. [44]. We employ this decomposition for direct 3D drawing, where accuracy is the dominant challenge. When considering an entire 3D drawing at once, the scaffold lines and shape strokes together are too complex for global algorithmic auto-correct. Separately, each stage leads to a highly constrained, solvable algorithmic auto-correct problem. Real-time algorithmic global auto-correct is tractable for 3D scaffold lines, since they are straight lines with a small set of known inter-relationships. Given auto-corrected scaffold lines, algorithmic auto-correct of shape strokes becomes tractable, since each shape stroke must pass through key points of the scaffold. Accurate scaffold lines also provide visual reference when designers draw 3D shape strokes, preventing compound errors. An example of the user input scaffold lines and shape curves, and the output auto-corrected strokes, can be seen in Figure 1, which makes it clear that ScaffoldSketch enables users to create accurate and expressive 3D drawings in VR with a particular design drawing style that can be part of many different artistic and professional workflows.

2 RELATED WORK

We consider previous work in the areas of direct 3D drawing and 2D-to-3D drawing.

Direct 3D Drawing. A variety of approaches have been proposed for drawing in 3D. Quill and TiltBrush are recent VR drawing systems for artistic 3D drawing. They do not perform auto-correct, and so output a set of disconnected, rough strokes. In contrast, our goal is to generate accurate curves in 3D for industrial design applications. Gravity Sketch is also a recent VR shape creation system targeting industrial design. Users can create freeform 3D strokes which are not algorithmically beautified or manipulate traditional CAD primitives, such as Bézier control point positions in space. There is no algorithmic auto-correct beyond snapping to a grid or snapping control points to each other. In contrast, we create accurate 3D curves by auto-correcting a 3D analog of 2D design drawing practice.

Schkolne et al. [43] introduced an early approach to in-air 3D shape creation with custom hardware that digitized a user's hand pose or tool. The novelty of this system was the input hardware. They did not aim to correct inaccurate user input. Keefe et al. [25] proposed several interactions for drawing smoother or more controllable individual strokes. These interactions do not consider global information, such as intersections and tangencies between strokes. In contrast, we auto-correct each drawn stroke with global information from all previously-drawn strokes.

Kim et al. [27, 28] proposed techniques for creating freeform 3D scaffolds based on designers' hands. In contrast, our approach does not support freeform scaffolds. Instead, we target existing 2D design drawing practices with the goal of creating accurate 3D drawings.

Multiplanes [31] and CASSIE [53] snap curve endpoints to existing planes, curves, or grid points based on spatial proximity. Multiplanes displays context-sensitive guides (e.g., planes). This snapping has the effect of neatening the drawn curves, but it is limited to the particular supported snapping relationships. CASSIE [53] creates well-connected 3D curve networks as we do. Drawn curves are neatened via an optimization process that considers planarity and snapping endpoints and tangents to other curves or a regular grid. CASSIE further automatically detects surface patches and lets users draw on them. We do not explore surfacing in ScaffoldSketch. In contrast, we explore accurate 3D curve drawing via a user-created scaffolding structure. This results in curves that respect global relationships such as lengths and angles, allowing users to create, for example, symmetric shapes. Smart3DGuides [32] aimed to improve accuracy purely by displaying appropriate visual guides, without directly snapping or otherwise auto-correcting user input. In our approach, designers explicitly create their own scaffold "guides" in a familiar and accurate manner based on their 2D drawing experience, by drawing lines and placing tick marks. Because users create their own scaffolds, ScaffoldSketch is able to infer user intent more accurately. Arora et al. [2] and Drey et al. [12] allow users to specify how their tablet's 2D surface maps to a 3D surface. Users then draw more precise 2D strokes on the tablet surface. The strokes are projected into 3D with the specified map. In contrast, ScaffoldSketch users draw directly in 3D and rely on auto-correct to create accurate 3D strokes. This eliminates many mode switches.

2D-to-3D Drawing. Many approaches considered the problem of lifting 2D sketched strokes into 3D. While these approaches do not require VR or 3D tracking equipment, 2D-to-3D lifting approaches must contend with the ambiguous nature of 2D projections of 3D forms. They require designers to learn new drawing approaches [5, 13, 24, 44], accommodate simplifying assumptions and limitations [1, 19, 52], or spend time annotating existing drawings [16, 49]. The most closely related works to ours are by Schmidt et al. [44] and Gryaditskaya et al. [19], which lift industrial design drawings from 2D to 3D interactively or as a post-process, respectively. They both distinguish between straight scaffold lines and smooth shape curves, with a primary focus on interpreting depth ambiguities such as accidental 2D intersections and perspective projections. Gryaditskaya et al. [19] aims to lift each individual stroke, no matter how roughly drawn, whereas Schmidt et al. [44] performs algorithmic auto-correct as we do. Our interface allows designers to draw directly in 3D. This avoids projection ambiguities but presents a different set of algorithmic auto-correct challenges. To auto-correct scaffold lines, we must solve a problem common to many beautification systems, finding a non-conflicting subset of satisfiable constraints. Our iteratively re-weighted least squares (IRLS) approach to constraint satisfaction provides a simple, unified solution to what is often solved via a complex decision tree [15].

3 INTERFACE

The design of the ScaffoldSketch interface is informed by established conventions of sketching for industrial design.

3.1 Industrial Design Practice

Industrial designers conventionally learn a particular approach to drawing 3D shapes in perspective [14, 20, 21, 26, 41] among other skills. This approach is based on drawing straight lines, often called construction or scaffold lines, which make up a scaffold for later curves depicting the shape of the 3D form (Figure 2). The approach is typically drawn free-hand. Construction lines include lines for establishing vanishing points for accurate perspective project. Construction lines also include scaffold lines (following the taxonomy in Gryaditskaya et al. [20]). Like the scaffolds used in the physical construction of a building, scaffold lines typically form a set of interrelated lines which enclose or otherwise support drawing the lines and curves which depict the shape itself. Scaffold lines are often parallel, perpendicular, intersecting, and co-planar. Scaffold lines may share the same length, such as the opposite edges of a box. Other approaches to form creation, which may be preferred by some industrial designers, are out of scope for ScaffoldSketch. We based our interface for accurate drawing in VR on this technique. This makes our interface immediately approachable by industrial designers who have learned to explore form development through design drawing. The 3D sketches ScaffoldSketch creates could be seen as a final result (if the designer's goal is an initial form or sketch) or as a step towards a rendering. ScaffoldSketch creates similar output to other industrial-design-inspired 3D modeling approaches [44, 52, 53].

3.2 The ScaffoldSketch Interface

Our VR sketching environment ScaffoldSketch is designed to translate the existing 2D design drawing process into 3D. Therefore, we have exactly two modes corresponding to the two types of lines, SCAFFOLD and SHAPE. We use different auto-correct algorithms in the different modes. Since we are mimicking a traditionally pen-and-paper drawing process, our interface is minimalistic (Figure 2). The VR environment is white like paper with light grid lines visible on the walls. Our approach to auto-correct makes no assumptions regarding world-space axes.

In SCAFFOLD mode, the VR controller creates an elastic line stretched between the controller trigger's press and release (Figure 3a). After a scaffold line is drawn, it is auto-corrected to more accurately align with other scaffold lines (Figure 3b). Scaffold lines are thin and gray, since they are drawn as light strokes in pen-and-paper. By default, tick marks are shown at the start, end, and middle of a scaffold line, which we call "keypoints." Shape strokes begin, end, and pass through these keypoints. Users can add additional keypoints by drawing an extremely short scaffold line across an existing scaffold line. Designers often add keypoints when drawing, for example, a rounded corner.

Shape strokes describe the intended form. We draw them as thick, black lines, since they are drawn as dark strokes in pen and paper. In SHAPE mode, the VR controller creates a curve directly from the controller movements between the trigger's press and release (Figure 3c). After a shape stroke is drawn, it is auto-corrected to be smooth while passing through and tangent to the scaffold lines (Figure 3d).

