WARPY: Sketching Environment-Aware 3D Curves in Mobile Augmented Reality

Rawan Alghofaili[*]
George Mason University

Cuong Nguyen
Adobe Research

Vojtěch Krs
Adobe Research

Nathan Carr
Adobe Research

Radomír Měch
Adobe Research

Lap-Fai Yu
George Mason University

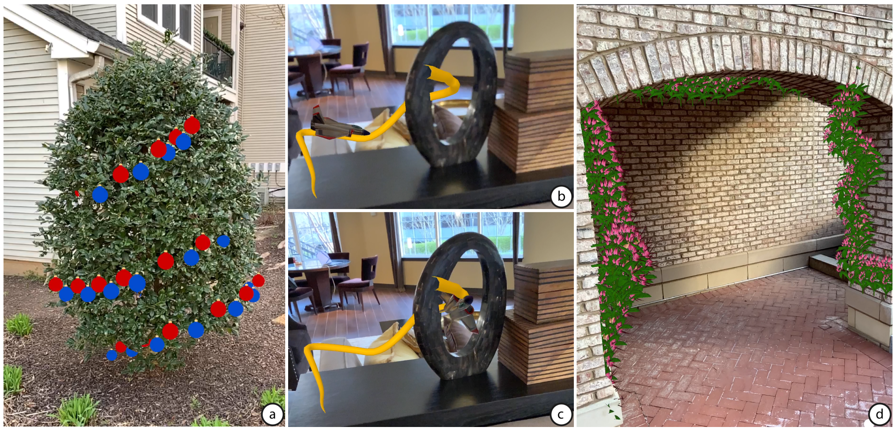


Figure 1: We introduce WARPY, a tablet-based 3D curve drawing tool for AR. Our curve drawing method facilitates the creation of complex (e.g. spirals) and large-scale curves. These curves enable us to add (a) decorative elements to a scene in-situ. They can also be used to guide animations by serving as motion paths. (b–c) For example, a user can draw a path for an airplane that flies through a sculpture. Because a tablet's screen size may restrict the size and angle of curves users can create, we introduce a novel multi-view drawing method that enables users to draw (d) large-scale curves from multiple angles. The user can draw multiple small curves (e.g. at the left, top and right sides of the arc) and our tool will automatically stitch them to form a large curve.

ABSTRACT

Three-dimensional curve drawing in Augmented Reality (AR) enables users to create 3D curves that fit within the real-world scene. It has applications in 3D design, sculpting, and animation. However, the task complexity increases when the desirable path for the curve is obstructed by the physical environment or by what the camera can see. For example, it is difficult to draw a curve that wraps around an object or scales to out-of-reach places.

We propose WARPY, an environment-aware 3D curve drawing tool for mobile AR. Our system enables users to draw freeform curves from a distance in AR by combining 2D-to-3D sketch inference with geometric proxies. Geometric Proxies can be obtained via 3D scanning or from a list of pre-defined primitives. WARPY also provides a *multi-view* mode to enable users to sketch a curve from multiple viewpoints, which is useful if the target curve cannot fit within the camera's field of view. We conducted two user studies and found that WARPY can be a viable tool to help users create complex and large curves in AR.

Index Terms: Computing methodologies—Computer graphics—Graphics systems and interfaces—Mixed / augmented reality

1 INTRODUCTION

The emergence of Augmented Reality (AR) provided a medium for in-situ content authoring and design. Because of their simplicity, sketching interactions were incorporated into authoring and design tools and thus were transferred to AR content authoring and design. Researchers have proposed several solutions to facilitate conceptual design via 3D strokes in AR [1, 24].

There are many benefits to 3D curve creation in AR. The ability to ideate in-situ in real world spaces can unleash designers' creativity and help them achieve their vision more easily. For modeling, a designer could create 3D curves that augment an existing physical object, such as wrapping 3D strings of holiday lights around a tree. For animation, 3D curves can be used as motion paths to guide a virtual character to navigate physical barriers in a real space [46].

Although 3D curves for supporting these applications can be created using desktop software such as Blender or Maya, the design workflow is not as intuitive and contextual as in AR. For example, using 3D scanning, one can capture a proxy of the physical world. This 3D proxy can then be imported into desktop design software where the relevant 3D curves can be drawn. Finally those 3D curves can be exported into an AR application for viewing in the originally captured space. This multi-step process can be tedious and error prone with alignment issues. It also does not lend itself to fast iterative design.

While creating 3D curves in AR is promising, such creations are challenging as the complexity and the scale of the curve increase. The standard solution is mid-air drawing, where a stroke is a geometric realization of the path of a 3D-tracked hand or controller [37]. Although this direct solution is intuitive, it requires the user to navigate the physical environment to create the AR curve. It is not always trivial when the target curve is intended to wrap around an object or scale to out-of-reach places.

Indirect solutions based on sketching on a tablet have been stud-

[*]e-mail: ralghofa@gmu.edu

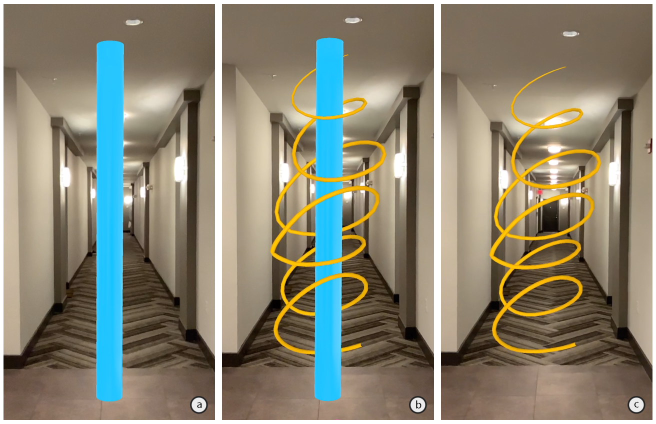


Figure 2: (a) The user placed transient geometry, then (b) drew a curve that wraps around the geometry. (c) This curve can be edited at a later point.

ied [1,22]. These techniques enable users to sketch on a 2D tablet screen, projecting the sketched points onto a planar or a curved surface. These projection-based methods are more practical because the user can stay in one place while creating a large-scale curve in the physical environment [22]. However, planar and curved surfaces limit the type of curves the designer can create. For example, a designer cannot easily create volumetric shapes nor can they use these surfaces to design spiral shapes like a helical spring that wraps around an object in the physical environment.

Nevertheless, indirect AR curve sketching is becoming more accessible to designers. Most modern consumer tablets now support AR out-of-the-box. Compared to AR glasses, tablet-based interactions are also easier to use and can be more precise [2]. Moreover, indirect drawing techniques enable users who have accessibility or mobility issues to still enjoy AR drawing activities that would otherwise be challenging to do using mid-air solutions. Therefore, we aim to explore an indirect drawing technique that can help designers create large-scale and complex depth-varying 3D curves on a mobile AR system.

To realize this goal, we need to project a user's 2D strokes on the tablet into freeform 3D strokes in AR. We utilize the 2D sketch to 3D curve projection method proposed in Skippy [21]. The Skippy system resolves the inherent ambiguity of this operation by analyzing both the negative space around 3D scene geometry and the trajectory of the input 2D strokes. Optimization is run using this information and a smooth 3D path is inferred that wraps in and around the geometry in the scene (Figure 1a and b).

However, simply porting Skippy as an indirect drawing technique in AR would be problematic. Curves drawn in AR must consider the context of the scene to infer the placement of the curve. In the original Skippy system, context information for curve wrapping is provided implicitly from the 3D models in the scene. In AR, the backdrop for the drawing task is the video feed of an environment. It is unclear how 3D geometries should be placed in the environment as context, and how users can interact with them using Skippy to create a desirable curve design in AR.

Moreover, the small field-of-view (FOV) of the tablet might affect the user's ability to draw on the environment context [22]. The original Skippy system operates in a desktop 3D modeling environment where curves can typically fit within a single viewpoint. In AR, when drawing on scenes wider than the tablet's FOV, the user might have to move around to see the whole context. As illustrated in Figure 3, in order to draw the flight path for the airplane, the user must move back farther enough to draw one large curve that swoops around the entire lobby. This approach might be possible to some extent in an open space, but in this scenario, the room's boundaries restrict the user's movement. These challenges necessitate a method that facilitates the creation of large-scale curves from various view angles.

