Mid-Air Drawing of Curves on 3D Surfaces in AR/VR

RAHUL ARORA, University of Toronto
KARAN SINGH, University of Toronto

Fig. 1. Drawing curves mid-air that lie precisely on the surface of a virtual 3D object in AR/VR is difficult (a). Projecting mid-air 3D strokes (black) onto 3D objects is an under-constrained problem with many seemingly reasonable solutions (b). We analyze this fundamental AR/VR problem of 3D stroke projection, define and characterize multiple novel projection techniques (c), and test the two most promising approaches—spraycan shown in blue and mimicry shown in red in (b)–(d)—using a quantitative study with 20 users (d). The user-preferred mimicry technique attempts to mimic the 3D mid-air stroke as closely as possible when projecting onto the virtual object. We showcase the importance of drawing curves on 3D surfaces, and the utility of our novel mimicry approach, using multiple artistic and functional applications (e) such as interactive shape segmentation (top) and texture painting (bottom). Horse model courtesy Cyberware, Inc. Spiderman bust base model © David Ruiz Olivares (CC BY 4.0).

1 INTRODUCTION

Drawing is a fundamental tool of human visual expression and communication. Digital sketching with pens, styli, mice, and even fingers in 2D is ubiquitous in visually creative computing applications. Drawing or painting on 3D virtual objects for example, is critical to interactive 3D modeling, animation, and visualization, where its uses include: object selection, annotation, and segmentation [Heckel et al. 2013; Jung et al. 2002; Meng et al. 2011]; 3D curve and surface design [Igarashi et al. 1999; Nealen et al. 2007]; strokes for 3D model texturing or painterly rendering [Kalnins et al. 2002] (Figure 1e). In 2D, digitally drawn on-screen strokes are WYSIWYG mapped onto 3D virtual objects, by projecting 2D stroke points through the given view onto the virtual object(s) (Figure 2a).

Sketching in immersive environments (AR/VR) has the mystical aura of a magical wand, allowing users to draw directly mid-air in 3D. Mid-air drawing thus has the potential to significantly disrupt interactive 3D graphics, as evidenced by the increasing popularity of AR/VR applications such as Tilt Brush [Google 2020] and Quill [Oculus 2020b]. A fundamental requirement for numerous interactive 3D applications in AR/VR however, remains the ability to directly draw, or project drawn 3D strokes, precisely on 3D virtual objects. While directly drawing on a physical 3D object is reasonably easy, it is near impossible without haptic constraints to draw directly on a virtual 3D object (Figure 3). Furthermore, unlike 2D drawing, where the WYSIWYG view-based projection of 2D strokes onto 3D objects is unambiguously clear, the user-intended mapping of a mid-air 3D stroke onto a 3D object is less obvious. We thus present the first detailed investigation into plausible user-intended projections of mid-air strokes onto 3D virtual objects.

Complex 3D curves can be created by directly drawing mid-air in immersive environments (AR/VR). Drawing mid-air strokes precisely on the surface of a 3D virtual object however, is difficult; necessitating a projection of the mid-air stroke onto the user "intended" surface curve. We present the first detailed investigation of the fundamental problem of 3D stroke projection in AR/VR. An assessment of the design requirements of real-time drawing of curves on 3D objects in AR/VR is followed by the definition and classification of multiple techniques for 3D stroke projection. We analyze the advantages and shortcomings of these approaches both theoretically and via practical pilot testing. We then formally evaluate the two most promising techniques spraycan and mimicry with 20 users in VR. The study shows a strong qualitative and quantitative user preference for our novel stroke mimicry projection algorithm. We further illustrate the effectiveness and utility of stroke mimicry, to draw complex 3D curves on surfaces for various artistic and functional design applications.
Interfaces for 2D/3D curve creation in general, use perceptual insights or geometric assumptions like smoothness and planarity, to project, neaten, or otherwise process sketched strokes. Some applications wait for user stroke completion before processing it in entirety, for example when fitting splines [Bae et al. 2008]. Our goal is to establish an application agnostic, base-line projection approach for mid-air 3D strokes. We thus assume a stroke is processed while being drawn and inked in real-time, i.e., the output curve corresponding to a partially drawn stroke is fixed/inked in real-time, based on partial stroke input [Thiel et al. 2011].

One might further conjecture that all "reasonable" and mostly continuous projections would produce similar results, as long as users are given interactive visual feedback of the projection. This is indeed true for tasks requiring discrete point-on-surface selection, where users can freely re-position the drawing tool until its interactively visible projection corresponds to user-intent. Real-time drawing, however, is very sensitive to the projection technique, where any mismatch between user intention and algorithmic projection, is continuously inked into the projected curve (Figure 1d).

2D Strokes Projected onto 3D Objects: The standard user-intended mapping of a 2D on-screen stroke is a raycast projection through the given monocular viewpoint, onto the visible surface of 3D objects. Raycasting is WYSIWYG (What You See Is What You Get) in that the 3D curve visually matches the 2D stroke from said viewpoint (see Figure 2a). Ongoing research on mapping 2D strokes to 3D objects assumes this fundamental view-centric projection, focusing instead on specific problems such as creating spatially coherent curves around ridge/valley features (where small 2D error can cause large 3D depth error upon projection, Figure 2b); or drawing complex curves with large scale variation (where multiple viewpoint changes are needed while drawing, Figure 2c). These problems are mitigated by the direct 3D input and viewing flexibility of AR/VR, assuming the mid-air stroke to 3D object projection matches user intent.

3D Strokes Projected onto 3D Objects: Physical analogies motivate existing approaches to defining a user-intended projection from 3D points in a mid-air stroke to 3D points on a virtual object (Figure 4). Graffiti-style painting with a spraycan, is arguably the current standard, deployed in commercial immersive paint and sculpt software such as Oculus Medium [2020a] and Gravity Sketch [2020]. A closest-point projection approximates drawing with the tool on the 3D object, without actual physical contact (used by the "guides" tool in Tilt Brush [2020]). Like view-centric 2D stroke projection, these approaches are context-free: processing each mid-air point independently. The AR/VR drawing environment comprising six-degree of freedom controller input and unconstrained binocular viewing, is however, significantly richer than 2D sketching. The user-intended projection of a mid-air stroke (§3) as a result is complex, influenced by the ever-changing 3D relationship between the view, drawing controller and virtual object. We therefore argue the need for historical context (i.e., the partially drawn stroke and its projection) in determining the projection of a given stroke point. We balance the use of this historical context, with the overarching goal of a general purpose projection that makes little or no assumption on the nature of the user stroke or its projection.

We thus explore anchored projection techniques, that minimally use the most recently projected stroke point, as context for projecting the current stroke point (§4). We evaluate various anchored projections, both theoretically and practically by pilot testing. Our most promising and novel approach anchored-smooth-closest-point (also called mimicry), captures the natural tendency of a user stroke to mimic the shape of the desired projected curve. A formal user study (§5), shows mimicry to perform significantly better than spraycan (the current baseline) in producing curves that match user intent (§6). This paper thus contributes, to the best of our knowledge, the first principled investigation of real-time inked techniques to project 3D mid-air strokes drawn in AR/VR onto 3D virtual objects, and a novel stroke projection benchmark for AR/VR: mimicry.