Besides the two drawing modes, our interface has buttons for undo/redo and zooming in/out. Zooming is important for user accuracy. A user's accuracy in the physical world is limited by their physical abilities. By shrinking themselves, a user improves their accuracy in the virtual world. Since our auto-correct thresholds are based on physical measurements in user coordinates, zooming is necessary to, for example, prevent undesirable snapping. Our system auto-saves, though a feature exists to export the entire drawing history. Text labels annotate controller buttons whenever the user brings the controller close to their face.

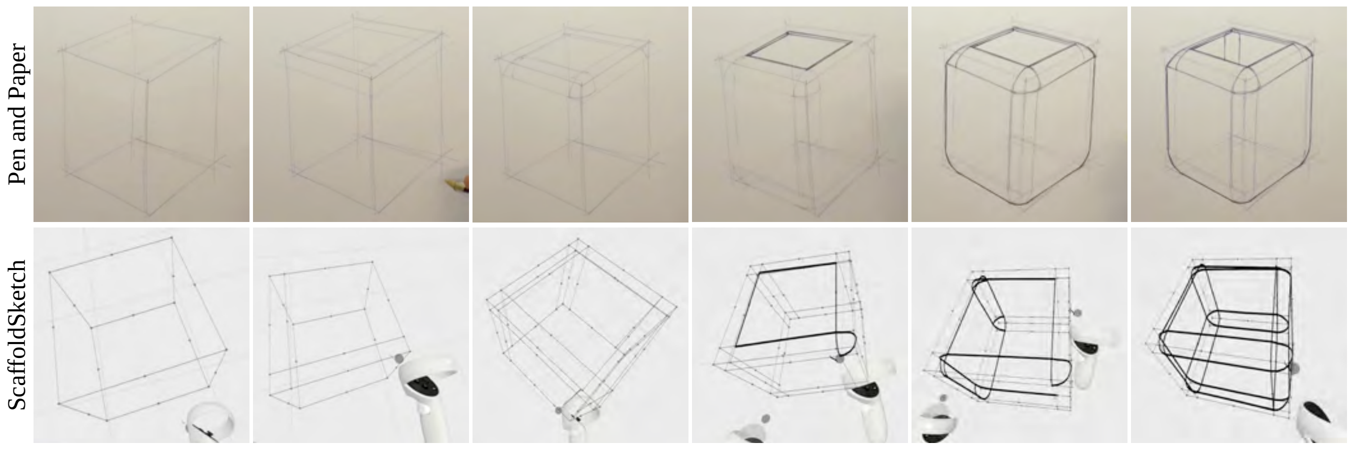

Figure 2: Top row: Frames from a time lapse pen-and-paper industrial design drawing ("Product Design Sketching with construction lines" © Chris Wilson). Designers first construct scaffolding and then draw the shape curves on top. Bottom row: The same two-stage design drawing process executed in 3D with ScaffoldSketch. The raw input and auto-corrected output for this example can be see in Figure 3.

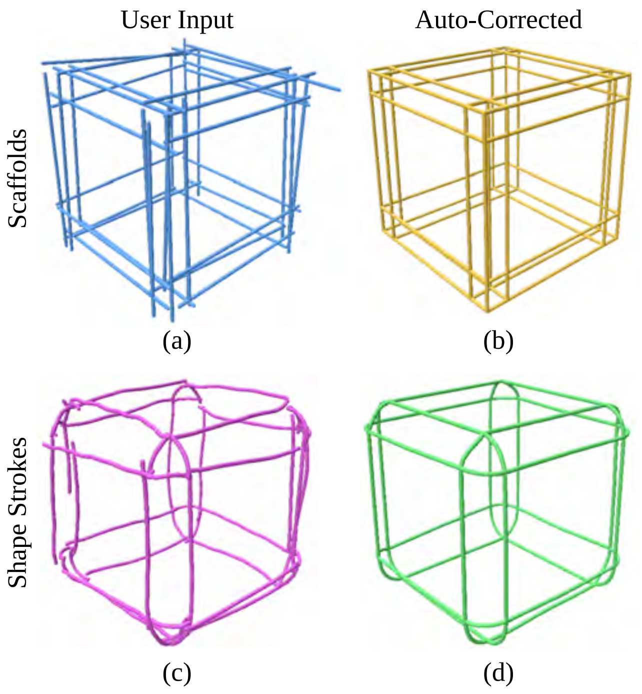

Figure 3: A user's raw input and auto-corrected output in ScaffoldSketch. (a) The raw input scaffold strokes. (b) The auto-corrected scaffolding. (c) The raw input shape strokes. (d) The auto-corrected result.

4 ALGORITHM

As scaffold and shape curves have different affordances, we use different algorithms for auto-correcting them.

4.1 Auto-Correcting Scaffolds

Scaffolds are composed of straight lines with a limited set of interrelationships [20, 21]. The lines may intersect, they may be parallel, perpendicular, or coplanar, and they may have the same length. The challenge is in disentangling which of the many possible relationships are relevant to a newly drawn line (Figure 4). Previous approaches to auto-correcting lines and circles have relied on snapping, heuristic prioritization, or least squares formulations [22, 23, 35, 38, 52]. Snapping and heuristic prioritization are greedy approaches. They will exactly satisfy some constraints in the order considered, but fail in the presence of cycles or compatible constraints affecting the same vertex. This is common in our scenario, since scaffolds support closed shapes. Traditional least squares solutions, on the other hand, can satisfy constraints even in the presence of cycles, but compromise in the presence of conflicting constraints, satisfying none. Since scaffold lines are dense, cycles and conflicting constraints are common. We use iteratively re-weighted least squares (IRLS) to converge on a set of mutually satisfiable constraints, discarding conflicts.

For an input scaffold line l, we formulate each candidate constraint as a function $C(l)$ that measures the squared deviation from constraint satisfaction. A line l is defined by start and end 3D points l_a and l_b which we call "keypoints" for constraint consideration. We also include the line midpoint l_m as a keypoint. User-drawn tick marks (very short strokes crossing a line, indicating intermediate positions) are additional keypoints. The constraints we consider are defined for the line l relative to another scaffold line k, keypoint p, or direction vector d.

Point-point. To construct a closed scaffold polygon, end points of adjacent scaffold strokes must meet. We measure constraint satisfaction with the point-to-point Euclidean distance (Fig. 4), squared. We include this constraint for each endpoint paired with all other existing scaffold keypoints if the constraint function on the input line evaluates to less than $(5\text{cm})^2$ in physical space.

$$C(l_i, p) = \|l_i - p\|^2 \tag{1}$$

Point-line. A line may abut another line but not at one of its ends. For this constraint, we compute the point-to-line distance (Fig. 4). This constraint is added for each endpoint of the input line if the point-line distance with each scaffold line initially evaluates to less than $(5\text{cm})^2$.

$$C(l_i, k) = \frac{\|(k_a - k_b) \times (k_a - l_i)\|^2}{\|k_a - k_b\|^2} \tag{2}$$

Perpendicular lines. Two lines may be perpendicular, particularly if the lines share an endpoint or otherwise abut (Fig. 4), which is measured by the inner product of the line and a scaffold direction d. The constraint energy is zero when it is satisfied. The threshold for inclusion is equivalent to 10°.

$$C(l, d) = ((l_b - l_a) \cdot d)^2 \tag{3}$$

Parallel direction. A line may also be parallel to an existing scaffold line, or to the vertical axis (Fig. 4), which is measured as one minus the inner product of the line and a direction. The constraint energy is also zero when it is satisfied. The threshold for inclusion is equivalent to 10°.

$$C(l, d) = (1 - |(l_b - l_a) \cdot d|)^2 \tag{4}$$

Equal length. Construction of many polyhedra requires edges of equal length (Fig. 4), so we compute the ratio of the new line's length to existing scaffold lines. The threshold for inclusion is 25%.