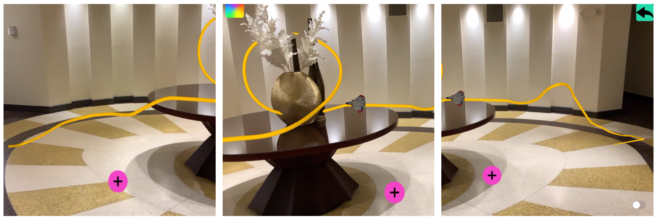


Figure 3: A motion path for a plane shown from multiple views. Due to the limited screen size of the tablet, this curve cannot be created from one view angle.

To address these challenges, we introduce a new drawing technique for AR called WARPY. Our system combines the efficiency of spatial interactions enabled by AR and the intelligent assistance of 2D to 3D sketch inference enabled by Skippy [21]. To use WARPY, there are two main steps. First, the user can add 3D geometric proxies onto the live AR video feed as context for the drawing. The 3D geometry may come from a 3D reconstruction of a physical object in the scene. We develop an interactive 3D scanning pipeline that captures vertex data from the tablet's LiDAR sensor and processes it in an external server to produce a clean watertight mesh that is compatible with the Skippy algorithm. The user can scan and select an interesting physical object in the scene as context, or simply choose from a list of pre-defined primitive geometries. In the second step, the user can start drawing 2D strokes on the tablet screen. Strokes that pass through the added context geometries are processed in real-time using the Skippy algorithm. Thus, using only 2D strokes, users can still create a wide variety of freeform 3D curves in AR (Figure 2).

Additionally, WARPY supports a novel multi-view drawing technique that enables a user to draw large-scale curves in AR. For example, a motion path for an airplane that flies around a room (Figure 3). Multi-view can be enabled with a toggle. In this mode, a user can create segments of the curve from a few viewpoints that are more accessible to her. Then, the system can merge the segments into a desirable 3D curve.

After describing WARPY below, we report on two qualitative user evaluations. We first examine how the two-step interactions in WARPY could be used to support 3D curve drawing tasks in AR. We then further examine an end-to-end drawing experience in WARPY, where participants were asked to perform 3D scanning and then design a complex 3D curve in a large space. The results demonstrate that WARPY is a viable technique for drawing 3D curves in AR, especially for complex and large curves.

2 RELATED WORK

2.1 AR Content Authoring

The ability to interact with the environment makes AR and an excellent medium for design and ideation. We reviewed several recent works which explored the use of 3D interactions to facilitate content authoring in AR.

Saquib et al. [36] introduced a system that enables users to create animations triggered by their body movements. Users can create graphical elements and set triggers in the form of specific hand movements or body poses to initiate their animation. Users map the graphics to the user's body by interacting with a stick figure representation in the system's interface. PoseTween [26] can also attach graphical elements to users' body to create animations triggered by their poses. However, PoseTween allows users to attach the graphical elements on users' body parts in a video of a pose instead of a static stick figure representation. RealitySketch [41] can also enable users to create interactive graphics in AR. Unlike Saquib et al.'s system, users can embed content directly in AR using sketching interactions. Our work is complementary to these existing work. Our approach enables users to create curves that can be used as motion paths for graphical elements.

Users can create animations in AR using a mobile device with ARAnimator [46]. The mobile device's orientation and position are used to create motion paths for a 3D character. Moreover, the character's pose is inferred from the mobile device's orientation during movement with a Support Vector Machine (SVM). Using our tool, users can create motion paths by sketching on a mobile device.

Several interfaces furnish users with the ability to embed content into AR via sketching interactions. Users can sketch content to be placed and animated in AR with Pronto [23]. However, users can only draw on planar surfaces in Pronto. Our approach allows users to draw freeform 3D curves in AR. Ye et al. [45] introduced an AR mid-air curve drawing method using a mobile device. This approach focuses specifically on reducing the tracking error in a mid-air drawing technique. Our approach is an indirect drawing technique. Users can draw 3D curves in the AR environment by drawing on the 2D tablet surface. Our approach focuses less on the precision of the curve and more on enabling users to create complex 3D shapes by utilizing contextual geometries placed in the AR scene.

2.2 3D Sketching with Proxies

Due to its effectiveness and simplicity for design, sketching as an interaction was explored by researchers to facilitate 3D design.

Some systems rely on users creating rough strokes on proxies. SketchingWithHands [20] was introduced to allow users to design with captured first-person hand postures. These captured hand postures are used to design hand graspable and interactive 3D objects. Similarly, Kim et al. [19] enables users to use their hands in the design process. However, instead of capturing the still hand posture for reference, users can move their hands in the air to create *air scaffolds*. Users can attach 2D planar proxies on these scaffolds. Inspired by these ideas, Warpy also allows users to insert 3D proxies as a context for the curve design. This interaction allows users to draw curves that wrap around the proxies more accurately and with a single stroke.

Other 3D curve drawing systems [5, 12, 28, 47] focus on helping users create smooth and clean curves. However, these curves must be defined according to some 2D planes or other curves present in the design. Cohen et al. [10] introduced a free-form curve drawing system that projects users' 2D strokes into 3D curves without the use of proxies. Nevertheless, it relies on users drawing strokes' shadows to compute their projection which may not accelerate the creation of real-time in-situ designs. In-situ sketch-based 3D modeling was explored by Xu et al. [44]. After users load 3D models, Xu et al.'s system generates planes using users' strokes and the 3D models' geometry. Users' strokes are restricted on these 2D planes, while our tool allows users to create free-form curves on 3D geometry.

The work of De Paoli et al. [11] provides an elegant method for sketching in the shell space region surrounding 3D proxy shapes. Arora et al. [3] introduced a projection method that enables mid-air sketching on the surfaces of 3D models in mid-air. The motivation behind our work was to enable sketching not just on, but also in, the spaces between multiple objects real or virtual. For this reason, we opted to follow the approach of Skippy by Krs et al. [21] to project users' 2D strokes directly into 3D curves to allow for real-time in-situ design in AR.

SymbiosisSketch [1] utilizes surface proxies for 3D sketching. Unlike previous systems [8, 19, 20] that use planar surfaces as the drawing proxies, SymbiosisSketch allows curved non-planar surfaces to be used as proxies. Moreover, SymbiosisSketch utilizes a motion capture camera to track a stylus for mid-air sketching which makes it impractical to sketch large-scale designs outside of the confined settings of a lab. We experimented our system with more complex proxies based on full watertight 3D models that users could scan or insert into the environment. Warpy allows users to draw freeform curves that can wrap around these proxies rather than just projecting onto them.

SweepCanvas [24] is a framework for rapidly prototyping objects in 3D with sketch-based interaction. Users sketch pairs of profile and trajectory strokes that define a sweep surface. These sweep surfaces allow users to produce intricate designs in 3D which may easily be transferred to AR. SweepCanvas generates a 3D object from guiding sketch strokes instead of merely providing proxies or sketching guides for users. However, the dependency on profile and trajectory strokes limits the types of objects that can be created using SweepCanvas. For example, users cannot design spiral shapes using SweepCanvas compared to using our tool.

Finally, in Mobi3DSketch [22], users are equipped with a single mobile device to sketch in AR. Similar to SymbiosisSketch, users may create planar or curved surface proxies for better precision. Mobi3DSketch was introduced to address the issues which restricted SymbiosisSketch's usability to be within a lab environment. Contrary to its predecessor, Mobi3DSketch introduced snapping points that allow its users to connect strokes to a surface proxy or other strokes. The snapping points, along with the portability of the sketching device, give users the ability to create large-scale 3D designs in nearly any indoor and outdoor environment. Although both snapping points and Warpy support the creation of large curves, Warpy enables drawing more complex curves by wrapping the drawing around 3D geometries placed in AR. Also, snapping points do not consider the shape of other geometry in the scene while connecting.

2.3 Using 3D Reconstruction for AR Authoring

Several commercial applications aimed to capture the scene geometry and reconstruct it as a 3D mesh [17, 38]. These applications focus on helping users create a high-quality 3D mesh scan. Though we mainly focus on facilitating a mobile authoring experience for users and not a high-resolution scene reconstruction, we took inspiration from prior research into high-fidelity mesh reconstruction in developing Warpy's pipeline [16, 25, 32, 42].