Overview. Following a review of related work (§2), we analyze the pros and cons of context-free projection (§3), laying the foundation for our novel anchored projection, mimicry (§4). We formally compare mimicry against the current baseline spraycan (§5). The study results and discussion (§6) are followed by applications showcasing the utility of mimicry (§7). We conclude with limitations and directions for future work (§8).

2 RELATED WORK

Our work is related to research on drawing and sculpting in immersive realities, interfaces for drawing curves on, near, and around surfaces, and sketch-based modelling tools.

2.1 Immersive Sketching and Modeling

Immersive creation has a long history in computer graphics. Immersive 3D sketching was pioneered by the HoloSketch system [Deering
1995], which used a 6-DoF wand as the input device for creating polyline sketches, 3D tubes, and primitives. In a similar vein, various subsequent systems have explored the creation of freeform 3D curves and swept surfaces [Google 2020; Keefe et al. 2001; Schkolne et al. 2001]. While directly turning 3D input to creative output is acceptable for ideation, the inherent imprecision of 3D sketching is quickly apparent when more structured creation is desired.

The perceptual and ergonomic challenges in precise control of 3D input is well-known [Arora et al. 2017; Keefe et al. 2007; Machuca et al. 2018, 2019; Wiese et al. 2010], resulting in various methods for correcting 3D input. Input 3D curves have been algorithmically regularized to snap onto existing geometry, as with the Free-Drawer [2001] system, or constrained physically to 2D input with additional techniques for "lifting" these curves into 3D [Arora et al. 2018; Jackson and Keefe 2016; Kwan and Fu 2019; Paczkowski et al. 2011]. Haptic rendering devices [Kamuro et al. 2011; Keefe et al. 2007] and tools utilizing passive physical feedback [Grossman et al. 2002] are an alternate approach to tackling the imprecision of 3D inputs. We are motivated by similar considerations.

Arora et al. [2017] demonstrated the difficulty of creating curves that lie exactly on virtual surfaces in VR, even when the virtual surface is a plane. This observation directly motivates our exploration of techniques for projecting 3D strokes onto surfaces, instead of coercing users to awkwardly draw exactly on a virtual surface.

2.2 Drawing Curves on, near, and around Surfaces

Curves on, near, and around surfaces are used to model or annotate structural features [Gal et al. 2009; Stanculescu et al. 2013], define trims and holes [Schmidt and Singh 2010], and to provide handles for shape deformation [Kara and Shimada 2007; Nealen et al. 2007; Singh and Fiume 1998], registration [Gehre et al. 2013], de/define trims and holes [Schmidt and Singh 2010], and model surfaces [Google 2006], texture painting [Adobe 2020], and even texture synthesis [Fisher et al. 2007]. Curve on surface creation in this body of research typically uses the established view-centric WYSIWYG projection [Fisher et al. 2007]. Curve on surface creation in this body of research typically uses the established view-centric WYSIWYG projection of on-screen sketched 2D strokes. While the sketch view-point in these interfaces is interactively set by the user, there has been some effort in automatic camera control for drawing [Ortega and Vincent 2014], auto-rotation of the sketching view for 3D planar curves [McCrae et al. 2014], and user assistance in selecting the most sketchable viewpoints [Bae et al. 2008]. Immersive 3D drawing enables direct, view-point independent 3D curve sketching, and is thus an appealing alternative to these 2D interfaces.

Our work is also related to drawing curves around surfaces. Such techniques are important for a variety of applications: modeling string and wire that wrap around objects [Coleman and Singh 2006]; curves that loosely conform to virtual objects or define collision-free paths around objects [Krs et al. 2017]; curve patterns for clothing design on a 3D mannequin model [Turquin et al. 2007]; curves for layered modeling of shells and armour [De Paoli and Singh 2015]; and curves for the design and grooming of hair and fur [Fu et al. 2007; Schmid et al. 2011; Xing et al. 2019]. Some approaches such as SecondSkin [2015] and Skippy [2017] use insights into spatial relationship between a 2D stroke and the 3D object, to infer a 3D curve that lies on and around the surface of the object. Other techniques like Cords [2006] or hair and clothing design [Xing et al. 2019] are closer to our work, in that they drape 3D curve input on and around 3D objects using geometric collisions or physical simulation. In contrast, this paper is application agnostic, and remains focused on the general problem of projecting a drawn 3D stroke to a real-time inked curve on the surface of a virtual 3D object. While we do not address curve creation with specific geometric relationships to the object surface (like distance-offset curve), our techniques can be extended to incorporate geometry-specific terms (§ 8).

2.3 Sketch-based 3D Modeling

Sketch-based 3D modeling is a rich ongoing area of research (see survey by Olsen et al. [2009]). Typically, these systems interpret 2D sketch inputs for various shape modeling tasks. One could categorize these modeling approaches as single-view (akin to traditional pen on paper) [Andre and Saito 2011; Chen et al. 2013; Schmidt et al. 2009; Xu et al. 2014] or multi-view (akin to 3D modeling with frequent view manipulation) [Bae et al. 2008; Fan et al. 2013, 2004; Igarashi et al. 1999; Nealen et al. 2007]. Single-view techniques use perceptual insights and geometric properties of the 2D sketch to infer its depth in 3D, while multi-view techniques explicitly use view manipulation to specify 3D curve attributes from different views. While our work utilizes mid-air 3D stroke input, the ambiguity of projection onto surfaces connects it to the interpretative algorithms designed for sketch-based 3D modeling. We aim to take advantage of the immersive interaction space by allowing view manipulation as and when desired, independent of geometry creation.

3 PROJECTING STROKES ON 3D OBJECTS

We first formally state the problem of projecting a mid-air 3D stroke onto a 3D virtual object. Let $M = (V, E, F)$ be a 3D object, represented as a manifold triangle mesh embedded in $\mathbb{R}^3$. A user draws a piece-wise linear mid-air stroke by moving a 6-DoF controller or drawing tool in AR/VR. The 3D stroke $\mathcal{P} \subset \mathbb{R}^3$ is a sequence of $n$ points $(p_i)_{i=0}^{n-1}$, connected by line segments. Corresponding to each point $p_i \in \mathbb{R}^3$, is a system state $S_i = (h_i, c_i, h_i, c_i)$, where $h_i, c_i \in \mathbb{R}^3$ are the positions of the headset and the controller, respectively, and $h_i, c_i \in \text{Sp}(1)$ are their respective orientations, represented as unit quaternions. Also, without loss of generality, assume $c_i = p_i$, i.e. the controller positions describe the stroke points $p_i$.

We want to define a projection $\pi$, which transforms the sequence of points $(p_i)_{i=0}^{n-1}$ to a corresponding sequence of points $(q_i)_{i=0}^{n-1}$ on the 3D virtual object, i.e. $q_i \in \mathcal{M}$. Consecutive points in this sequence are connected by geodesics on $M$, they describe the projected curve $\mathcal{Q} \subset \mathcal{M}$. The aim of a successful projection method of course, is to match the undisclosed user-intended curve. The projection is also designed for real-time inking of curves: points $p_i$ are processed upon input and projected in real-time (under 100ms) to $q_i$ using the current system state $S_i$, and optionally, prior system states $(S_j)_{j=0}^{i-1}$ stroke points $(p_j)_{j=0}^{i-1}$ and their projections $(q_j)_{j=0}^{i-1}$.