$$C(l, k) = \left(\frac{\|l_a - l_b\|}{\|k_a - k_b\|} - 1\right)^2 \tag{5}$$

For a new input scaffold line, we compute the value of all possible constraints that may be applied with respect to all existing keypoints, directions, and lines. For each possible constraint, if its value is below a threshold, we include it in our list of constraints. Naively adding all constraint terms together with equal weights $w_i = 1$ leads to a least squares formulation:

$$\min_l \sum w_i C_i(l) \tag{6}$$

Problematic solutions occur with any fixed choice of weights, since optimization will converge to a non-zero balance among conflicting constraints (e.g. two distinct point-point constraints for a single endpoint). This is unsatisfying for the user, as a compromise among constraints is perceived as no constraint being satisfied. In contrast, our IRLS formulation repeatedly re-solves the problem with weights updated after each iteration:

$$w_i = \frac{1}{\epsilon + C_i(l)} \tag{7}$$

This induces a positive feedback loop in which more satisfiable constraints are given larger and larger weights, ultimately assigning binary weights approaching either ∞ (satisfied) or 0 (discarded) for each constraint. We favor point-point constraints by setting their initial weight to $w_i = 100$, and all other initial weights to 1. In our experiments, we have found that this IRLS auto-correct approach converges after 2–6 iterations and is capable of exactly satisfying constraints and intelligently resolving conflicts. In a complex scene (~45 scaffold lines), this takes 0.2 seconds for a scaffold line with few constraints and 1.1 seconds for a scaffold line with many constraints. The end result of this auto-correct is accurate scaffolds with no global distortion, suitable for shape stroke auto-correct (Figures 1, 2, 3, 6, 7, and 10).

4.2 Auto-Correcting Shape Strokes

The shape strokes that define a 3D form appear complex and non-planar yet smooth and fair. In the absence of scaffold lines, they are challenging to draw accurately, even in 2D. In ScaffoldSketch, designers first draw the scaffolds they are accustomed to creating directly in 3D. Then designers draw 3D shape strokes and we auto-correct each curve using the scaffolds as constraints. Our shape strokes have two goals: to respect the scaffold constraints, and to be as beautiful as possible. To satisfy the first goal, we use the user's input 3D curve to select scaffold keypoints and tangents for the curve construction, ensuring that the auto-corrected curve respects the user's intended shape. There are many possible curves that satisfy these constraints. To satisfy our second goal, we perform an optimization on the remaining curve degrees of freedom to achieve a beautiful 3D shape. We use minimum variation of curvature (MVC) curves [34], whose solution space includes French Curves (Euler spirals or clothoids) used in design [9, 33, 40, 47] (but may not be identical depending on boundary conditions [30]). The results are aesthetically pleasing curves that accurately match the user's intended design. Example input and auto-corrected shape strokes can be seen in Figure 5.

4.2.1 *Selecting scaffold constraints.*

From the user input shape stroke, we select a sequence of keypoints and tangent directions to form the basis of our 3D curve. We model our 3D curves as piecewise cubic Bézier splines. We sample the input stroke uniformly in arclength (1 cm) as $P = \{p_1, p_2, \dots p_n\}$. For each p_i we find the nearest scaffold keypoint within a distance threshold (3 cm defined in the user's physical coordinates, so zooming can be used to adjust the results), yielding a sequence $Q = \{q_1, q_2, \dots q_m\}$. It is likely that the same keypoint will be selected multiple times by consecutive shape points that are all within the distance threshold, so we remove repeated elements from Q.

If the input shape curve is nearly tangent to a scaffold at a keypoint, we constrain the curve to have that tangent direction at that keypoint. The input curve's tangent is computed with forward finite differencing, $d = \frac{p_{i+1} - p_i}{\|p_{i+1} - p_i\|}$. We compute the smallest angle between the input curve tangent and any scaffold line passing through the keypoint. If the angle is less than a threshold of 25°, we constrain the curve to have the scaffold line's tangent. If the input curve is not nearly tangent (less than 25°) to a scaffold line, then the local average direction of the input curve is used instead. The result is a sequence $D = \{d_1, d_2, \dots d_m\}$ of tangent directions, one for each keypoint in Q.

The points Q and directions D define our piecewise Bézier spline. Each pair of adjacent q_i, q_{i+1} are the endpoints of a Bézier segment, and $d_i, -d_{i+1}$ are the tangent directions. For each Bézier segment, there are two remaining degrees of freedom: the magnitudes $x_{i,1}$ and $x_{i,2}$ of the tangent control points, defined as $q_i + x_{i,1} d_i$ and $q_{i+1} - x_{i,2} d_{i+1}$. This is the basis of our minimum variation of curvature optimization.

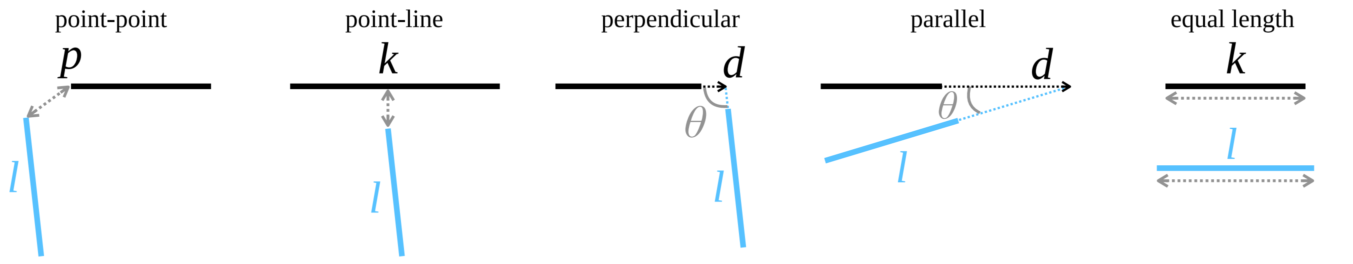

Figure 4: A newly drawn scaffold line l is shown in blue and a pre-existing scaffold line is shown in black. The point-point constraint measures the distance from an endpoint of l to a keypoint p. The point-line constraint measures the distance from an endpoint of l to a previous scaffold line k. The perpendicular constraint measures the angle deviation from 90° between l and a previous scaffold line direction d. The parallel constraint measures the angle deviation between l and a previous scaffold line direction d. The equal length constraint measures the ratio between the length of l and a previous scaffold line k.

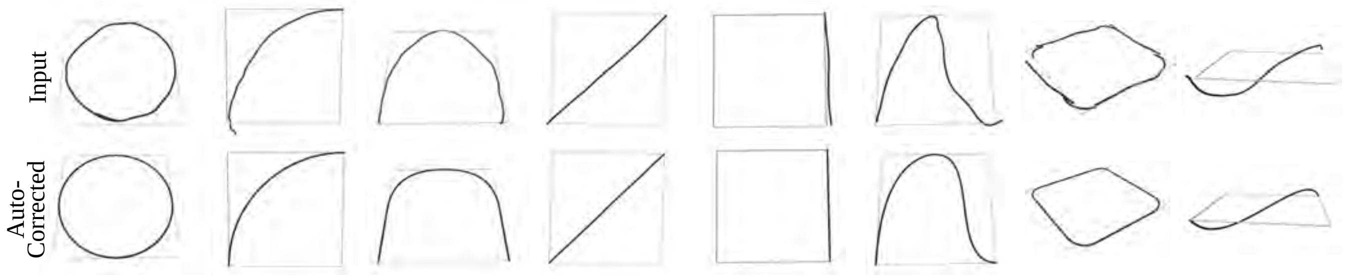

Figure 5: Top row: Input shape strokes. Bottom row: Auto-corrected output. All examples use the same square scaffold. All examples except for the rounded square are a single shape stroke. The auto-corrected results exhibit different position and tangent constraints and the curves achievable by MVC optimization.