Like Warpy several works have utilized 3D scanning within a content authoring pipeline. For example, Chen et al. [9] enables users to scan a mesh of furniture in a scene and animate them using their body poses. VRFromX [18] allows users to edit a scanned point cloud via mid-air sketching. The point cloud is then used to automatically detect and place 3D models into the scene. Users can create interactive VR experiences by assigning behaviours and affordances to the models in the scene. Similarly, Warpy utilizes the scanned scene geometry to facilitate animation creation by allowing users to draw 3D motion paths on scanned geometry.

3 The Warpy System

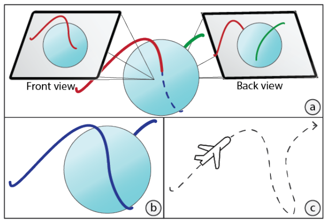
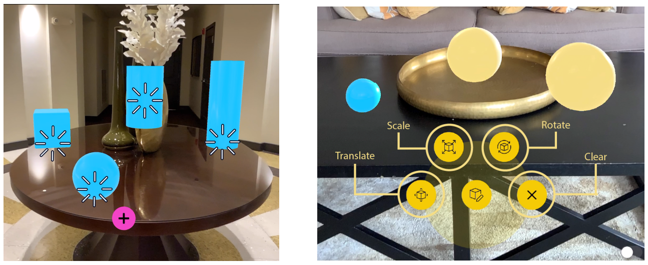


Figure 4: Our method allows a user to draw a curve on a 3D geometry from multiple views in AR. (a) A user can draw multiple curves from different angles. (b) These curves will be connected to form a single overall curve. (c) The resulting curve can be used as an animation motion path.

Figure 4 illustrates our multi-view curve projection pipeline in Warpy. Our tool facilitates real-time curve drawing on proxy geometry in AR using a tablet (2nd generation iPad Pro with a LIDAR scanner). We also introduce a multi-view curve drawing method to enable the creation of curves that span outside of the tablet's view. Figure 2 illustrates how a user would use Warpy to draw 3D curve in AR using a 2D tablet interface. In the first step (Figure 2a), the user places *proxy geometries* in the environment where she would want to draw the 3D curve. Transient geometries, like the cylinder shown in Figure 2, are 3D primitives placed by the user at a detected plane in AR. 3D proxy



(a) Users can place geometries in the scene by tapping on the iPad's screen. The markers indicate the tap locations

(b) Two spheres (in yellow) were selected for editing.

Figure 5: Users can place and edit transient geometry.

geometries, similar to planar surfaces used in Mobi3DSketch [22] or Pronto [23], facilitate the 3D sketching process in the environment. Then, on the tablet, the user can draw curves that wrap or pivot around the placed geometries (Figure 2b). This interaction is akin to wrapping a rope around a pole at a distance. Finally, the curves are automatically converted into 3D in real-time, and the user can inspect or edit the results in AR (Figure 2c).

Figure 6 shows our tool's user interface. The tool enables users to place *geometry proxies* in AR and draw curves on this geometry. Users can edit both the geometry and curves using our tool. Additionally, we introduced a method that facilitates drawing large-scale multi-view curves in the scene.

3.1 Proxy Generation

WARPY relies on geometric proxies to generate the user's curve. These proxies can be in the form of a scanned mesh (i.e. for drawing on physical objects) or primitive geometry (i.e. for drawing mid-air shapes) placed in the scene. Users can also place a combination of both types of geometric proxy for more complex drawing. Before drawing the user will use the proxy generation menu to setup the environment proxies and place transient geometry.

3.1.1 Environment Proxies

The user can capture the environment proxies by first scanning the scene. WARPY will automatically store any mesh or planar geometries detected as the user scans the scene. Any planar geometries that are classified as wall, ceiling or floor will be surrounded by a bounding box with a height of 5cm. These planar bounding boxes will be subtracted from the scanned mesh. The stored planar geometries can be enabled and utilised for drawing by toggling a button.

WARPY enables the user to define boundaries around the object/s that they intend to draw on. WARPY uses the intersection between these boundaries to define the positive space surrounding the object/s. The user can define a boundary by tapping on the screen clockwise or counter-clockwise to place the boundary's edges. The user can place multiple boundaries which will be aggregated to extract the mesh and define the positive space.

WARPY then processes and cleans the extracted mesh using the Blender API [15] to make the watertight mesh needed to compute the SDF. First, WARPY removes any vertices that are a 0.0001m distance or smaller apart and merges them by computing the centroid. The mesh normals are reecalculated so they point outside of the object. Then, the mesh is smoothed with one iteration of Laplacian smoothing ($\lambda = 5e{-}5$). WARPY fills all holes in the mesh, then dissolves vertices and edges to simplify the mesh. Finally, WARPY performs a smooth remeshing operation with an octree of depth 9 to further simplify, fill holes, and smooth the mesh. The watertight mesh produced by the previous step is used to compute the SDF [31]. WARPY computes the sign of the SDF by using the mesh's depth buffers [29].

Finally, to allow users to more easily load and view the mesh, a low-resolution of the mesh is produced by applying the "Decimate" and collapse modifier [15]. We run this operation to maintain 0.3% of the mesh's faces.

3.1.2 Transient Geometry

The user can tap on the tablet's screen to place transient geometry on planes detected in the scene. We cast a ray from the tap location to find the nearest intersection point in the 3D scene. The ray was cast parallel to the camera direction to correctly identify the plane the user intends to place the geometry on. If the intersection point lies on a horizontal plane, we place the geometry upright with the bottom face of the geometry parallel to the plane and the geometry centered around the intersection point. If the plane intersected is vertical, we place the geometry perpendicular to the vertical plane and center the geometry around the intersection point. Figure 5a shows examples of transient geometry placed in the scene.

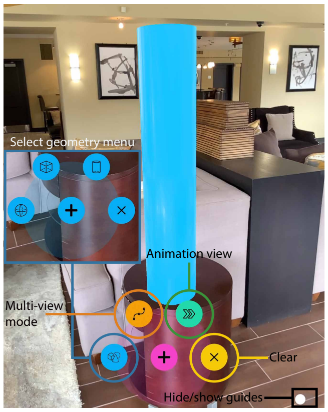


Figure 6: The user interface of our tool. Users can choose to create a curve in multi-view mode; view animations on their curves using the animation view; clear all transient geometries and curves in the environment; and also hide or show the geometry guides (i.e. mesh or transient geometry).

By default, we place cylinders when the user taps on the screen. Nonetheless, the user can choose to change the type of geometry to place in the scene from the main menu of our user interface. In addition to cylinders, the user can select to place cubes and/or spheres. We place the geometries facing the direction of the camera as can be seen in Figure 5a.

We also allow the user to edit the geometry. By tapping on a geometry placed in the scene, the geometry is illuminated to indicate selection. We use the aforementioned ray-casting method used to locate geometry placement location to identify the geometry the user selected to edit. The user is presented with options to scale, translate or rotate the selected geometry. The user can edit the geometry by changing the values on sliders (e.g., scaling a sphere by changing the value of the radius slider). Note that the user may select multiple geometries of the same type to be edited concurrently (Figure 5b).

3.2 Curve Drawing

The user can draw 2D strokes on and around the geometry, which are projected in real-time into 3D curves using the method proposed in Skippy [21]. Similar to Skippy, we first sample vertices regularly on the 2D stroke. Then, we cast a ray from the camera position through each of the vertices. We use sphere tracing with a step size of 0.01 meters along with the signed distance field to efficiently find the intersection points of the rays and the geometry. We halt sphere tracing when we reach a distance of 5 meters from the camera along the ray.

After finding the intersection points on the transient geometry, we create a graph and run the optimization approach proposed in Skippy to produce the curve. Skippy projects the sketches to curves by classifying the curve segments as 'on' and 'off' segments which lie on and off the geometry respectively. Skippy uses non-intersecting rays to define points that are off the geometry. Unless an AR device is being used in an open space, it is unlikely that there will be any non-intersecting rays on the stroke. One could use background planes (e.g., floors, walls, ceilings) to define the 'off' segments, however, plane classification in AR is not yet robust enough to result in noiseless classifications of all points on the stroke. Such

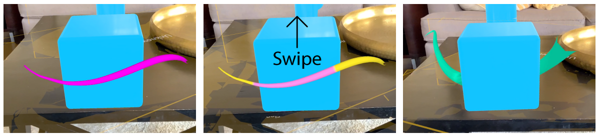


Figure 7: Users can edit curve segments by overdrawing on the curve. Left: the user intended to edit a segment of this curve. Middle: the selected segment was highlighted in pink. Right: the user swiped up on the tablet's screen to move the segment behind the cube.

challenges prompted us to use 3D primitives as our definition of 'on' segments and the rest of the real-world as 'off' segments. Our approach can be extended to allow for drawing on scene geometry without the use of 3D primitives by employing a robust background plane labeling mechanism. We smooth the curves akin to the curve smoothing method utilized in SymbiosisSketch [1].