Stroke dynamics, captured from the controller’s inertial sensors, or as finite differences of stroke position, have been effective in...
interactive sketch neatening [Arora et al. 2018; Thiel et al. 2011]. We do not however, explicitly model stroke dynamics in our proposed projections, since early pilot testing did not suggest a relationship between stroke velocity/acceleration and intended stroke projection.

3.1 Context-Free Projection Techniques

Context-free techniques project points independent of each other, simply based on the spatial relationships between the controller, HMD, and 3D object (Figure 4). We can further categorize techniques as raycast or proximity based.

3.1.1 Raycast Projections. View-centric projection in 2D interfaces project points from the screen along a ray from the eye through the screen point, to where the ray first intersects the 3D object. In an immersive setting, raycast approaches similarly use a ray emanating from the 3D stroke point to intersect 3D objects. This ray \((o, d)\) with origin \(o\) and direction \(d\) can be defined in a number of ways. Similar to pointing behavior, Occlude defines this ray from the eye through the controller origin (also the stroke point, Figure 4a) \((e, (e_i - h_i)/\|e_i - h_i\|)\). If the ray intersects \(M\), then the closest intersection to \(p_i\) defines \(q_i\). In case of no intersection, \(p_i\) is ignored in defining the projected curve, i.e., \(q_i\) is marked undefined and the projected curve connects \(q_{i-1}\) to \(q_{i+1}\) (or the proximal index points on either side of \(i\) for which projections are defined). The Spraycan approach treats the controller like a spraycan, defining the ray like a nozzle direction in the local space of the controller (Figure 4b). For example the ray could be defined as \((e_i, f_i)\), where the nozzle \(f_i = c_i \cdot [0, 0, 1]^T\) is the controller’s local z-axis (or forward direction). Alternately, Head-centric projection can define the ray using the HMD’s view direction as \((h_i, h_i \cdot [0, 0, 1]^T)\) (Figure 4c).

Pros and Cons: The strengths of raycasting are: a predictable visual/propiroceptive sense of ray direction; a spatially continuous mapping between user input and projection rays; and AR/VR scenarios where it is difficult or undesirable to reach and draw close to the virtual object. The biggest limitation of raycast projection stems from the controller/HMD-based ray direction being completely agnostic of the shape or location of the 3D object. Projected curves can consequently be very different in shape and size from drawn strokes, and ill-defined for stroke points with no ray/object intersection.

3.1.2 Proximity-Based Projections. In 2D interfaces, the on-screen 2D strokes are typically distant to the viewed 3D scene, necessitating some form of raycast projection onto the visible surface of 3D objects. In AR/VR, however, users are able to reach out in 3D and directly draw the desired curve on the 3D object. While precise mid-air drawing on a virtual surface is very difficult in practice (Figure 3), projection methods based on proximity between the mid-air stroke and the 3D object are certainly worth investigation.

The simplest proximity-based projection technique Snap, projects a stroke point \(p_i\) to its closest-point in \(M\) (Figure 4d).

\[
q_i = \pi_{\text{snap}}(p_i) = \arg \min_{x \in M} d(p_i, x),
\]

where \(d(\cdot, \cdot)\) is the Euclidean distance between two points. Unfortunately, for triangle meshes, closest-point projection tends to snap to the edges of the mesh (blue curve inset), resulting in unexpectedly jaggy projected curves, even for smooth 3D input strokes (black curve inset) [Panozzo et al. 2013]. These discontinuities are due to the discrete nature of the mesh representation, as well spatial singularities in closest point computation even for smooth 3D objects. We mitigate this problem by formulating an extension of Panozzo et al.’s Phong projection [2013] in §3.2, that simulates projection of points onto an imaginary smooth surface approximated by the triangle mesh. We denote this smooth-closest-point projection as \(\pi_{\text{SCP}}\) (red curve inset).

Pros and Cons: The biggest strength of proximity-based projection is it exploits the immersive concept of drawing directly on or near an object, using the spatial relationship between a 3D stroke point and the 3D object to determine projection. The main limitation is that since users rarely draw precisely on the surface, discontinuities and local extrema persist when projecting distantly drawn stroke points, even when using smooth-closest-point. In §4.1, we address this problem using stroke mimicry to anchor distant stroke points close to the object to be finally projected using smooth-closest-point.

3.2 Smooth-Closest-Point Projection

Our goal with smooth-closest-point projection is to define a mapping from a 3D point to a point on \(M\) that approximates the closest point projection but tends to be functionally smooth, at least for points near the 3D object. We note that computing the closest point to a Laplacian-smoothed mesh proxy, for example, will also provide a smoother mapping than \(\pi_{\text{SCP}}\), but a potentially poor closest-point approximation to the original mesh.

Phong projection, introduced by Panozzo et al. [2013], addresses these goals for points expressible as weighted-averages of points.
For the problem of computing weighted averages on surfaces, one only needs to project 3D points of the form \( y^d = \sum w_i x^d_i \), where all \( x^d_i \in M^d \). The point \( y^d \) is lifted into \( \mathbb{R}^d \) by simply defining \( y^d = \sum w_i x^d_i \), where \( x^d_i \) is defined as the point on \( M^d \) with the same implicit coordinates (triangle and barycentric coordinates) as \( x^d_i \) does on \( M^3 \). Therefore, their approach only embeds \( M^3 \) into \( \mathbb{R}^d \) (Figure 5a,c). In contrast, we want to project arbitrary points near \( M^3 \) onto it using the Phong projection. Therefore, we compute the offset surfaces at signed-distance \( \pm \mu \) from \( M \). We then compute a tetrahedral mesh \( T^3_M \) of the space between these two surfaces in \( \mathbb{R}^3 \). In the final step, we embed the vertices of \( T^3_M \) in \( \mathbb{R}^d \) using MDS and LS-Meshes as described above. Note that all of the above steps are realized in a precomputation.

Now, given a 3D point \( y^3 \) within a distance \( \mu \) from \( M^3 \), we situate it within \( T^3_M \), use tetrahedral Barycentric coordinates to infer its location in \( \mathbb{R}^d \), and then compute its Phong projection (Figure 5b,c). We fallback to closest-point projection for points outside \( T^3_M \), since Phong projection converges to closest-point projection when far from \( M \). Furthermore, we set \( \mu \) large enough to easily handle our smooth-closest-point queries in § 4.1.

3.3 Analysis of Context-Free Projection

We implemented the four different context-free projection approaches in Figure 4, and had 4 users informally test each, drawing a variety of curves on the various 3D models seen in this paper. Qualitatively, we made a number of observations:

- **Head-centric** and **Occlude** projections become unpredictable if the user is inadvertently changing their viewpoint while drawing. These projections are also only effective when drawing frontally on an object, like with a 2D interface. Neither as a result exploits the potential gains of mid-air drawing in AR/VR.
- **Spraycan** projection was clearly the most effective context-free technique. Commonly used for graffiti and airbrushing, usually on fairly flat surfaces, we noted however, that consciously reorienting the controller while drawing on or around complex objects was both cognitively and physically tiring.
- **Snap** projection was quite sensitive to changes in the distance of the stroke from the object surface, and in general produced the most undulating projections due to closest-point singularities.
- All projections converge to the mid-air user stroke when it precisely conforms to the surface of the 3D object. But as the distance between the object and points on the mid-air stroke increases, their behavior diverges quickly.
- While users did draw in the vicinity and mostly above the object surface, they rarely drew precisely on the object. The average distance of stroke points from the target object was observed to be 4.8 cm in a subsequent user study (§ 5).
- The most valuable insight however, was that the user stroke in mid-air often tended to mimic the expected projected curve.