4.2.2 Shape curve optimization. Piecewise Bézier curves with minimum variation of curvature (MVC) will form circular arcs when the endpoints and tangent directions allow and approximate French curves (clothoids or Euler spirals) when not [30]. MVC was introduced as a technique for creating aesthetically pleasing curves and surfaces that interpolate user constraints in industrial design applications [34]. The shape curves in industrial design sketches rarely subtend arcs of greater than 90 degrees in between keypoints, so Bézier segments are sufficiently expressive and efficient to optimize due to their small number of degrees-of-freedom.

We compute the variation of curvature as the sum of the Menger curvature [51] at sample points along the curve. From our piecewise Bézier defined by Q and D, we uniformly sample in arclength points along it as $B = \{b_1, b_2, \dots b_k\}$, ensuring each Bézier segment has at least one sample. The curvature is computed as

$$c(b_i) = \frac{1}{R_i} = \frac{4A_i}{\|b_{i-1} - b_i\| \, \|b_i - b_{i+1}\| \, \|b_{i+1} - b_{i-1}\|} \tag{8}$$

where R_i is the radius of the circle subtended by b_{i-1}, b_i, b_{i+1} and A_i is the area of the triangle b_{i-1}, b_i, b_{i+1}. The objective of our optimization is the total curvature variation:

$$\sum (c_i - c_{i+1})^2 \tag{9}$$

The degrees of freedom of our optimization are $X = \{(x_{1,1}, x_{1,2}), (x_{2,1}, x_{2,2}), \dots (x_{k-1,1}, x_{k-1,2})\}$, the magnitudes of the tangents of each Bézier segment. There are $2k - 2$ unknowns. X is initialized with values that ensure no cusps exist: $x_{i,1} = x_{i,2} = \frac{1}{3}\|q_i - q_{i+1}\|$. We minimize Equation 9 using SciPy's BFGS implementation. This takes around 50 milliseconds for simple curves like a circle and approximately 1 second for very complex curves involving 9–10 keypoints.

Users sometimes wish to draw perfectly straight shape strokes between two keypoints not aligned with any scaffold tangents. In these cases, our optimization may produce a curve with some undesirable curvature because the detected start and end tangents may not be co-linear. We detect approximately straight input and directly output a perfectly straight line between the first and last keypoint. We use principal component analysis (PCA) to find the best-fitting straight line to the input point samples. We take the segment of the PCA line that bounds the projection of all input point samples. We use the PCA line if the maximum distance between the input points and the PCA line is less than 8% of the PCA line length and if the PCA line length is within 10% of the arc length of the input curve.

5 EVALUATION

To evaluate our approach, we implemented a usable ScaffoldSketch application for the Oculus Quest. The interface is built on THREE.js and WebXR, while the auto-correct engine is a Python service connected to the interface via a WebSocket. Our system is deployable to users' home VR setups, making it easy for industrial designers to experiment with 3D sketching. Figure 6 shows a professional

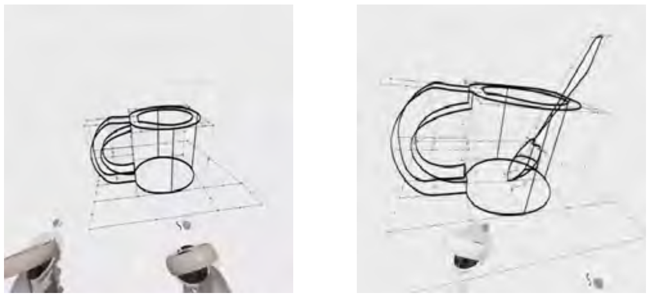

Figure 6: A user in headset using ScaffoldSketch to create a coffee mug with spoon. ScaffoldSketch does not impose global axis alignment. The spoon scaffold and mug scaffolds are aligned to different axes.

industrial designer using ScaffoldSketch to draw a coffee mug and spoon.

We tested ScaffoldSketch to produce a variety of different types of abstract and product designs with different shapes including straight lines and right angles, circular arcs, and planar and non-planar clothoid curves. Some exemplary results can be seen in Figure 7. Compared to recent previous works, ScaffoldSketch is able to demonstrate both accurate and well-aligned 3D shapes including parallel and perpendicular lines of equal length, while also supporting more organic and curved shapes characteristic of freeform 3D painting.

We also conducted a user study to evaluate the usability of ScaffoldSketch.

5.1 Methodology

Due to COVID-19, we performed a remote user study and relied on finding industrial designers who had their own VR headsets (or access to one). We recruited six participants, P1–P6, (ages 20–22, three women, three men) all students of university industrial design programs. Three participants rated themselves as using VR less than one hour per week, two said 1-3 hours per week, and one more than 3 hours per week. Two participants had less than one year experience with product design sketching, while four had 1–5 years experience. All participants had normal or corrected-to-normal stereo vision. Remuneration was a $50 gift card. The study took approximately 1.5 hours to complete.

Because we conducted our study remotely, we relied on the hardware and environment of participants' homes. Four participants used an Oculus Quest, P2 used an Oculus Quest 2, and P6 used an Oculus Rift. All participants used Windows computers. To conduct the study, participants joined a video call with the practitioner and cast their VR headset to the Google Chrome browser on the computer, which was shared via the video call. Because of technical difficulties with the video call, P6 was unable to complete the study. Their results are omitted from the remainder of this discussion.

Each participant used ScaffoldSketch in two conditions, with and without auto-correct. Previous work has established that free-hand 3D drawing in VR is low accuracy [3, 53]. Yet industrial designers are able to create highly accurate free-hand 2D drawings using scaffold lines without auto-correct. Our *without auto-correct* condition tests if scaffold lines themselves are enough to improve VR drawing. Our *with auto-correct* condition allows us to see how much of an improvement auto-correct can provide. Conditions were balanced: half of the participants started with auto-correct, and half started without auto-correct.

Our protocol was approved by an institutional review board. Participants all provided verbal informed consent at the start of the study. In advance of the study, users were given instructions describing ScaffoldSketch and how to prepare their computers. We started the study by reviewing a two minute video tutorial showing the creation of a cylinder with rounded rectangular faces, and then asked the user to reproduce the tutorial object (Figure 8) to familiarize themselves with the system (max ten minutes). The user then performed a directed task given a prompt. We showed the user an industrial design sketch of a trash can outside the headset (Figure 9). The user was asked to draw a trash can in 3D inspired by the prompt using ScaffoldSketch for ten minutes. (The user was not asked to reproduce the exact prompt, and the prompt was not visible during the task.) The user was then asked to create a drawing of their own for another ten minutes. After the open-ended task, users filled out a questionnaire. Then we repeated these tasks (tutorial, directed, open-ended, questionnaire) for the second condition (with or without auto-correct). Finally, participants filled out a post-study questionnaire. Participants were given breaks between tasks. We did not strictly enforce the ten minute time limit if the user requested to spend more time on the task.

5.2 Results

The 3D drawings of participants P1–P5 for the directed and open tasks, with and without auto-correct, can be seen in Figure 10. Users chose a range of objects for the open-ended task, including a shoe, a chair, and a microwave. They also made some creative interpretations of the trash can for the directed task. (The prompt was not visible during the task.)

The data from our questionnaires are presented in Figures 11 and 12. Eight of the questions, Q1–Q8, were repeated for each condition. They are presented together for comparison in Figure 11. We measured statistical significance with a two-sample non-parametric permutation test [39]. Uncorrected p-values are reported in the figure. Four questions, Q9–Q12, were specifically about the auto-correct condition and so do not have paired comparisons. They are presented in Figure 12. Their p-values were computed with a one-sample non-parametric permutation test against the theoretical median response of 'neutral'. A post-hoc Holm-Sidak analysis [45] across all questions found none of the individual comparisons present sufficient evidence to reject the null hypothesis at the $p < .05$ threshold. Aggregating responses across all questions reveals a strong effect in favor of auto-correct using the same tests ($p = 0.0121$ for Q1–Q8; $p = 0.0001$ for Q9–Q12).

6 DISCUSSION

Despite some individually promising results, our study is not powerful enough to find evidence of a significant effect in our individual questions. We believe this is due to the small number of participants, owing to the difficulty of recruiting users with the necessary industrial design skill and access to VR hardware during a pandemic.