Due to the interpolation method used in Skippy to connect two 'on' segments with an 'off' segment, some portions of the curve may be placed below planes in the scene. This creates curves that intersect and pass through surfaces like floors, ceilings, tables, or seats. To avoid this problem, we project any points on the curve that lie below any detected plane above it using the plane's normal.

3.3 Curve Editing

We enable the user to edit the curve by moving any 'on' segment backward and forward. We attach a bounding box to each point in an 'on' segment. We set the bounding boxes' lengths, widths and heights to 0.02 meters. The user may select to edit an 'on' segment by drawing over the majority (i.e. more than 50%) of the bounding boxes attached to the segment. Once selected the segment is highlighted as depicted in Figure 7. Then the user may swipe up on the tablet's screen to move the segment back and swipe down to move it forward.

3.4 Multi-view Curve Drawing

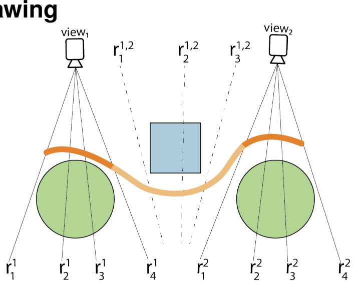


Figure 8: Top-down view of two curves (in dark orange) drawn onto spheres from two views. The curves are stitched to form a final curve. The solid lines denote the original rays from the cameras. The dashed lines denote the interpolated rays. For illustration simplicity, we show a fewer number of rays than what exists in practice.

The limited size of the tablet's screen may restrict the size of the curves the user can draw. Furthermore, the user may not be able to draw curves that span multiple view angles for the same reason. We facilitate the creation of multi-view curves by enabling the user to draw multiple curves that are stitched into a larger curve using our approach. Figure 9 shows an example of a multi-view curve created using this method.

We illustrate our multi-view curve creation method in Figure 8. The user can create a multi-view curve by selecting the multi-view mode in our user interface's main menu as shown in Figure 6. Subsequently, the user can draw the curves $\{C_1, C_2, \ldots, C_M\}$ from multiple views, where M is the number of curves. The curves are arranged in the order of their drawing time. These curves are drawn using the method discussed in Section 3.2 such that each curve C_i is created by casting N rays $\{r_1^i, r_2^i, \ldots, r_N^i\}$, where each ray deviates from the previous ray by 1 degree by default.

We connect the curves in the order they are drawn. In other words, we merge the M curves $\{C_1, C_2, \ldots, C_M\}$ by connecting every curve C_i with the curve drawn in succession C_{i+1}. To connect the curve C_i to the subsequent curve C_{i+1}, we use the last ray r_N^i of C_i and the first ray r_1^{i+1} of C_{i+1}. We perform an interpolation between the rays r_N^i and r_1^{i+1} to create N' interpolated rays $\{r_1^{i,i+1}, r_2^{i,i+1}, \ldots, r_{N'}^{i,i+1}\}$, where each interpolated ray deviates from the previous interpolated ray by 10 degree by default. We run sphere tracing on the interpolated rays to find the intersection points with the transient geometry as explained in Section 3.2.

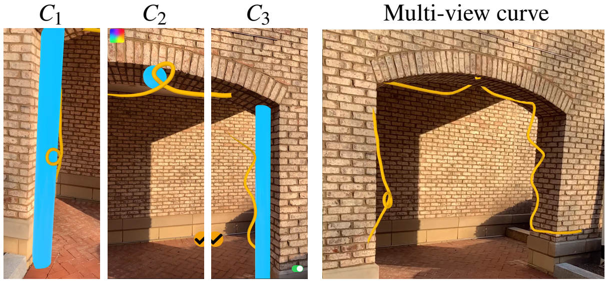


Figure 9: The multi-view curve used to create Figure 1d. This curve was created by stitching the curves C_1, C_2 and C_3 drawn from multiple views.

Finally, we use the original rays (i.e. $\{r_1^i, r_2^i, \ldots, r_N^i\}$ for $i \in [1, M]$) and the intersection points used to create the curves $\{C_1, C_2, \ldots, C_M\}$, together with the interpolated rays (i.e. $\{r_1^{i,i+1}, r_2^{i,i+1}, \ldots, r_{N'}^{i,i+1}\}$ for $i \in [1, M-1]$) and their corresponding intersection points, to form the multi-view curve using the method proposed in Skippy [21].

4 Evaluation of AR Drawing

We conducted a user study to evaluate how our tool could help users draw 3D curves on a mobile AR device. We asked participants to replicate a set of representative ground truth 3D curves in AR with different complexity and scales. We compared our tool with conventional AR drawing methods like mid-air and surface drawing.

As an initial exploration into this new drawing technique, we focus on evaluating the feasibility of WARPY as a 3D drawing tool in AR. Thus, we designed the tasks in this study to be highly structured. We also did not evaluate the 3D scanning component in this study. Participants were only allowed to add simple transient geometries as described in section 4.1. In the next section, we will describe a follow up study where we evaluate the end-to-end workflow in WARPY in an open-ended creative scenario.

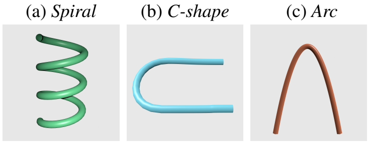

(a) *Spiral* (b) *C-shape* (c) *Arc*

Figure 10: Participants were asked to draw the above curves shown to them in the AR scene.

Participants. We recruited seventeen college and graduate students whose ages ranged from 18-30 to participate in our user study. Among the participants, ten were identified as females and seven as males. None of the participants had a design background or tablet-based drawing experience. Participants provided us with written consent to participate and the study received approval from the Institutional Review Board.

Setup. Participants used an 11-inch second-generation iPad Pro that was running iOS 13.6. Participants were also provided with an Apple Pencil stylus (2nd Generation). We implemented our prototype on the ARKit platform.

Tasks. Participants were asked to draw a set of ground truth curves which are rendered at scale in AR without tracing. We designed three representative curves with different complexity and scale Figure 10. Specifically, the *C-shape* curve has a simple shape and a small size (0.3 meters height). The *Spiral* curve has the most complex shape and a moderate size (0.6 meters height). And finally, the *Arc* curve has a simple shape but the largest size (3.4 meters height).

Baseline System. In previous works, two standard techniques to create 3D curve in AR include mid-air drawing [27, 37] and surface drawing [22, 23]. However, the types of curves that any of these techniques can create are limited. Surface drawing can be used to create

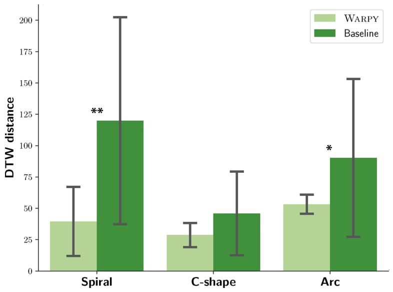


Figure 11: The average Dynamic Time Warping (DTW) distance between the created curves and the ground truth curves. A lower DTW distance means a curve is more similar in shape to the ground truth curve. The error bars represent the standard deviation.

large-scale curves, but it is restricted to a planar surface and therefore cannot be used to create complex multi-depth curves (Figure 10a). Mid-air drawing can be used to create arbitrary complex shapes, but it might not be as robust for creating large-scale curves. Because of the trade-off between flexibility and scale, we equipped the baseline with both mid-air drawing and surface drawing capabilities. To draw in mid-air, the participant entered the mid-air drawing mode in the main menu of the baseline tool. Their curves were placed in mid-air using the position and orientation of the device akin to the mid-air absolute drawing method explored in Mobi3DSketch [22]. For surface drawing, participants can place planar surfaces in the scene by tapping on the iPad's screen. The planar surface is placed using the same method of placing the transient geometry discussed in Section 3.1.2. Strokes drawn on the iPad's screen are projected on the planar surfaces by using the surface-based drawing method used in Mobi3DSketch. Participants can freely switch between these drawing techniques in the study.