Context-free approaches, by design, are unable to capture this mimicry, i.e., the notion that the change between projected point as we draw a stroke is commensurate with the change in the 3D points along the stroke. This inability due to a lack of curve history or context, materializes as problems in different forms.


3.3.1 Projection Discontinuities. Proximal projection (including smooth-closest-point) can be highly discontinuous with increasing distance from the 3D object, particularly in concave regions (Figure 6a). Mid-air drawing along valleys without staying in precise contact with virtual object is thus extremely difficult. Raycast projections can similarly suffer large discontinuous jumps across occluded regions (in the ray direction) of the object (Figure 6d).

While this problem theoretically exists in 2D interfaces as well, it is less observed in practice for two reasons: 2D drawing on a constraining physical surface is significantly more precise than mid-air drawing in AR/VR [Arora et al. 2017]; and artists minimize such discontinuities by carefully choosing appropriate views (raycast directions) before drawing each curve. Automatic direction control of view or controller, while effective in 2D [Ortega and Vincent 2014], is detrimental to a sense of agency and presence in AR/VR.

3.3.2 Undesirable Snapping. Proximity-based methods also tend to get stuck on sharp (or high curvature) convex features of the object (Figure 6b). While this can be useful to trace along a ridge feature, it is particularly problematic for general curve-on-surface drawing.

3.3.3 Projection depth disparity. The relative orientation between the 3D object surface and raycast direction can cause large depth disparities between parts of user strokes and curves projected by raycasting (Figure 6c). Such irregular bunching or spreading of points on the projected curve also goes against our observation of stroke mimicry. Users can arguably reduce this disparity by continually re-orienting the view/controller to keep the projection ray well aligned with object surface normal. Unfortunately, the difference between Occlude and Spraycan ray directions was often large enough to make even smooth ray transitions abrupt and hard to control. All these problems point to the projection function ignoring the shape of the mid-air stroke $P$ and the projected curve $Q$, and can be addressed using projection functions that explicitly incorporate both. We call these functions anchored.

4 Anchored Stroke Projection

The limitations of context-free projection can be addressed by equipping stroke projection with the context/history of recently drawn points and their projections. In this paper we minimally use only the most recent stroke point $p_{i-1}$ and its projection $q_{i-1}$, as context to anchor the current projection.

Any reasonable context-free projection can be used for the first stroke point $p_0$. We use spraycan $\pi_{\text{spray}}$, our preferred context-free technique. For subsequent points ($i > 0$), we compute:

$$r_i = q_{i-1} + \Delta p_i,$$

where $\Delta p_i = (p_i - p_{i-1})$. We then compute $q_i$ as a projection of the anchored stroke point $r_i$ onto $M$, that attempts to capture $\Delta p_i = \Delta q_i$. Anchored projection captures our observation that the mid-air user stroke tends to mimic the shape of their intended curve on surface. While users to do not adhere consciously to any precise geometric formulation of mimicry, we observe that users often draw the intended projected curve as a corresponding stroke on an imagined offset or translated surface (Figure 7). A good general projection for the anchored point $r_i$ to $M$ thus needs to be continuous, predictable, and loosely capture this notion of mimicry.

4.1 Mimicry Projection

Controller sampling rate in current AR/VR systems is 50Hz or more, meaning that even during ballistic movements, the distance $\|\Delta p_i\|$
for any stroke sample $i$ of the order of a few millimetres. Consequently, the anchored stroke point $r_i$ is typically much closer to $M$, than the stroke point $p_i$, making closest-point snap projection a compelling candidate for projecting $r_i$. Such an anchored closest-point projection explicitly minimizes $\|\Delta p_i - \Delta q_i\|$, but precise minimization is less important than avoiding projection discontinuities and undesirably snapping, even for points close to the mesh. Our formulation of a smooth-closest-point projection $\pi_{SCP}$ in § 3.2 addresses these goals precisely. Also note that the maximum observed $\|\Delta p\|$ for the controller readily defines the offset distance $\mu$ for our pre-computed tet mesh $T^3$. We define mimicry projection as

$$\Pi_{mimicry}(p_i) = \begin{cases} \pi_{spray}(p_i) & \text{if } i = 0, \\ \pi_{SCP}(r_i) & \text{otherwise.} \end{cases} \tag{3}$$

4.2 Refinements to Mimicry Projection

We further explore refinements to mimicry projection, that might improve curve projection in certain scenarios.

**Planar curves** are very common in design and visualization [McCrae et al. 2011]. We can locally encourage planarity in mimicry projection by constructing a plane $N_i$ with normal $\Delta p_i \times \Delta p_i - 1$ (i.e. the local plane of the mid-air stroke) and passing through the anchor point $r_i$ (Figure 7b). We then intersect $N_i$ with $M$. $q_i$ is defined as the closest-point to $r_i$ on the intersection curve that contains $q_{i-1}$. Note, we use $\pi_{spray}(p_i)$ for $i < 2$, and we retain the most recently defined normal direction ($N_{i-1}$ or prior) when $N_i = \Delta p_i \times \Delta p_i - 1$ is undefined. We find this method works well for near-planar curves, but the plane is sensitive to noise in the mid-air stroke (Figure 9f), and can feel sticky or less responsive for non-planar curves.

**Offset and Parallel surface drawing** captures the observation that users tend to draw an intended curve as a corresponding mid-air stroke on an imaginary offset or parallel surface of the object $M$. While we do not expect users to draw precisely on such a surface, we note that is unlikely a user would intentionally draw orthogonal to such a surface along the gradient of the 3D object.

In scenarios when a user is sub-consciously drawing on an offset surface of $M$ (an isosurface of its signed-distance function $d_M$), we can remove the component of a user stroke segment that lies along the gradient $\nabla d_M$, when computing the desired anchor point $r_i$ in Equation 4 as (Figure 7c):

$$r_i' = q_{i-1} + \Delta p_i - (\Delta p_i \cdot \nabla d_M(p_i)) \nabla d_M(p_i)$$

$$\tag{4}$$

We can similarly locally constrain user strokes to a parallel surface of $M$ in Equation 5 as:

$$r_i'' = q_{i-1} + \Delta p_i - (\Delta p_i \cdot \nabla d_M(r_i)) \nabla d_M(r_i).$$

$$\tag{5}$$

Note that the difference from Eq. 4 is the position where $\nabla d_M$ is computed, as shown in Figure 7d. A parallel surface better matched user expectation than an offset surface in our pilot testing, but both techniques produced poor results when user drawing deviated from these imaginary surfaces (Figure 9g–i).

4.3 Anchored Raycast Projection

For completeness, we also investigated raycast alternatives to projection of the anchored stroke point $r_i$. We used similar priors of local planarity and offset or parallel surface transport as with mimicry refinement, to define ray directions. Figure 8 shows two such options.