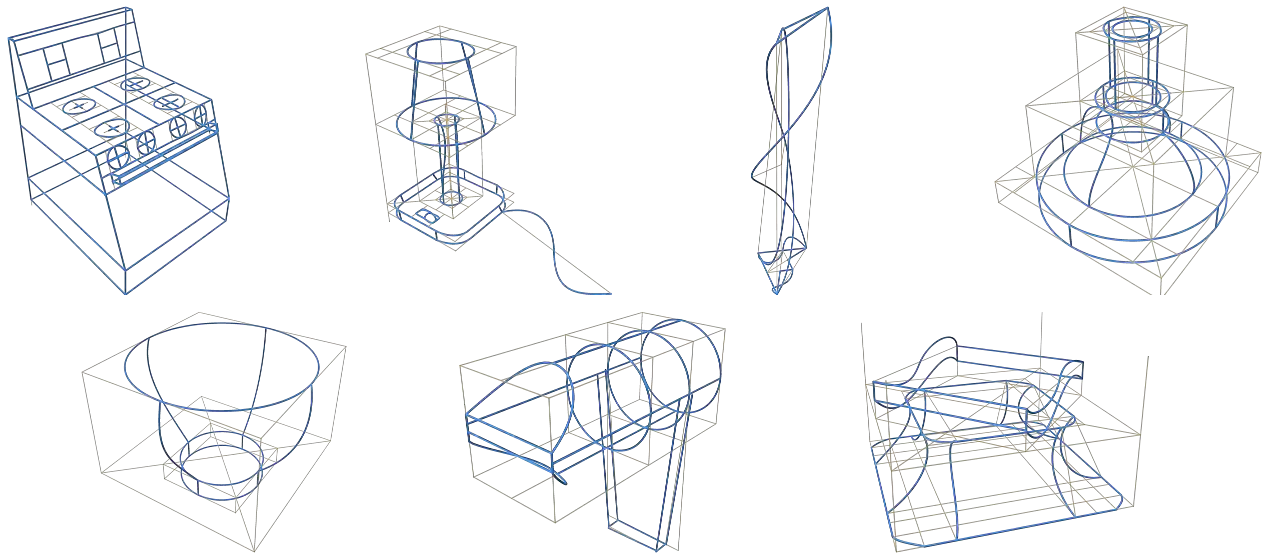
Figure 7: Results created by expert users in our system, rendered with Polyscope [46]. Drawn objects include a kitchen range, a lamp, an abstract sculpture, a flange, a bowl, a hair dryer, and a complex structure. Scaffold lines are thin pale yellow and shape strokes are thick blue.

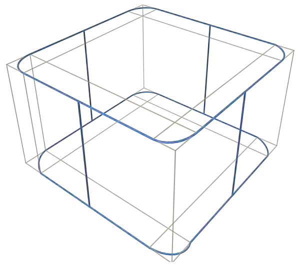
Figure 8: A study participant's tutorial drawing.

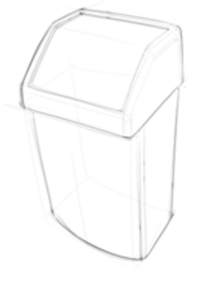
Figure 9: The prompt for the directed task (from Gryaditskaya et al. [19]).

Nevertheless, we find the questionnaire results encouraging and the output drawings display consistent visible differences. Figures 7 and 10 clearly show that auto-correct has a dramatic effect on the achieved quality of the 3D designs. Participants answered unanimously that they strongly agree that "the scaffolds helped [them] draw more accurately" with auto-correct (Q5). Sentiment was unanimously positive-to-neutral for all questions in the auto-correct condition. Further evidence appears in participants' comments.

Participants unanimously agreed that ScaffoldSketch is easy to learn to use (Q1). P1 commented, "The software was very intuitive. The UI was very easy to understand, and it wasn't confusing to understand how to use the software." P5, who did not have previous VR experience, remarked, "The 3D drawing tool was easy to use." This seems consistent with the consensus that VR drawing tools such as Quill and TiltBrush are easy to learn in a similar way to traditional media drawing. The two stage approach of drawing scaffolds and shape curves separately is an additional complexity of ScaffoldSketch, but the participants' industrial design experience likely made it easier for them to learn.

Participants drew comparisons to a prominent commercial competitor, Gravity Sketch, in their comments. While P6 did not complete the study, they did provide comments, including, "The program is great, far more accurate than Gravity Sketch, feels similar to using a pencil with a ruler and a sharpie, because of that there it feels a lot more accurate than other sketch programs in VR," and P2 said, "Scaffolding would help a ton in programs such as Gravity Sketch."

The most closely related recent work to ours is the recent CASSIE system [53]. Their results embody a free flowing and organic expressive style, but have few straight lines or planar curves or other geometric constraints. In contrast, our results include both sweeping non-planar curves as well as straight and rounded lines that satisfy precise relationships.

While participants unanimously agreed that "auto-correct helped [them] draw more accurately" (Q9), there was less general agreement that auto-correct *correctly* interpreted participants' intended scaffold and shape strokes (Q10 and Q11). We measured the number