Procedure. Participants were first asked to complete a training task for each condition. We showed them the functionalities of our tool, which they could freely explore. Similarly, we demonstrated the baseline tool's features and allowed them to freely explore the tool. We did not constrain the training task time. Participants on average completed training using our tool in ($M = 206.05, SD = SD = 285.98$) seconds while completed training using the baseline in ($M = 213.58, SD = 100.08$) seconds.

Following the training task, participants completed six (6) tasks. In each task, participants completed one of the three drawings shown in Figure 10 using our tool or the baseline. To avoid any carryover effects, we randomly assigned each participant the tool and drawing.

Participants were instructed to replicate the ground truths curve as accurately as possible. When the task begins, the ground truth curve of the task is rendered at scale in AR, and participants were told to draw in the nearby area and without tracing the shape. They were not told which method is best for each curve. Though, they were shown the capabilities of each method (e.g. the ability to create large curves using multi-view, wrap curves around proxies). They could re-draw the curves as many times as they wanted.

After each drawing task, participants rated its the difficulty by answering the following question using a 5-point Likert scale similar to [33] with a rating of 1 (strongly disagree) to 5 (strongly agree): I found the drawing task difficult. Finally, participants freely explored our tool and the baseline and provided feedback on their experience.

4.1 Results

4.1.1 Shape Matching

We conducted a statistical shape analysis to ascertain whether participants were able to correctly re-create the curves shown to them in the user study. First, we used Procrustes analysis [39] to align participants' drawings to the ground truth curves shown in Figure 10. Thereafter, we computed the nearest-neighbor DTW distance [14,30] between each participant's drawing and its corresponding ground truth curve. A lower DTW distance means that a curve is more similar in shape to the ground truth curve. For each trial, participants could draw as many curves as they like. Thus, for the analysis, we aggregated the results only from optimal curves—those with the lowest DTW distance to the corresponding ground truths. Figure 11 shows the average computed distances for each tool in each task.

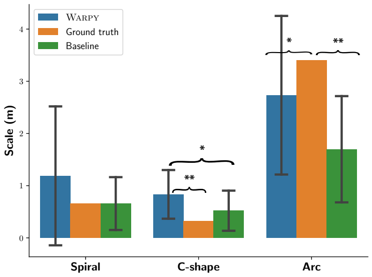


Figure 12: The average scale of curves drawn by participants and the scale of the ground truth curves participants were asked to match.

Participants were able to draw the *Spiral* more accurately (i.e. at a smaller DTW distance from the ground truth) using our tool ($M = 39.55, SD = 28.40$) compared to the baseline ($M = 119.88, SD = 85.07$). We conducted a paired t-test and found a significant difference in the DTW distance of the *Spiral* curves drawn using our tool compared to the baseline ($t(16) = 4.23, p < 0.01$, Cohen's $d = 1.03$). Participants also drew the *Arc* more accurately using our tool ($M = 53.27, SD = 7.9$) compared to the baseline ($M = 90.17, SD = 64.88$), the difference was statistically significant ($t(16) = 2.27, p < 0.05$, Cohen's $d = 0.55$).

For *C-shape*, the average DTW distance of curves drawn using our tool ($M = 28.67, SD = 9.96$) was also lower compared to the baseline ($M = 45.9, SD = 34.37$). However, paired t-test showed no significant difference ($t(16) = 1.83, p = 0.09$).

The DTW distance results indicate that participants were able to match the shape of the ground truth curves with higher accuracy, especially when the task requires interacting with complex curves (e.g. *Spiral*) or with a bigger size (e.g. *Arc*).

4.1.2 Scale Matching

Figure 12 shows the average scale of the curves measured as the length of the diagonal of the curves' bounding boxes. We utilize this metric to assess whether users are able to match their drawings' scale to the reference ground truth curves.

Participants matched the *Arc*'s scale more closely using our tool compared to the baseline. A Friedman test showed a significant difference in the *Arc*'s scale depending on the curve drawing tool used to create the *Arc* ($\chi^2(2) = 11.77, p < 0.01$). A Wilcoxon signed-rank test with Bonferroni correction did not show a significant difference between the scale of *Arc*s drawn using our tool ($M = 2.73, SD = 1.57$) compared to the scale of the ground truth *Arc* shown to participants during the user study (3.4) ($W = 1.73, p = 0.08$). Conversely, we found a significant difference between the average scale of the *Arc*s drawn using the baseline ($M = 1.70, SD = 1.05$) compared to the ground truth *Arc* ($W = 3.48, p < 0.01, r = 0.42$) and to our tools' curves ($W = 2.27, p < 0.05, r = 0.55$).

The Friedman test found no significant difference in the scale of the *Spiral* curves according to the tool used ($\chi^2(2) = 4.59, p = 0.10$). Participants used our tool to draw *Spiral*s with an average scale of ($M = 1.19, SD = 1.37$) meters using our tool and ($M = 0.66, SD = 0.52$) meters using the baseline. These measures were not statically different from the ground truth curve's scale of 0.66 meters.

Participants drew the *C-shape* curves using our tool ($M = 0.83, SD = 0.48$) at a larger scale than the curve shown to them during the study (0.32). On the other hand, participants were able to draw the curves using the baseline ($M = 0.52, SD = 0.40$) at a similar scale to the ground truth *C-shape*. A Friedman test

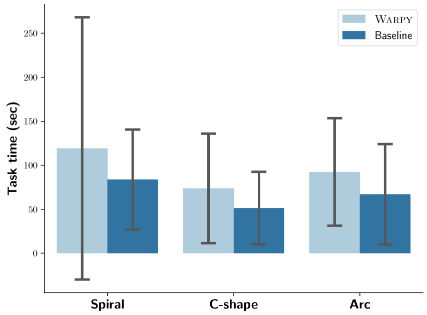


Figure 13: The average time taken by participants to complete the user study curve creation tasks.

showed an effect of the drawing tool used on the *C-shape*'s scale ($\chi^2(2) = 14.59, p < 0.01$). The post-hoc test showed a significant difference between the scale of the *C-shape* created using our tool compared to the ground truth ($W = 3.48, p < 0.01, r = 0.84$) and compared to the baseline ($W = 2.44, p < 0.05, r = 0.59$). However, we did not find a statistical difference between the scale of the curves drawn using the baseline and the ground truth ($W = 1.49, p = 0.14$).

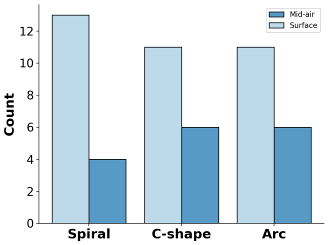


Figure 14: The drawing methods participants used in the baseline.

These results suggest that the scale of the curves created in our approach also closely match the scale of the ground truth curves, especially when the task requires replicating a large curve (e.g. the *Arc* is at 3.4 meters height). Moreover, participants in the baseline condition could better match the scale when the ground truth curves are simple and small (e.g. the *Spiral* and *C-shape* tasks, respectively).

The baseline condition log revealed that most participants opted to use mid-air drawing (Figure 14). Previous research has shown that mid-air drawing tends to be less accurate compared to drawing on a physical surface like the tablet screen in our approach [2, 43]. For the *Spiral* and *Arc* curves, participants might have had to maneuver the tablet at an undesirable motor distance (e.g. reaching or circling motion), which can introduce further inaccuracies.

4.1.3 Task Completion Time

We also recorded the total amount of time in seconds participants took to complete the tasks using each tool as shown in Figure 13.

Overall, participants took slightly longer time to complete tasks when using our tool. We conducted a paired t-test and found no significant differences in the task completion time for all drawing tasks using our tool compared to the baseline. Specifically, participants completed the *Spiral* drawing task in an average of ($M = 119.13, SD = 153.56$) seconds using our tool and in an average of ($M = 83.68, SD = 58.58$) seconds using the baseline ($t(16) = 0.97, p = 0.35$). For *C-shape*, participants completed the task in ($M = 73.72, SD = 64.04$) seconds using our tool and in ($M = 51.31, SD = 42.51$) seconds using baseline ($t(16) = 1.20, p = 0.25$). For *Arc*, participants completed the task in ($M = 92.32, SD = 62.92$) seconds using our tool and in ($M = 66.97, SD = 58.74$) seconds using baseline ($t(16) = 1.13, p = 0.28$).

4.1.4 Subjective Difficulty Ratings

Figure 15 visualizes participants' answer to the question *"I found the drawing task difficult."* A Wilcoxon signed-rank test found no significant difference between the difficulty of the *Spiral* drawing task using our tool ($Md = 3$ neutral) compared to the baseline ($Md = 2$ disagree) ($W = 1.76, p = 0.08$).