In Figure 8a, we cast a ray in the local plane of motion, orthogonal to the user stroke, given by $\Delta p_i$. We construct the local plane containing $r_i$ spanned by $\Delta p_i$ and $p_{i-1} - q_{i-1}$, and then define the direction orthogonal to $\Delta p_i$ in this plane. Since $r_i$ may be inside $M$, we cast two rays bi-directionally ($r_i \pm \Delta p_i^\pm$), where

$$\Delta p_i^\pm = \Delta p_i \times ((\Delta p_i \times (p_{i-1} - q_{i-1}))$$

If both rays successfully intersect $M$, we choose $q_i$ to be the point closer to $r_i$, a heuristic that works well in practice. As with locally planar mimicry projection, this technique suffered from instability in the local plane.

Motivated by mimicry, in Figure 8b, we also explored parallel transport of the projection ray direction along the user stroke. For $i > 0$, we parallel transport the previous projection direction $q_{i-1} - p_{i-1}$ along the mid-air curve by rotating it with the rotation that aligns $\Delta p_{i-1}$ with $\Delta p_i$. Once again bi-directional rays are cast from $r_i$, and $q_i$ is set to the closer intersection with $M$.

In general, we found that all raycast projections, even when anchored, suffered from unpredictability over long strokes and stroke discontinuities when there are no ray-object intersections (Figure 9n,o).

4.4 Final Analysis and Implementation Details

In summary, extensive pilot testing of the anchored techniques revealed that they seemed generally better than context-free approaches, specially when users drew further away from the 3D object. Among the anchored techniques, stroke mimicry captured as an anchored-smooth-closest-point projection, proved to be theoretically elegant, and practically the most resilient to ambiguities of user intent and differences of drawing style among users. Anchored closest-point can be a reasonable proxy to anchored smooth-closest-point when pre-processing the 3D virtual objects is undesirable.

Our techniques are implemented in C#, with interaction, rendering, and VR support provided by the Unity Engine. For the smooth closest-point operation, we modified Panozzo et al.’s [2013] reference implementation, which includes pre-processing code written in MATLAB and C++, and real-time code in C++. The real-time projection implementation is exposed to our C# application via a compiled dynamic library. In their implementation, as well as ours,
Fig. 9. Mimicry vs. other anchored stroke projections: Mid-air strokes are shown in black and mimicry curves in red. Anchored closest-point (blue), is similar to mimicry on smooth, low-curvature meshes (a,b) but degrades with mesh detail/noise (c,d). Locally planar projection (blue) is susceptible local plane instability (e,f). Parallel (purple h,k) or offset (blue i,l) surface based projection fail in (h,l) when the user stroke deviates from said surface, while mimicry remains reasonable (g, j). Compared to mimicry (m), anchored raycasting based on a local plane (purple n), or ray transport (blue o) can be discontinuous.

$d = 8$; that is, we embed $\mathcal{M}$ in $\mathbb{R}^8$ for computing the Phong projection. We use $\mu = 20$cm, and compute the offset surfaces using libigl [Jacobson et al. 2018]. We then improve the surface quality using TetWild [Hu et al. 2018], before computing the tetrahedral mesh $\mathcal{T}_\mathcal{M}$ between the two surfaces using TetGen [Si 2015].

We support fast closest-point queries, using an AABB tree implemented in geometry3Sharp [Schmidt 2017]. Signed-distance is also computed using the AABB tree and fast winding number [Barill et al. 2018], and gradient $\nabla d \mathcal{M}$ computed using central finite differences.

To ease replication of our various techniques and aid future work, we will open-source our implementation.

We now formally compare our most promising projection mimicry, to the best state-of-the-art context-free projection spraycan.

5 USER STUDY

We designed a user study to compare the performance of the spraycan and mimicry methods for a variety of curve-drawing tasks. We selected six shapes for the experiment (Figure 10), aiming to cover a diverse range of shape characteristics: sharp features (cube), large smooth regions (trebol, bunny), small details with ridges and valleys (bunny), thin features (hand), and topological holes (torus, fertility).

We then sampled ten distinct curves on the surface of each of the six objects. A canonical task in our study involved the participant attempting to re-create a given target curve from this set. We designed two types of drawing tasks shown in Figure 11:

- Tracing curves, where a participant tried to trace over a visible target curve using a single smooth stroke.
- Re-creating curves, where a participant attempted to re-create from memory, a visible target curve that was hidden as soon as the participant started to draw. An enumerated set of keypoints on the curve however, remained as a visual reference, to aid the participant in re-creating the hidden curve with a single smooth stroke.

The rationale behind asking users to draw target curves is both to control the length, complexity, and nature of curves drawn by users, and to have an explicit representation of the user-intended curve. Curve tracing and re-creating are fundamentally different
We wanted to design target curves that could be executed using a single smooth motion. Since users typically draw sharp corners using multiple strokes [Bae et al. 2008], we constrain our target curves to be smooth, created using cardinal cubic B-splines on the meshes, computed using Panozzo et al. [2013]. We also control the length and curvature complexity of the curves, as pilot testing showed that very simple and short curves can be reasonably executed by almost any projection technique. Curve length and complexity is modeled by placing spline control points at mesh vertices, and specifying the desired geodesic distance and Gaussian map distance between consecutive control points on the curve.

We represent a target curve using four parameters \((n, i_0, k_G, k_N)\), where \(n\) is the number of spline control points, \(i_0\) the vertex index of the first control point, and \(k_G, k_N\) constants that control the geodesic and normal map distance between consecutive control points. We define the desired geodesic distance between consecutive control points as, \(D_G = k_G \times \|\text{BBox}(M)\|\), where \(\|\text{BBox}(M)\|\) is the length of the bounding box diagonal of \(M\). The desired Gaussian map distance (angle between the unit vertex normals) between consecutive control points is simply \(k_N\).

A target curve \(C_{i_0}, \ldots, C_{i_{n-1}}\) starting at vertex \(v_{i_0}\) of the mesh is generated incrementally for \(i > 0\) as:

\[
C_i = \arg \min_{v \in V'} \left( d_G(C_{i-1}, v) - D_G \right)^2 + \left( d_N(C_{i-1}, v) - k_N \right)^2, \tag{6}
\]

where \(d_G\) and \(d_N\) compute the geodesic and normal distance between two points on \(M\), and \(V' \subset V\) contains only those vertices of \(M\) whose geodesic distance from \(C_0, \ldots, C_{i-1}\) is at least \(D_G/2\). The restricted subset of vertices conveniently helps prevent (but doesn’t fully avoid) self-intersecting or nearly self-intersecting curves. Curves with complex self-intersections are less important practically, and can be particularly confusing for the curve re-creation task. All our target curve samples were generated using \(k_G \in [0.05, 0.25]\), \(k_N \in [\pi/6, 5\pi/12]\), \(n = 6\), and a randomly chosen \(i_0\). The curves were manually inspected for self-intersections, and enforcing curves rejected.

We then defined keypoints on the target curves as follows: curve endpoints were chosen as keypoints; followed by greedily picking extrema of geodesic curvature, while ensuring that the arclength distance between any two consecutive keypoints was at least 3cm; and concluding the procedure when the maximum arclength distance between any consecutive keypoints was below 15cm. Our target curves had between 4–9 keypoints (including endpoints).