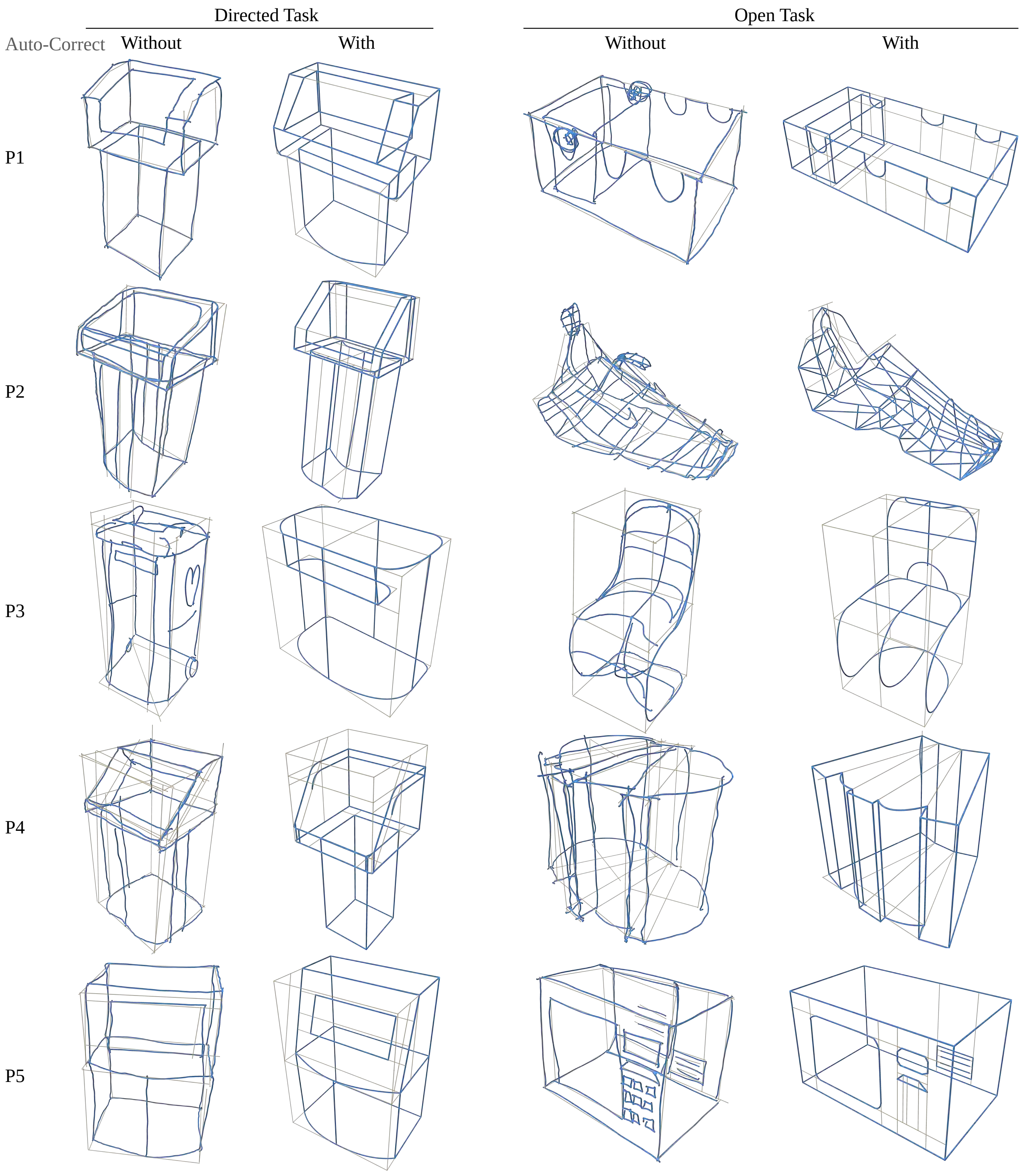

Figure 10: The results of our user study, from all participants (P1–P5), both tasks (directed and open-ended), and both conditions (without and with auto-correct). Scaffold lines are thin pale yellow and shape strokes are thick blue. The prompt for the directed task can be seen in Figure 9.

of times participants used the undo feature during a given session and found that the median number of undos per session in the without auto-correct condition was 16 versus 29 in the with auto-correct condition. While we tuned the auto-correct thresholds to our own preferences, we observed participants sometimes struggling with them. This suggests that calibration or personalization of thresholds per-user may improve auto-correct performance, and that more sustained use may help users become accustomed to the thresholds. Participants sometimes struggled with the zoom functionality, which could have provided a solution to some of these problems.

Two users were neutral regarding whether their auto-correct drawings more accurately reflected their intent than without auto-correct (Q12). P3 elaborated that, "when just used for sketching and ideating, without auto correct does prove to be a valuable ideation tool with the ability to visualize your idea in space quickly," and P5 said "I didn't like it that the drawings would be limiting to make when I wanted to create details, so in that aspect I really liked using the drawing tools because there was no limit." It is a limitation of ScaffoldSketch that we do not currently support turning auto-correct on and off during a drawing session. In 2D design drawing, once the overall form of a shape is defined, designers often add finishing touches such as shading or coloring regions or drawing fine details on or near the implied surface. The user requests support the findings of CASSIE [53] that even when using armatures, users still chose to add some freehand strokes for details or looser, more organic objects.

Overall, we are heartened to find that all users agreed they enjoyed using ScaffoldSketch (Q6) and want to use it in the future for their industrial design needs (Q7). Comments reflect this sentiment: P1 said, "I definitely plan on using this program once it becomes available," and P2, "Great stuff, would love to see this come to market in the future for me to utilize!"

6.1 Reflections

Our primary goal in designing ScaffoldSketch was to create accurate 3D strokes, a well-known challenge when drawing in 3D, by borrowing from conventional industrial design practices. Our user study baseline was not a comparison to raw input. Rather, it was a comparison to scaffolds and shape strokes without auto-correct. 2D designers do not have auto-correct but are able to make high-quality drawings with scaffolds. Our hypothesis was that scaffolds alone would be sufficient to improve VR drawing over the raw input "true" baseline. We found that wasn't the case. We believe this is because the visual reference provided by scaffold lines is insufficient to overcome the depth ambiguity and more complex physical motor program. We didn't include a comparison to strictly raw input, since previous work has established that raw input is problematically imprecise [3, 7, 48, 50]. We determined that a comparison to Gravity Sketch or CASSIE [53], for example, would distract from the central question we wanted to answer, because those tools have different affordances and produce output that differs from ours in important ways. Users may choose to use all three tools for complementary reasons. Our smoothing algorithm was designed around our specific set of constraints. Any curve smoothing algorithm that respects our constraints would probably also generate aesthetically pleasing curves and could be used instead.

When and how much algorithmic auto-correct or beautification to apply is a classic problem. This often boils down to a question of thresholds and user-facing controls to enable/disable or select the degree of desired auto-correction. Overly aggressive auto-correct undesirably modifies user input. Overly conservative auto-correct disappoints users by failing to improve their input.

Our thresholds were determined empirically. They often work well, but we believe threshold personalization is an interesting avenue for future work. We implemented zoom functionality to allow for more precise drawing than our thresholds allow. We believe our users generally overlooked this functionality.

Auto-correct can impede creativity. We believe this to be true for all systems in which errors of interpretation [4] may occur. In our "without auto-correct" study scenario, users sometimes drew details that could not have been drawn easily in the "with auto-correct" scenario. They would have required tedious scaffold creation, since our shape curves must be attached to scaffolds. They cannot have details far from scaffold key points. We could address such curves in the future with new relationships, freeform or detail curve tools, or simply allowing users to toggle auto-correct.

In 2D, high quality design drawings can be created without auto-correct by drawing scaffolds. Very quick sketches or very experienced sketchers may use few scaffolds. In 3D, for the clean line drawing style of our results, we think auto-correct is necessary since it is so much harder to draw accurate smooth curves. Scaffolds allow us to divide-and-conquer the auto-correct problem.

ScaffoldSketch is designed to enable drawing accurate 3D curves in VR. We use design drawing to achieve that goal. The 3D drawings that can be created are more informative than their 2D analog, since they can be rotated. ScaffoldSketch can be used in industrial design processes. However, ScaffoldSketch does not support tools that would be needed for more sophisticated 2D drawings or renderings often created in industrial design pipelines, such as shading or details. ScaffoldSketch also does not visualize inferred constraints or allow users to directly specify properties such as lengths, angles, or congruence.

6.2 Future Work

There are exciting opportunities for continuing to improve VR drawing tools.

Depth Accuracy. Depth perception may be a source of inaccuracy in 3D sketching [3, 7, 48, 50]. Depth *perception* biases are known to be individual [29] and related to the bas-relief ambiguity [8]. We conducted a preliminary experiment ($N = 3$) to assess the importance of depth inaccuracy in our design drawing scenario. In perspective, depth estimation errors would occur along a ray from the eye to a point, not in the "look" direction perpendicular to the image plane. For this reason, we did not detect systematic bias along the look direction for an axis-aligned box-drawing task. In a second experiment, participants drew lines between random points on the surface of a sphere and its center. We analyzed the center endpoints with principal component analysis. Two participants did not exhibit a depth bias while a third did. This third participant has anomalous stereo vision. We plan to explore an optional error

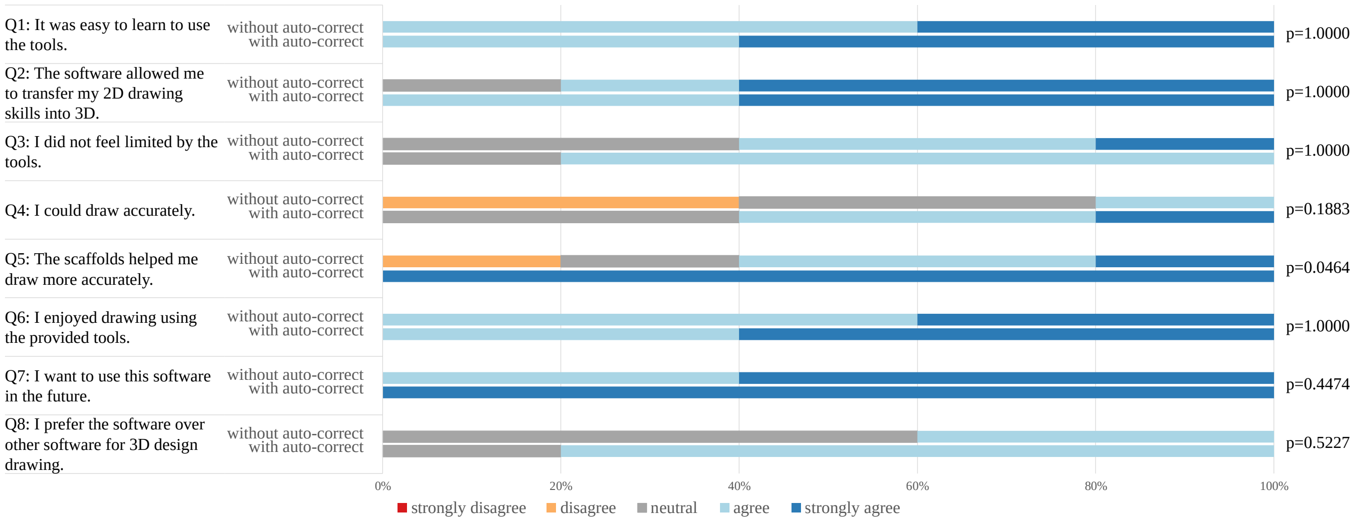

Figure 11: Visualization of the paired question responses. The questions are provided on the left. For each question there are two stacked bars, for the without and with auto-correct conditions. Each bar has five responses from a five point Likert scale. The uncorrected *p*-value comparing each question pair is on the right.