Participants were neutral about the difficulty of the *C-shape* drawing task using our tool ($Md = 3$ neutral) but reported a rating of 1 for the baseline ($Md = 1$ strongly disagree). The Wilcoxon signed-rank test also found a significant difference between the *C-shape* difficulty ratings ($W = 2.84, p < 0.01, r = 0.69$). Similarly, participants were neutral about the difficulty of the *Arc* drawing task with our tool ($Md = 3$ neutral) while rating the task as easier using the baseline ($Md = 1$ strongly disagree). We found a significant difference between participants' ratings while using our tool compared to the baseline ($W = 2.35, p < 0.05, r = 0.57$).

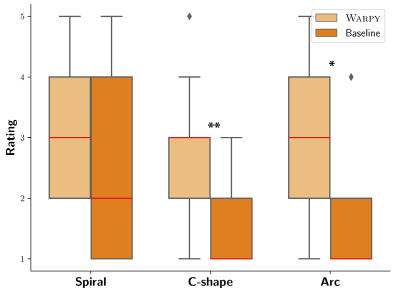


Figure 15: Participants answered the question *"I found the drawing task difficult"* with a rating of 1 (strongly disagree) to 5 (strongly agree) after each task. The diamonds show outliers. The error bars represent the Interquartile Range (IQR).

Even though participants did not report drawing with our tool to be difficult, they found surface drawing and mid-air drawing methods to be more intuitive for simple shapes such as the *Arc*. Several participants ($P4, P11, P17$) voiced that they enjoyed drawing on 3D geometry but they needed more practice to get familiar with the technique compared to using surface and mid-air drawing.

4.2 Discussion

In this first study, we seek to understand how the drawing interaction supported by Warpy would be used in AR. We found that our system enabled participants to create challenging 3D curves that are either comparable or in some cases better than those produced by standard AR methods like mid-air drawing or tablet-based drawing.

More specifically, when the target curves are complex, such as those in the *Arc* and *Spiral* tasks, our approach enabled users to match the shape of the target more closely. Additionally, for scale matching, we found that for a large curve like *Arc*, participants using our approach were able to better match the scale of the target. At 3.4 meters, the *Arc* curve was tall enough to be out of reach for participants who used the mid-air technique, so most participants in the baseline condition created a smaller replication of the Arc. In contrast, participants using our approach benefited from being able to draw from a distance, which allowed them to have a better view to compare and validate their curves with the ground truth.

For simple curves, such as in the *C-shape* task, the benefit of our approach is not as prominent compared to the baseline condition. The *C-shape* target was small and flat so participants could easily trace it accurately using either the mid-air or the surface drawing technique in the baseline condition. In contrast, in our approach, participants could replicate the shape of the *C-shape* curve well, but the scale of the curves is generally much larger than the ground truth curve. We observed that participants placed a cylinder in the task and then stepped back at a distance to draw the curve that would be interpolated around the cylinder to create the *C-shape* curve. As they were farther away, it was likely that the interpolated results might have been larger and/or curved away from the geometry differently than they expected.

5 Evaluation of the Entire Workflow

The results from the first study validated that users could use Warpy as a drawing tool in AR. However, the first study did not evaluate the entire workflow of Warpy. Thus, we conducted a second study to evaluate how users can utilize the scene geometry in a physical space to engage in a creative task. We asked participants to

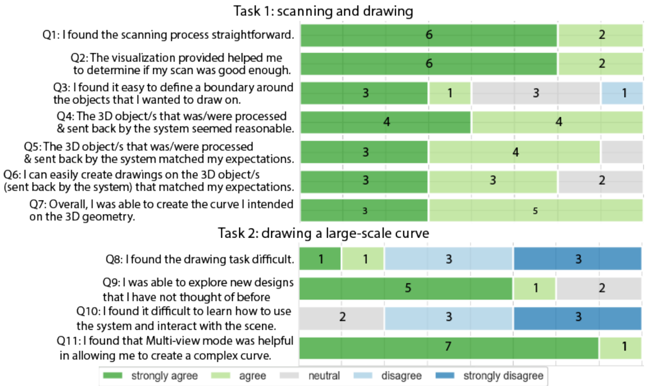


Figure 16: Results of the survey conducted after our second study.

use both the scanning and drawing components in WARPY to design a 3D animated roller coaster track in a community lounge.

We conducted two smaller design sessions within this study. In the first session, participants were asked to scan an arbitrary object in the room and then design the roller coaster track based on that object. In the second session, participants were asked to design a roller coaster track around a large pre-scanned pool table. We chose this study design to evaluate the scanning and the drawing experiences separately. Due to the experimental nature of our mesh processing server, scanning the large pool table might take participants a few trials to complete. Thus we did not ask them to scan the pool table and instead ask them to focus on the creative task.

Participants. Eight college and graduate students whose ages ranged from 23-29 participated in this study. Three participants identified as female and five as male. Participants provided us with a written consent to participate, and the study received approval from the Institutional Review Board.

Setup. We utilized the same setup as our first AR drawing study. Additionally, because this task is situated in a large physical space, participants might have to traverse the environment more. Thus, we enabled only stylus drawing to allow participants to more easily maneuver the iPad without accidentally drawing any curves.

Tasks. We asked participants to complete two free-form design sessions. In both sessions, participants were asked to create a 3D track for a roller coaster animation in AR. In the first session, participants tested the entire WARPY by scanning an object in the environment and then creating a 3D curve on the scanned object. In the second session, participants were only asked to design the roller coaster track for a pre-scanned pool table (supplementary material). In this session, participants were encourage to create a large design.

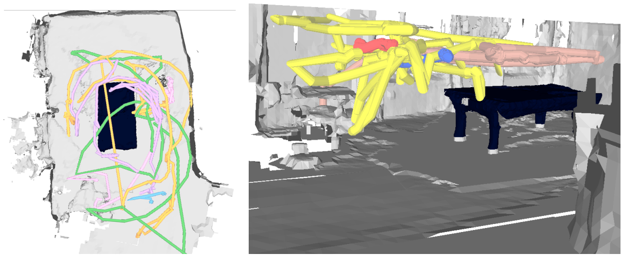
Procedure. Akin to our previous study, participants first completed a training task before starting the study. They were shown the full functionality of WARPY and given the opportunity to practice the drawing interaction. We used the pre-scanned pool table mesh in the training phase. After the training task, participants completed both sessions. Following the completion of each session, they completed a survey to rate their experience using a 5-point Likert scale with a rating of 1 (strongly disagree) to 5 (strongly agree).

We measured participant's movement during the task by recording the *xyz* 3D coordinate of the iPad device. We also saved participants' final roller coaster design. For the qualitative survey, participants answered the questions shown in Figure 16.

5.1 Results

5.1.1 Session 1: Scan and draw

In Session 1, we asked participants to pick an arbitrary object in the room, scan and then draw a roller coaster track based on it. The goal of this session was to evaluate the role of the scanning component in helping users draw in WARPY. Our supplementary material contains example objects that participants scanned and processed by WARPY.



(a) *P*2, *P*5, *P*7, and *P*8 (b) *P*1, *P*3, *P*4, and *P*6

Figure 17: Movements of participants.

Figure 16 (top) shows the participants' ratings of several aspects of the workflow. Overall, participants agreed that the scanning process was straightforward (8/8, Q1) and the returned scan results matched their expectations (8/8 in Q4 and 7/8 in Q5).

Capturing a 3D scan using an AR device is not a trivial task. Scanning applications need to provide some visual guidance to users [35]. WARPY also provides an AR bounding box technique to help users capture the desirable model. All participants agreed that the visualization was helpful (8/8, Q2). However, only four participants found it easy to define the bounding box for the scan processing server (Q3). In the post-study interview, participants reported small drifting issues in the AR device caused the confusion (P3, P4, P6). When the bounding box moves due to drifts, participants had to walk around the object to re-inspect all the boundaries.

Nevertheless, participants were able to scan and draw the intended curves. Participants agreed that they could create a drawing based on the scanned object (6/8, Q6) and that they were able to create the curve they intended to (8/8, Q7). Participants explained that the strength of the technique is the ability to interact with real-world objects using virtual interactions, "*It was impressive to interact with actual objects in the real world with my virtual interactions*" (P2). P5 shared similar remarks: "*I can get an automatically completed trajectory by simply drawing a few strokes, considering the complex geometry in the 3d environment*". Participants also shared thoughts on improving the drawing technique by increasing the rendering speed (P4 "*Sometimes the lag between my drawing and the rendering confuses me. I don't know if I should repeat the stroke again*") and providing more fine-grain editing options (P6, "*It is hard that we need to clear everything after failing and start over*").