### 5.2 Experiment Design

The main variable studied in the experiment was Projection method—spraycan vs. mimicry—realized as a within-subjects variable. The order of methods was counterbalanced between participants. For each method, participants were exposed to all the six objects. Object order was fixed as torus, cube, trebol, bunny, hand, and fertility, based on our personal judgment of drawing difficulty. The torus was used as a tutorial, where participants had access to additional instructions visible in the scene and their strokes were not utilized for analysis. For each object, the order of the 10 target strokes was randomized. The first five were used for the tracing curves task, while the remaining five were used for re-creating curves.

The target curve for the first tracing task was repeated after the first five unique curves, to gauge user consistency and learning effects. A similar repetition was used for curve re-creation. Participants thus performed 12 curve drawing tasks per object, leading to a total of \(12 \times 5\) (objects) \(\times 2\) (projections) = 120 strokes per participant.

Owing to the COVID-19 physical distancing guidelines, the study was conducted in the wild, on participants’ personal VR equipment at their homes. A 15-minute instruction video introduced the study tasks and the two projection methods. Participants then filled out a consent form and a questionnaire to collect demographic information. This was followed by them testing the first projection method and filling out a questionnaire to express their subjective opinions of the method. They then tested the second method, followed by a similar questionnaire, and questions involving subjective comparisons between the two methods. Participants were required to take a break after testing the first method, and were also encouraged to take breaks after drawing on the first three shapes for each method. The study took approximately an hour, including the questionnaires.

### 5.3 Participants

Twenty participants (5 female) aged 21–47 from five countries participated in the study. All but one were right-handed. Participants self-reported a diverse range of artistic abilities (min. 1, max. 5, median 3 on a 1–5 scale), and had varying degrees of VR experience, ranging from below 1 year to over 5 years. Thirteen participants had a technical computer graphics or HCI background, while ten had experience with creative tools in VR, with one reporting professional usage. Participants were paid \(\approx 22\) USD as a gift card.

### 5.4 Apparatus

As the study was conducted on personal VR setups, a variety of commercial VR devices were utilized—Oculus Rift, Rift S, and Quest using Link cable, HTC Vive and Vive Pro, Valve Index, and Samsung Odyssey using Windows Mixed Reality. All but one participant used a standing setup allowing them to freely move around.
5.5 Procedure
Before each trial, participants could use the “grab” button on their controller (in the dominant hand) to grab the mesh to position and orient it as desired. The trial started as soon as the participant started to draw by pressing the “main trigger” on their dominant hand controller. This action disabled the grabbing interaction—participants could not draw and move the object simultaneously. As noted earlier, for curve re-creation tasks, this had the additional effect of hiding the target curve, but leaving keypoints visible.

6 STUDY RESULTS AND DISCUSSION
We recorded the head position h and orientation h, controller position e and orientation e, projected point q, and timestamp t, for each mid-air stroke point p ∈ e. In general, we will refer to a task target curve by X, P and Pm as the mid-air strokes executed, and Q and Qm, the corresponding curves created using spraycan and mimicry projection, respectively. We drop the superscript when the projection method used is not relevant, referring to a mid-air stroke as P and its projected curve as Q.

6.1 Data Processing and Filtering
We formulated three criteria to filter out meaningless user strokes:

- Short Curves: we ignore projected curves Q that are too short as compared to the length of the target curves X (conservatively curves less than half as long as the target curve). While it is possible that the user stopped drawing mid-way out of frustration, we found it more likely that they prematurely released the controller trigger by accident. Both curve lengths are computed in \( \mathbb{R}^3 \) for efficiency.
- Stroke Noise: we ignore strokes for which the mid-air stroke is too noisy. Specifically, mid-air strokes with distance consecutive points (\( i \) s.t. \( ||p_i - p_{i-1}|| > 5 \text{cm} \)) are rejected.
- Inverted Strokes: while we labelled keypoints with numbers and marked start and end points in green and red (Figure 11), some users occasionally drew the target curve in reverse. The motion inversion method used is not relevant, referring to a mid-air stroke as P and its projected curve as Q.

6.2 Quantitative Analysis
We define 10 different statistical measures (Table 1) to compare \( \pi_{spray} \) and \( \pi_{mimicry} \) curves in terms of their accuracy, aesthetic, and effort in curve creation. We consistently use the non-parametric Wilcoxon signed-rank test for all quantitative measures instead of a parametric test such as the paired t-test, since the recorded data for none of our measures was normally distributed (normality hypothesis rejected via the Kolmogorov-Smirnov test, \( p < .005 \)).

6.2.1 Curve Accuracy. Accuracy is computed using two measures of distance between points on the projected curve Q and target curve X. Both curves are densely re-sampled using \( m = 101 \) sample points equi-spaced by arc-length.

Given \( Q = q_0, \ldots, q_{m-1} \) and \( X = x_0, \ldots, x_{m-1} \), we compute the average equi-parameter distance \( D_{ep} \) as

\[
D_{ep}(Q) = \frac{1}{m} \sum_{i=0}^{m-1} d_{ep}(q_i, x_i), \tag{7}
\]

where \( d_{ep} \) computes the Euclidean distance between two points in \( \mathbb{R}^3 \). We also compute the average symmetric distance \( D_{sym} \) as

\[
D_{sym}(Q) = \frac{1}{2m} \sum_{i=0}^{m-1} \min_{x \in X} d_{ep}(q_i, x) + \frac{1}{m} \sum_{i=0}^{m-1} \min_{q \in Q} d_{ep}(q, x_i).
\]

In other words, \( D_{ep} \) computes the distance between corresponding points on the two curves and \( D_{sym} \) computes the average minimum distance from each point on one curve to the other curve.

Table 1. Quantitative results (mean ± std-dev.) of the comparisons between mimicry and spraycan projection. All measures are analyzed using Wilcoxon signed-rank tests, lower values are better, and significantly better values (\( p < .05 \)) are shown in boldface. Accuracy, aesthetic, and physical effort measures are shown with green, red, and blue backgrounds, respectively.
The smoothness quality of the user stroke $\mathcal{P}$, was similar to $\mathcal{Q}$ and significantly poorer than the target curve $\mathcal{X}$. This is expected since drawing in mid-air smoothly and with precision is difficult, and such strokes are usually neanened post-hoc [Arora et al. 2018]. We therefore avoid comparisons to the target curve and simply report three aesthetic measures for a projected curve $Q = q_0, \ldots, q_{n-1}$. We first refine $Q$ by computing the exact geodesic on $\mathcal{M}$ between consecutive points of $Q$ [Surazhsky et al. 2005], to create $\hat{Q}$ with points $\hat{q}_0, \ldots, \hat{q}_{k-1}$, $k \leq n$. We choose to normalize our curvature measures using $L_X$, the length of the corresponding target stroke $\mathcal{X}$. The normalized Euclidean curvature for $Q$ is defined as

$$K_E(Q) = \frac{1}{L_X} \sum_{i=1}^{k-1} \theta_i,$$

where $\theta_i$ is the angle between the two segments of $\hat{Q}$ incident on $\hat{q}_i$. Thus, $K_E$ is the total discrete curvature of $\hat{Q}$, normalized by the target curve length.