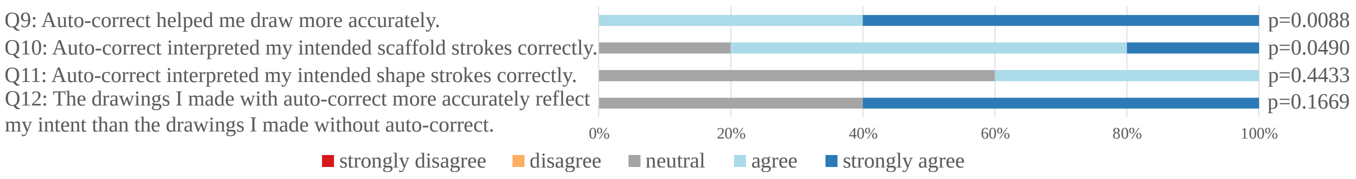

Figure 12: Visualization of the unpaired question responses. The questions are provided on the left. Each question was only asked of the with auto-correct condition. Each bar has five responses from a five-point Likert scale. The uncorrected *p*-value comparing each question to a uniform neutral distribution is on the right.

estimation task for users to personalize the parameters of our auto-correct algorithm based on their psychophysical characteristics, and to incorporate view-dependence into our constraint selection and optimization steps.

Holistic Sketching. Our participants commented that they wished to include both auto-corrected strokes and free-form strokes in their compositions, to support more drawing styles. When showing ScaffoldSketch to potential users, one of the most common requests is that it also includes some form of surfacing, which has been explored in previous work [36, 42, 53]. Surfaces can also support finishing details such as shading or decals, which may be supported by free-form strokes, surface-based edits, and repeated over-sketching [5, 10, 28]. In particular, supporting shading and detail strokes from industrial design practice would allow the creation of product design "renders" in 3D. The combination of all these techniques into a single VR drawing system could be greater than the sum of its parts and move VR drawing beyond single-use tools and into the broader arena of general purpose 3D ideation and creation experiences with applications across computer graphics including fine art, concept art and storyboarding, 3D modeling, CAD, architecture, fabrication, and more.

7 CONCLUSION

ScaffoldSketch is a natural approach for industrial designers to transfer their 2D design drawing skills into 3D. The nature of design drawing allows ScaffoldSketch to auto-correct scaffold and shape strokes separately with algorithms targeting the unique properties of each, resulting in aesthetic and accurate 3D drawings. Industrial designers easily learned to use ScaffoldSketch. There is a tradeoff between freedom and power for any interface which interprets and improves user input. It is inherently inhibiting for users to worry about a computer mis-interpreting their input. Our user study participants find that ScaffoldSketch makes a worthwhile tradeoff.

ACKNOWLEDGMENTS

We are grateful to our reviewers and user testers who invested substantial time pondering and using ScaffoldSketch. We are particularly grateful to Don Park for remote technical support during our user study. Authors Yu and Gingold were supported in part by the United States National Science Foundation (IIS-1453018) and a gift from Adobe Systems Inc.

REFERENCES

[1] Alexis Andre and Suguru Saito. 2011. Single-view sketch based modeling. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling (SBIM '11)*. Association for Computing Machinery, Vancouver, British Columbia, Canada, 133–140. https://doi.org/10.1145/2021164.2021189

[2] 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 (CHI '18)*. ACM, New York, NY, USA, 185:1–185:15. https://doi.org/10.1145/3173574.3173759 event-place: Montreal QC, Canada.

[3] 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. 5643–5654.

[4] Rahul Arora and Karan Singh. 2019. Private Communication.

[5] 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 - UIST '08*. ACM Press, Monterey, CA, USA, 151. https://doi.org/10.1145/1449715.1449740

[6] 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 (UIST '99)*. ACM, New York, NY, USA, 161–169. https://doi.org/10.1145/320719.322598 event-place: Asheville, North Carolina, USA.

[7] Mayra Donaji Barrera Machuca, Wolfgang Stuerzlinger, and Paul Asente. 2019. The Effect of Spatial Ability on Immersive 3D Drawing. In *Proceedings of the 2019 on Creativity and Cognition*. ACM, San Diego CA USA, 173–186. https://doi.org/10.1145/3325480.3325489

[8] Peter N. Belhumeur, David J. Kriegman, and Alan L. Yuille. 1997. The Bas-Relief Ambiguity. In *Proceedings of IEEE CVPR*. 1060–1066.

[9] David Ben-Haim, Gur Harary, and Ayellet Tal. 2010. Piecewise 3D Euler spirals. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling (SPM '10)*. Association for Computing Machinery, New York, NY, USA, 201–206. https://doi.org/10.1145/1839778.1839810

[10] Chris De Paoli and Karan Singh. 2015. SecondSkin: sketch-based construction of layered 3D models. *ACM Transactions on Graphics* 34, 4 (July 2015), 1–10. https://doi.org/10.1145/2766948

[11] Joachim Deisinger, Roland Blach, Gerold Wesche, Ralf Breining, and Andreas Simon. 2000. Towards Immersive Modeling — Challenges and Recommendations: A Workshop Analyzing the Needs of Designers. In *Virtual Environments 2000*, W. Hansmann, W. Purgathofer, F. Sillion, Jurriaan Mulder, and Robert van Liere (Eds.). Springer Vienna, Vienna, 145–156. https://doi.org/10.1007/978-3-7091-6785-4_16

[12] 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.* Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376628

[13] Lynn Eggli, Ching-Yao Hsu, Beat D. Bruderlin, and Gershon Elber. 1997. Inferring 3D models from freehand sketches and constraints. *Computer-Aided Design* 29, 2 (February 1997), 101–112. https://doi.org/10.1016/S0010-4485(96)00039-5

[14] Koos Eissen and Roselien Steur. 2012. *Sketching: basics.* Stiebner Verlag GmbH.

[15] J. Fišer, P. Asente, and D. Sýkora. 2015. ShipShape: A Drawing Beautification Assistant. In *Proceedings of the Workshop on Sketch-Based Interfaces and Modeling* (Istanbul, Turkey) *(SBIM '15)*. Eurographics Association, Goslar, DEU, 49–57.

[16] Yotam Gingold, Takeo Igarashi, and Denis Zorin. 2009. Structured annotations for 2D-to-3D modeling. In *ACM SIGGRAPH Asia 2009 papers*. ACM Press, Yokohama, Japan, 1. https://doi.org/10.1145/1661412.1618494

[17] Tovi Grossman, Ravin Balakrishnan, Gordon Kurtenbach, George Fitzmaurice, Azam Khan, and Bill Buxton. 2001. Interaction Techniques for 3D Modeling on Large Displays. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics (I3D '01)*. ACM, New York, NY, USA, 17–23. https://doi.org/10.1145/364338.364341

[18] Tovi Grossman, Ravin Balakrishnan, Gordon Kurtenbach, George Fitzmaurice, Azam Khan, and Bill Buxton. 2002. Creating Principal 3D Curves with Digital Tape Drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '02)*. ACM, New York, NY, USA, 121–128. https://doi.org/10.1145/503376.503398 event-place: Minneapolis, Minnesota, USA.