5.1.2 Session 2: drawing a large curve

In the second part of the study, we asked participants to focus only on creating a large roller coaster track in a big space. We provided participants with a pre-scanned mesh of a big pool table in the middle of a lounge area. Figure 16 (bottom) shows an overview of participants' ratings of their drawing experience. Overall, participants did not find the drawing task difficult (6/6 rated, Q8).

This is a surprising finding as participants produced curves that were mostly complex in style and large in size (Figure 18). The resulting roller coaster tracks interact with the pool table in a variety of ways. For example, P7 and P3 created tracks that loops underneath the pool table. P2 and P6 created tracks with more height, making loops that were close to the ceiling or to the decorative ceiling lights. Six participants ranked their agreement highly to the statement "*I was able to explore new designs that I have not thought of before*".

Looking further into the creative process behind the drawings. Six participants rated that it was not too difficult to interact with the scene (6/6, Q10). We looked into participants' movement in the study to examine their strategies in interacting with the pool table. Figure 17a and 17b shows two distinct movement patterns observed in the study. One group of participant clustered in one corner of the room, another group wandered around. Both groups did not cross the pool table or try to raise the iPad toward the ceiling. Yet, they are able to produce the complex designs in Figure 18. These results show that they were able to use WARPY to create these 3D designs comfortably from a distance using 2D strokes on the tablet.

Participants leveraged transient geometries to control the curve segments outside of the mesh. For example, P2 placed a sphere on the pendant light above the pool table and used it to draw a loop segment of the curve. This resulted in a roller coaster path that travels from the pool table to the light. P2 and P6 also placed large cylinders to increase their curve heights.

The Multi-view mode was instrumental in helping participants complete the design task. Seven participants ranked highly their agreement to the statement "*I found that Multi-view mode was helpful in allowing me to create a complex curve*". In the post-study interview, P1 and P2 explained that the Multi-view mode helps make the drawing task easier and more creative because they do not have to worry about filling every corners or every gaps in the scene. P5 explained further: "*Drawing directly on 3d is quite challenging due to some occlusion, and this technique really helps.*"

5.2 Discussion

The drawing interaction in Warpy is essentially 2D, but most of our participants felt satisfied about the 3D curves they created and felt they could design curves that interact with real-world objects. This is an important finding as a well-known problem of 2D drawing is the difficulty in expressing 3D shapes [6]. In AR, it is possible to create expressive 3D curves, but users have to use more direct approaches that would require them to move through the physical space. This assumption is not always feasible if the user has limited mobility due to the environment (i.e. having to crouch down to draw under the pool table) or their own accessibility needs (i.e., being on a wheelchair). With Warpy, users can draw indirectly from a tablet. A majority of the 3D shapes are inferred in real-time from their 2D strokes, helping to reduce the range of motion required to accomplish the full design. Thus, our technique provides a viable alternative to make creative 3D drawing in AR more accessible.

The Multi-view mode also assist users by softening the physical restrictions the users may face. An implicit assumption about 3D drawing in AR is that a user has a good view of the drawing target on the AR display. When creating a 3D curve that spans over a large distance, such as a roller coaster track that go from one end of the pool table to another, the user must move the AR device to maintain a view of the object for the entire time. This requirement could make drawing more tedious and error-prone. The Multi-view mode softens this constraint by enabling user to create smaller segments of the curve and then having them merged later. Furthermore, this mode encourages users to change their viewpoint which allows them to better plan their next stroke [7]. This leads to a quicker and more enjoyable experience.

To sum, the results from the two sessions in this study further show that Warpy is a viable technique for designing environment-aware 3D curves in AR. Warpy gives users a set of tools such as environment proxy, 2D-to-3D curve inference, and Multi-view to design 3D curves using 2D strokes. These tools work together in a streamline workflow to soften the mobility and viewpoint constraints that users may face when drawing in AR, giving users more creative power to design freeform 3D curves in an environment.

6 Limitations and Future Work

Our workflow currently requires users to manually define the boundaries of object/s in the scan that they intend to draw on. By incorporating scene semantic segmentation [34], users can simply select the objects by tapping on the screen and casting a ray to find the closest point cloud semantic cluster. This cluster can be thereafter used to generate the final mesh.

In some cases like the curve created by P3 in Figure 18, the curve may intersect with the edges of the scanned proxies. This visual artifact is due to a slight imprecision in the scan that spilled-over to the computation of the SDF, which in turns affect the interpolation in the Skippy algorithm. By implementing better mesh reconstruction and processing mechanisms and/or utilizing a more robust SDF computation, we can avoid these artifacts.

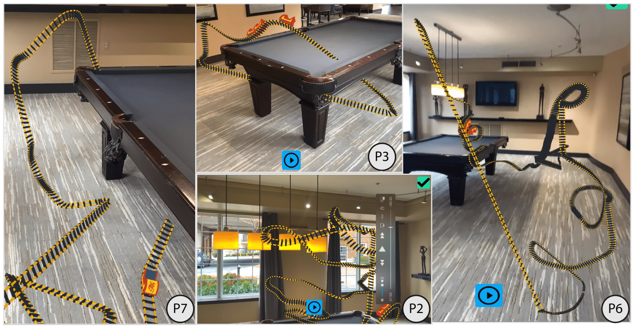


Figure 18: Curves created using Warpy animated into roller coaster tracks. The tracks are rendered and the carts animated by computing the Frenet–Serret frames from the curve points.

Warpy does not allow for fine-grained control over the optimized curve results. It currently only provides a simple edit tool for users to modify whether a curve segment should be in front or behind the geometry. We believe developing more advanced interfaces for a user to control the output of the interpolated results will make our system even more practical. Some examples of edit tools include supporting drawing more than one curve and fine-tuning the smoothness and shapes of individual segments in the curve.

Finally, our approach was implemented on a mobile AR device. Our findings are generally applicable to other tablet-based content creation applications in AR and VR [1, 13, 40]. However, we have not evaluated our approach on a more immersive platform like AR glasses. One unique advantage of AR glasses compared to mobile AR is more robust 6DOF hand tracking while freeing the user's hand from holding the tablet. One interesting direction is to explore how spatial and tangible interactions around the tablet could be integrated to reinforce our 3D curve creation workflow [4].

7 Conclusion

We introduced Warpy, a 3D curve drawing tool for Augmented Reality. Warpy uses a 2D-to-3D sketch inference method and geometric proxies to facilitate in-situ curve creation. Warpy let users create geometric proxies by scanning physical objects or placing virtual primitives in AR. Warpy also provides a novel *Multi-view* drawing technique to address the challenges that arise when drawing large-scale curves that do not fit within an AR camera's field of view.

We conducted two user studies to evaluate Warpy. We first compared Warpy with a baseline mid-air drawing technique in a structured drawing task. We found that for curves with challenging shapes such as a spiral or a tall arc, our system can serve as a viable AR drawing technique. Encouraged by this result, we conducted a follow up study to evaluate an end-to-end design experience where users can use Warpy to create a large 3D curve design for a physical space. We found the design of the curves that users created interact well with the geometry in the environment and users felt the experience was satisfactory and enjoyable.

Designing AR spaces in-situ provides numerous challenges in accessibility due to scale, structure, and variety of physical environments. Our work circumvents this issue by proposing a computer assisted curve drawing system which we hope can make the AR creation process more approachable for all.

Acknowledgments

This project was supported by NSF grants (award numbers: 1942531 and 2128867). We would like to thank Adobe Research for their generous donations to GMU's DCXR group. We would also like to thank the anonymous reviewers for their constructive feedback.

References

[1] R. Arora, R. Habib Kazi, T. Grossman, G. Fitzmaurice, and K. Singh. *SymbiosisSketch: Combining 2D & 3D Sketching for Designing Detailed 3D Objects in Situ*, p. 1–15. Association for Computing Machinery, New York, NY, USA, 2018.

[2] R. Arora, R. H. Kazi, F. Anderson, T. Grossman, K. Singh, and G. Fitzmaurice. Experimental evaluation of sketching on surfaces in vr. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, p. 5643–5654. Association for Computing Machinery, New York, NY, USA, 2017.