Since $\hat{Q}$ is embedded in $\mathcal{M}$, we can also compute discrete geodesic curvature, computed as the deviation from the straightest geodesic for a curve on surface. Using a signed $\theta_i$ defined at each point $\hat{q}_i$ via Polthier and Schmies’s definition [2006], we compute normalized geodesic curvature as

$$K_G(Q) = \frac{1}{L_X} \sum_{i=1}^{k-1} |\theta_i|.$$

Finally, we define fairness [Arora et al. 2017; McCrae and Singh 2008] as a first-order variation in geodesic curvature, thus defining the normalized fairness deficiency as

$$F_G(Q) = \frac{1}{L_X} \sum_{i=1}^{m-1} |\theta_i - \theta_{i-1}|.$$

For all three measures, a lower value indicates a smoother, pleasing, curve. Wilcoxon signed-rank tests on all three measures indicated that mimicry produced significantly smoother and better curves than spraycan (Table 1).

6.2.3 Physical Effort. Quantitatively, we use the amount of head (HMD) and hand (controller) movement, and stroke execution time $\tau$, as proxies for physical effort.

For head and hand translation, we first filter the position data with a Gaussian-weighted moving average filter with $\sigma = 20ms$. We then define normalized head/controller translation $T_h$ and $T_c$ as the length of the poly-line defined by the filtered head/controller positions normalized by the length of the target curve $L_X$.

An important ergonomic measure is the amount of head/hand rotation required to draw the mid-air stroke. We first de-noise or filter the forward and up vectors of the head/controller frame, using the same filter as for positional data. We then re-orthogonalize the frames and compute the length of the curve defined by the filtered orientations in $SO(3)$, using the angle between consecutive orientation data-points. We define normalized head/controller rotation $R_h$ and $R_c$ as its orientation curve length, normalized by $L_X$.

Table 1 summarizes the physical effort measures. We observe lower controller translation (effect size $\approx 5\%$) and execution time (effect size $\approx 12\%$) in favour of spraycan; lower head translation and
6.2.1, use the correspondence as in Eq. 7 and look at the variation in the distance (distance between the closest pair of corresponding points subtracted from that of the farthest pair) as a percentage of the target length \( L_X \). We call this measure the mimicry violation of a stroke. Intuitively, the lower the mimicry violation, the closer the stroke \( P \) is to being a perfect mimicry of \( X \), going to zero if it is a precise translation of \( X \). Notably, users depicted very similar trends to mimic both for the techniques—with 86% (mimicry), 80% (spraycan) strokes exhibiting mimicry violation below 25% of \( L_X \), and 71%, 66% below 20% of \( L_X \)—suggesting that mimicry is indeed a natural tendency.

### 6.2.5 Consistency across Repeated Strokes
Recall that users repeated 2 of the 10 strokes per shape for both the techniques. To analyze consistency across the repeated strokes, we compared the values of the stroke accuracy measure \( D_{eq} \) and the aesthetic measure \( F_g \) between the original stroke and the corresponding repeated stroke. Specifically, we measured the relative change \( |f(i) − f(i′)|/f(i) \), where \((i, i′)\) is a pair of original and repeated strokes, and \(f(i)\) is either \( D_{eq} \) or \( F_g \). Users were fairly consistent across both the techniques, with the average consistency for \( D_{eq} \) being 35.4% for mimicry and 36.8% for spraycan, while for \( F_g \), it was 36.5% and 34.1%, respectively. Note that the averages were computed after removing extreme outliers outside the 5\( \sigma \) threshold.

### 6.3 Qualitative Analysis
The mid- and post-study questionnaires elicited qualitative responses from participants on their perceived difficulty of drawing, curve accuracy and smoothness, mental and physical effort, understanding of the projection methods, and overall method of preference.

Participants rated their perceived difficulty in drawing on the 6 study objects (Figure 13), validating our ordering of shapes in the experiment based on expected drawing difficulty.

Accuracy, smoothness, physical/mental effort responses were collected via 5-point Likert scales. We consistently order the choices from 1 (worst) to 5 (best) in terms of user experience in Figure 14, and reported median (\( M \)) scores here. Mimicry was perceived to be a more accurate projection method (tracing, re-creating \( M = 3, 3.5 \)) compared to spraycan (\( M = 2, 2 \)), with 9 participants perceiving their traced curves to be either very accurate or somewhat accurate with mimicry (compared to 2 for spraycan) (Figure 14a). User perception of stroke smoothness was also consistent with quantitative results, with mimicry (tracing, re-creating \( M = 4, 4 \)) clearly outperforming spraycan (tracing, re-creating \( M = 1, 2 \)) (Figure 14b). Lastly, with no need for controller rotation, mimicry (\( M = 3 \)) was perceived as less physically demanding than spraycan (\( M = 2 \), as expected (Figure 14c).

The response to understanding and mental effort was more complex. Spraycan, with its physical analogy and mathematically precise definition was clearly understood by all 20 participants (17 very well, 3 somewhat) (Figure 15a). Mimicry, conveyed as “drawing a mid-air stroke on or near the object as similar in shape as possible to the intended projection”, was less clear to users (7 very well, 11 somewhat, 3 not at all). Despite not understanding the method, the 3 participants were able to create curves that were both accurate and smooth. Further, users perceived mimicry (\( M = 2.5 \)) as less

**Figure 13.** Perceived difficulty of drawing for the six 3D shapes in the study.

**Figure 14.** Participants perceived mimicry to be better than spraycan in terms of accuracy (a), curve aesthetic (b) and user effort (c).

**Figure 15.** Participants stated understanding spraycan projection better (left); 17/20 users stated an overall preference for mimicry over spraycan (right).
cognitively taxing than spraycan (M = 2) (Figure 14c). We believe this may be because users were less prone to consciously controlling their stroke direction and rather focused on drawing. The tendency to mimic may have thus manifested sub-consciously, as we had observed in pilot testing.

The most important qualitative question was user preference (Figure 15b). 85% of the 20 participants preferred mimicry (10 highly preferred, 7 somewhat preferred). The remaining users were neutral (1/20) or somewhat preferred spraycan (2/20).

6.4 Participant Feedback

We also asked participants to elaborate on their stated preferences and ratings. Participants (P4,8,16,17) noted discontinuous “jumps” caused by spraycan, and felt the continuity guarantee of mimicry: “seemed to deal with the types of jitter and inaccuracy VR setups are prone to better” (P6); “could stabilize my drawing” (P9). P9,15 felt that mimicry projection was smoothing their strokes (no smoothing was employed): we believe this may be the effect of noise and inadvertent controller rotation, which mimicry ignores, but can cause large variations with spraycan, perceived as curve smoothing.

Some participants (P4,17) felt that rotating the hand smoothly while drawing was difficult, while others missed the spraycan ability to simply use hand rotation to sweep out long projected curves from a distance (P2,7). Participants commented on physical effort: “Mimicry method seemed to required [sic] much less head movement, hand rotation and mental planning” (P4).

Participants appreciated the anchored control of mimicry in high-curvature regions (P1,2,4,8) also noting that with spraycan, “the curvature of the surface could completely mess up my stroke” (P1). Some participants did feel that spraycan could be preferable when drawing on near-flat regions of the mesh (P3,14,19,20).

Finally, participants who preferred spraycan felt that mimicry required more thinking: “with mimicry, there was extra mental effort needed to predict where the line would go on each movement” (P9), or because mimicry felt “unintuitive” (P7) due to their prior experience using a spraycan technique. Some who preferred mimicry found it difficult to use initially, but felt it got easier over the course of the experiment (P4,17).