[19] Yulia Gryaditskaya, Felix Hähnlein, Chenxi Liu, Alla Sheffer, and Adrien Bousseau. 2020. Lifting freehand concept sketches into 3D. *ACM Transactions on Graphics* 39, 6 (Nov. 2020), 1–16. https://doi.org/10.1145/3414685.3417851

[20] Yulia Gryaditskaya, Mark Sypesteyn, Jan Willem Hoftijzer, Sylvia Pont, Fredo Durand, and Adrien Bousseau. 2019. OpenSketch: A richly-annotated dataset of product design sketches. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 232.

[21] James W. Hennessey, Han Liu, Holger Winnemöller, Mira Dontcheva, and Niloy J. Mitra. 2017. How2Sketch: generating easy-to-follow tutorials for sketching 3D objects. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '17)*. ACM Press, San Francisco, California, 1–11. https://doi.org/10.1145/3023368.3023371

[22] Takeo Igarashi, Sachiko Kawachiya, Hidehiko Tanaka, and Satoshi Matsuoka. 1998. Pegasus: a drawing system for rapid geometric design. In *CHI 98 Conference Summary on Human Factors in Computing Systems*. ACM, Los Angeles California USA, 24–25. https://doi.org/10.1145/286498.286511

[23] Takeo Igarashi, Satoshi Matsuoka, Sachiko Kawachiya, and Hidehiko Tanaka. 2006. Interactive beautification: a technique for rapid geometric design. In *ACM SIGGRAPH 2006 Courses (SIGGRAPH '06)*. Association for Computing Machinery, Boston, Massachusetts, 8–es. https://doi.org/10.1145/1185657.1185769

[24] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. 1999. Teddy: a sketching interface for 3D freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99)*. ACM Press, 409–416. https://doi.org/10.1145/311535.311602

[25] 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 (Sept. 2007), 1067–1081. https://doi.org/10.1109/TVCG.2007.1060

[26] Swarna Keshavabhotla, Blake Williford, Shalini Kumar, Ethan Hilton, Paul Taele, Wayne Li, Julie Linsey, and Tracy Hammond. 2017. Conquering the cube: learning to sketch primitives in perspective with an intelligent tutoring system. In *Proceedings of the Symposium on Sketch-Based Interfaces and Modeling (SBIM '17)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3092907.3092911

[27] 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* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173812

[28] Yongkwan Kim and Seok-Hyung Bae. 2016. SketchingWithHands: 3D Sketching Handheld Products with First-Person Hand Posture. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. Association for Computing Machinery, New York, NY, USA, 797–808. https://doi.org/10.1145/2984511.2984567

[29] J J Koenderink, A J van Doorn, A M L Kappers, and J T Todd. 2001. Ambiguity and the 'mental eye' in pictorial relief. *Perception* 30 (2001), 431–448. Issue 4. http://www.perceptionweb.com/abstract.cgi?id=p3030

[30] Raph Levien. 2009. *From Spiral to Spline: Optimal Techniques in Interactive Curve Design.* Ph.D. Dissertation. University of California, Berkeley.

[31] 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*. 36–47.

[32] Mayra D Barrera Machuca, Wolfgang Stuerzlinger, and Paul Asente. 2019. Smart3DGuides: Making Unconstrained Immersive 3D Drawing More Accurate. 12.

[33] James McCrae and Karan Singh. 2011. Neatening sketched strokes using piecewise French curves. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling - SBIM '11*. ACM Press, Vancouver, British Columbia, Canada, 141. https://doi.org/10.1145/2021164.2021190

[34] Henry P. Moreton and Carlo H. Séquin. 1992. Functional Optimization for Fair Surface Design. *SIGGRAPH Comput. Graph.* 26, 2 (July 1992), 167–176. https://doi.org/10.1145/142920.134035 Place: New York, NY, USA Publisher: Association for Computing Machinery.

[35] S. Murugappan, S. Sellamani, and Karthik Ramani. 2009. Towards beautification of freehand sketches using suggestions. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling - SBIM '09*. ACM Press, New Orleans, Louisiana, 69. https://doi.org/10.1145/1572741.1572754

[36] 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* 34, 4 (July 2015), 1–10. https://doi.org/10.1145/2766990

[37] Theo Pavlidis and Christopher J. Van Wyk. 1985. An Automatic Beautifier for Drawings and Illustrations. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '85)*. ACM, New York, NY, USA, 225–234. https://doi.org/10.1145/325334.325240

[38] Jiantao Pu and Karthik Ramani. 2007. Priority-Based Geometric Constraint Satisfaction. *Journal of Computing and Information Science in Engineering* 7 (Dec. 2007). https://doi.org/10.1115/1.2795301

[39] Sebastian Raschka. 2018. MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *The Journal of Open Source Software* 3, 24 (April 2018). https://doi.org/10.21105/joss.00638

[40] Robert J. Renka. 2005. Shape-preserving interpolation by fair discrete G3 space curves. *Computer Aided Geometric Design* 22, 8 (Nov. 2005), 793–809. https://doi.org/10.1016/j.cagd.2005.03.003

[41] Scott Robertson and Thomas Bertling. 2013. *How to draw: drawing and sketching objects and environments from your imagination* (first edition ed.). Design Studio Press, Los Angeles, CA.

[42] S. Schaefer, J. Warren, and D. Zorin. 2004. Lofting curve networks using subdivision surfaces. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing - SGP '04*. ACM Press, Nice, France, 103. https://doi.org/10.1145/1057432.1057447 ISSN: 17278384.

[43] 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 (CHI '01)*. ACM Press, Seattle, Washington, United States, 261–268. https://doi.org/10.1145/365024.365114

[44] Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. 2009. Analytic drawing of 3D scaffolds. In *ACM SIGGRAPH Asia 2009 papers*. 1–10.

[45] Skipper Seabold and Josef Perktold. 2010. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.

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

[47] Karan Singh. 1999. Interactive curve design using digital French curves. In *Proceedings of the 1999 symposium on Interactive 3D graphics (I3D '99)*. ACM Press, Atlanta, Georgia, United States, 23–30. https://doi.org/10.1145/300523.300525

[48] Julian J. Tramper and C. C. a. M. Gielen. 2011. Visuomotor Coordination Is Different for Different Directions in Three-Dimensional Space. *Journal of Neuroscience* 31, 21 (May 2011), 7857–7866. https://doi.org/10.1523/JNEUROSCI.0486-11.2011 Publisher: Society for Neuroscience Section: Articles.

[49] Steve Tsang, Ravin Balakrishnan, Karan Singh, and Abhishek Ranjan. 2004. A suggestive interface for image guided 3D sketching. In *Proceedings of ACM SIGCHI* (Vienna, Austria). 591–598. https://doi.org/10.1145/985692.985767

[50] E. Wiese, J. H. Israel, A. Meyer, and S. Bongartz. 2010. Investigating the Learnability of Immersive Free-hand Sketching. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium (SBIM '10)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 135–142. http://dl.acm.org/citation.cfm?id=1923363.1923387 event-place: Annecy, France.

[51] Wikipedia contributors. 2020. Menger curvature — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Menger_curvature&oldid=951033306 [Online; accessed 28-January-2021].

[52] 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 (July 2014), 1–13. https://doi.org/10.1145/2601097.2601128

[53] Emilie Yu, Rahul Arora, Tibor Stanko, J. Andreas Bærentzen, Karan Singh, and Adrien Bousseau. 2021. CASSIE: Curve and Surface Sketching in Immersive Environments. In *ACM Conference on Human Factors in Computing Systems (CHI)*. http://www-sop.inria.fr/reves/Basilic/2021/YASBS21