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

[4] D. Avrahami, J. O. Wobbrock, and S. Izadi. Portico: Tangible interaction on and around a tablet. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, p. 347–356. Association for Computing Machinery, New York, NY, USA, 2011. doi: 10.1145/2047196.2047241

[5] S.-H. Bae, R. Balakrishnan, and K. Singh. 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, p. 151–160. Association for Computing Machinery, New York, NY, USA, 2008. doi: 10.1145/1449715.1449740

[6] S.-H. Bae, R. Balakrishnan, and K. Singh. 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, p. 151–160. Association for Computing Machinery, New York, NY, USA, 2008. doi: 10.1145/1449715.1449740

[7] M. D. Barrera Machuca, W. Stuerzlinger, and P. Asente. The effect of spatial ability on immersive 3d drawing. In *Proceedings of the 2019 on Creativity and Cognition*, C&C '19, p. 173–186. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3325480.3325489

[8] M. D. Barrera Machuca, W. Stuerzlinger, and P. Asente. Smart3dguides: Making unconstrained immersive 3d drawing more accurate. In *25th ACM Symposium on Virtual Reality Software and Technology*, VRST '19. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3359996.3364254

[9] J. Chen, S. Izadi, and A. Fitzgibbon. *KinÊTre: Animating the World with the Human Body*, p. 435–444. Association for Computing Machinery, New York, NY, USA, 2012.

[10] J. M. Cohen, L. Markosian, R. C. Zeleznik, J. F. Hughes, and R. Barzel. An interface for sketching 3d curves. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, I3D '99, p. 17–21. Association for Computing Machinery, New York, NY, USA, 1999. doi: 10.1145/300523.300655

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

[12] T. Dorta, G. Kinayoglu, and M. Hoffmann. Hyve-3d and the 3d cursor: Architectural co-design with freedom in virtual reality. *International Journal of Architectural Computing*, 14(2):87–102, 2016. doi: 10.1177/1478077116638921

[13] T. Drey, J. Gugenheimer, J. Karlbauer, M. Milo, and E. Rukzio. Vrsketchin: Exploring the design space of pen and tablet interaction for 3d sketching in virtual reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, p. 1–14. Association for Computing Machinery, New York, NY, USA, 2020.

[14] A. Efrat, Q. Fan, and S. Venkatasubramanian. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *J. Math. Imaging Vis.*, 27(3):203–216, Apr. 2007. doi: 10.1007/s10851-006-0647-0

[15] B. Foundation. Blender python api.

[16] R. Hanocka, G. Metzer, R. Giryes, and D. Cohen-Or. Point2mesh: A self-prior for deformable meshes. *ACM Trans. Graph.*, 39(4), jul 2020. doi: 10.1145/3386569.3392415

[17] A. Inc. Create 3d models with object capture.

[18] A. Ipsita, H. Li, R. Duan, Y. Cao, S. Chidambaram, M. Liu, and K. Ramani. *VRFromX: From Scanned Reality to Interactive Virtual Experience with Human-in-the-Loop*. Association for Computing Machinery, New York, NY, USA, 2021.

[19] Y. Kim, S.-G. An, J. H. Lee, and S.-H. Bae. Agile 3d sketching with air scaffolding. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18. Association for Computing Machinery, New York, NY, USA, 2018.

[20] Y. Kim and S.-H. Bae. 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, p. 797–808. Association for Computing Machinery, New York, NY, USA, 2016. doi: 10.1145/2984511.2984567

[21] V. Krs, E. Yumer, N. Carr, B. Benes, and R. Měch. Skippy: Single view 3d curve interactive modeling. *ACM Trans. Graph.*, 36(4), July 2017. doi: 10.1145/3072959.3073603

[22] K. C. Kwan and H. Fu. Mobi3dsketch: 3d sketching in mobile ar. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, p. 1–11. Association for Computing Machinery, New York, NY, USA, 2019.

[23] G. Leiva, C. Nguyen, R. H. Kazi, and P. Asente. Pronto: Rapid augmented reality video prototyping using sketches and enaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, p. 1–13. Association for Computing Machinery, New York, NY, USA, 2020.

[24] Y. Li, X. Luo, Y. Zheng, P. Xu, and H. Fu. Sweepcanvas: Sketch-based 3d prototyping on an rgb-d image. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, p. 387–399. Association for Computing Machinery, New York, NY, USA, 2017. doi: 10.1145/3126594.3126611

[25] Z. Li, P. C. Gogia, and M. Kaess. Dense surface reconstruction from monocular vision and lidar. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6905–6911, 2019. doi: 10.1109/ICRA.2019.8793729

[26] J. Liu, H. Fu, and C.-L. Tai. Posetween: Pose-driven tween animation. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, p. 791–804. Association for Computing Machinery, New York, NY, USA, 2020. doi: 10.1145/3379337.3415822

[27] L. Logical Animal. Lightspace - 3d painting in ar, Sep 2017.

[28] M. D. B. Machuca, P. Asente, W. Stuerzlinger, J. Lu, and B. Kim. Multiplanes: Assisted freehand vr sketching. In *Proceedings of the Symposium on Spatial User Interaction*, SUI '18, p. 36–47. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3267782.3267786

[29] M. K. (marian42). Calculate signed distance fields for arbitrary meshes.

[30] R. Niels. Dynamic time warping: An intuitive way of handwriting recognition? 2004.

[31] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *CoRR*, abs/1901.05103, 2019.

[32] S. Peng, C. M. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger. Shape as points: A differentiable poisson solver. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[33] K. Pfeuffer and H. Gellersen. Gaze and touch interaction on tablets. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, p. 301–311. Association for Computing Machinery, New York, NY, USA, 2016. doi: 10.1145/2984511.2984514

[34] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[35] A. Sankar and S. M. Seitz. Interactive room capture on 3d-aware mobile devices. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, pp. 415–426. ACM, New York, NY, USA, 2017. doi: 10.1145/3126594.3126629

[36] N. Saquib, R. H. Kazi, L.-Y. Wei, and W. Li. Interactive body-driven graphics for augmented video performance. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2019.

[37] S. Schkolne, M. Pruett, and P. Schröder. 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, p. 261–268. Association for Computing Machinery, New York, NY, USA, 2001. doi: 10.1145/365024.365114

[38] E. G. S. s.r.o. Realitycapture: Mapping and 3d modeling photogeometry.

[39] M. B. Stegmann and D. D. Gomez. A brief introduction to statistical shape analysis, mar 2002. Images, annotations and data reports are placed in the enclosed zip-file.

[40] H. B. Surale, A. Gupta, M. Hancock, and D. Vogel. Tabletinvr: Exploring the design space for using a multi-touch tablet in virtual reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, p. 1–13. Association for Computing Machinery, New York, NY, USA, 2019.

[41] R. Suzuki, R. H. Kazi, L.-Y. Wei, S. DiVerdi, W. Li, and D. Leithinger. Realitysketch: Embedding responsive graphics and visualizations in ar with dynamic sketching. In *Adjunct Publication of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20 Adjunct, p. 135–138. Association for Computing Machinery, New York, NY, USA, 2020. doi: 10.1145/3379337.3415892

[42] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss. Poisson Surface Reconstruction for LiDAR Odometry and Mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[43] P. Wacker, A. Wagner, S. Voelker, and J. Borchers. Physical guides: An analysis of 3d sketching performance on physical objects in augmented reality. In *Proceedings of the Symposium on Spatial User Interaction*, SUI '18, p. 25–35. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3267782.3267788

[44] P. Xu, H. Fu, Y. Zheng, K. Singh, H. Huang, and C.-L. Tai. Model-guided 3d sketching. *IEEE Transactions on Visualization and Computer Graphics*, 25(10):2927–2939, 2019. doi: 10.1109/TVCG.2018.2860016

[45] H. Ye, K. Kwan, and H. Fu. 3d curve creation on and around physical objects with mobile ar. *IEEE Transactions on Visualization & Computer Graphics*, (01):1–1, jan 5555. doi: 10.1109/TVCG.2020.3049006

[46] H. Ye, K. C. Kwan, W. Su, and H. Fu. Aranimator: In-situ character animation in mobile ar with user-defined motion gestures. 39(4), July 2020. doi: 10.1145/3386569.3392404

[47] E. Yu, R. Arora, T. Stanko, J. A. Bærentzen, K. Singh, and A. Bousseau. *CASSIE: Curve and Surface Sketching in Immersive Environments*. Association for Computing Machinery, New York, NY, USA, 2021.