7 APPLICATIONS

Complex 3D curves on arbitrary surfaces can be drawn in AR/VR with a single stroke, using mimicry (Figure 16). Drawing such curves on 3D virtual objects is fundamental to many applications, including direct painting of textures [Schmidt et al. 2006]; tangent vector field design [Fisher et al. 2007]; texture synthesis [Lefebvre and Hoppe 2006; Turk 2001]; interactive selection, annotation, and object segmentation [Chen et al. 2009]; and seams for shape parametrization [Lévy et al. 2002; Rabinovich et al. 2017; Sawhney and Crane 2017], registration [Gehre et al. 2018], and quad meshing [Tong et al. 2006]. We showcase the utility and quality of mimicry curves within example applications (also see supplemental video).

Texture Painting: Figures 1e, 17 show examples of textures painted in VR using mimicry. The long, smooth, wraparound curves on the torus, are especially hard to draw with 2D interfaces. Our implementation uses Discrete Exponential Maps (DEM) [Schmidt et al. 2006] to compute a dynamic local parametrization around each projected point $q_i$, to create brush strokes or geometric stamps on the object.

Mesh Segmentation: Figures 1e and 18 show mimicry used for interactive segmentation in VR. In our implementation users draw an almost-closed curve $Q = \{q_0, \ldots, q_{n-1}\}$ on the object using mimicry. We snap points $q_i$ to their nearest mesh vertex, and use Dijkstra’s shortest path to connect consecutive vertices, and to close the cycle of vertices. A mesh region is selected or segmented using mesh faces partitioned by these cycles that are easy to draw in AR/VR, but often require view changes and multiple strokes in 2D.

Vector Field Design: Vector fields on meshes are commonly used for texture synthesis [Turk 2001], guiding fluid simulations [Stam 2003], and non-photorealistic rendering [Hertzmann and Zorin 2000]. We use mimicry curves as soft constraints to guide the vector field generation of Fisher et al. [2007]. Figure 19 shows example
We have presented a detailed investigation of the problem of real-
VR, but we preferred to remain application agnostic, rendering the
vector fields, visualized using Line Integral Convolutions [Cabral
participants’ hands. Further, no participants explicitly mentioned
familiar, default controller that matched the physical controller in
impact of controller size by rendering a standard drawing tool in
to a smaller Oculus Touch controller. We could have mitigated the
is also significant: a larger Vive controller for example, has a higher
impact mid-air drawing posture and stroke behavior. Controller size
size of the 3D controllers. Controller grip and weight can certainly
be of little consequence, there was a notable difference in shape and
young, adult males. While the variation in VR headsets seemed to
us to readily recruit 20 diligent users, albeit with a bias towards
hardware and setups. The enthusiasm of the VR community allowed
a more natural study in the wild, with a variety of consumer VR
presented both a challenge and an opportunity to remotely conduct
nascent, with many directions for future work.

vector fields, visualized using Line Integral Convolutions [Cabral
and Leedom 1993] in the texture domain.

8 CONCLUSION
We have presented a detailed investigation of the problem of real-
time inked drawing on 3D virtual objects in immersive environ-
ments. We show the importance of stroke context when project-
ing mid-air 3D strokes, and explore the design space of anchored
projections. A 20-participant remote study showed mimicry to be
preferred over the established spraycan projection for projecting
mid-air strokes on 3D objects in AR/VR. Both mimicry projection
and performing VR studies in the wild do have some limitations.
Further, while user stroke processing for 2D interfaces is a mature
field of research, mid-air stroke processing for AR/VR is relatively
nascent, with many directions for future work.

“In the wild” VR Study Limitations. Ongoing pandemic restrictions
presented both a challenge and an opportunity to remotely conduct
a more natural study in the wild, with a variety of consumer VR
hardware and setups. The enthusiasm of the VR community allowed
us to readily recruit 20 diligent users, albeit with a bias towards
young, adult males. While the variation in VR headsets seemed to
be of little consequence, there was a notable difference in shape and
size of the 3D controllers. Controller grip and weight can certainly
impact mid-air drawing posture and stroke behavior. Controller size
is also significant: a larger Vive controller for example, has a higher
chance of occluding target objects and projected curve, as compared
to a smaller Oculus Touch controller. We could have mitigated the
impact of controller size by rendering a standard drawing tool in
VR, but we preferred to remain application agnostic, rendering the
familiar, default controller that matched the physical controller in
participants’ hands. Further, no participants explicitly mentioned
the controller getting in the way of their ability to draw. Overall,
our study contributes a high-quality VR data corpus comprising
≈ 2400 user strokes, projected curves, intended target curves, and
corresponding VR system states. This data can serve as a bench-
mark for future work in mid-air stroke projection, and data-driven
learning techniques for mid-air stroke processing.

Mimicry Limitations. Our lack of a concise mathematical de-
inition of observed stroke mimicry, makes it harder to precisely
communicate it to users. While a precise mathematical formula-
tion may exist, conveying it to non-technical users can still be a
challenging task. Mimicry ignores controller orientation, produc-
ing smoother strokes with less effort, but can give participants a
reduced sense of sketch control (P2,3,6). We hypothesize that the
reduced sense of control is in part due to the tendency for anchored
smooth-closest-point to shorten the user stroke upon projection,
sometimes creating a feeling of lag. Spraycan like techniques in con-
trast, have a sense of amplified immediacy, and the explicit ability
to make lagging curves catch-up by rotating a controller in place.

Future work. Our goal was to develop a general real-time inked
projection with minimal stroke context via anchoring. Optimizing
the method to account for the entire partially projected stroke may
improve the projection quality. Relaxing the restriction of real-time
inking would allow techniques such as spline fitting and global op-
timization that can account for the entire user stroke and geometric
features of the target object. Local parametrizations such as DEM
(§ 7) can be used to incrementally grow or shrink the projected curve,
so it does not lag the user stroke. Hybrid projections leveraging
both proximity and raycasting are also subject to future work.

On the interactive side, we did experiment with feedback to en-
courage users to draw closer to a 3D object. For example, we tried
varying the appearance of the line connecting the controller to the
projected point based on line length; or providing aural/haptic feed-
back if the controller got further than a certain distance from the
object. While these techniques can help users in specific drawing
or tracing tasks, we found them to be distracting and harmful to
stroke quality for general stroke projection. Bimanual interaction
in VR, such as rotating the shape with one hand while drawing on it
with the other (suggested by P3,19), can also be explored.

Perhaps the most exciting area of future work is employing data-
driven techniques to infer the user-intended projection, perhaps
customized to the drawing style of individual users. Our study code
and data will be made publicly available to aid in such endeavours.

In summary, this paper presents early research on processing and
projection of mid-air strokes drawn on and around 3D objects, that
we hope will inspire further work and applications in AR/VR.

ACKNOWLEDGMENTS
We are thankful to Michelle Lei for developing the initial imple-
mentation of the context-free techniques, and to Jiannan Li and
Debanjana Kundu for helping pilot our methods. We also thank
various 3D model creators and repositories for the models we uti-
lized: Stanford bunny model courtesy of the Stanford 3D Scanning
Repository, trebol model provided by Shao et al.[2012], fertility
model courtesy the Aim@Shape repository, hand model provided
