Multiplanes: Assisted Freehand VR Sketching

Mayra D. Barrera Machuca^1, Paul Asente^2, Wolfgang Stuerzlinger^1, Jingwan Lu^2, Byungmoon Kim^2
^{1}SIAT, Simon Fraser University, Vancouver, Canada, ^{2}Adobe Research, San Jose, USA
^{1}{mbarrera, w.s}@sfu.ca, ^{2}{asente, jlu, bmkim}@adobe.com

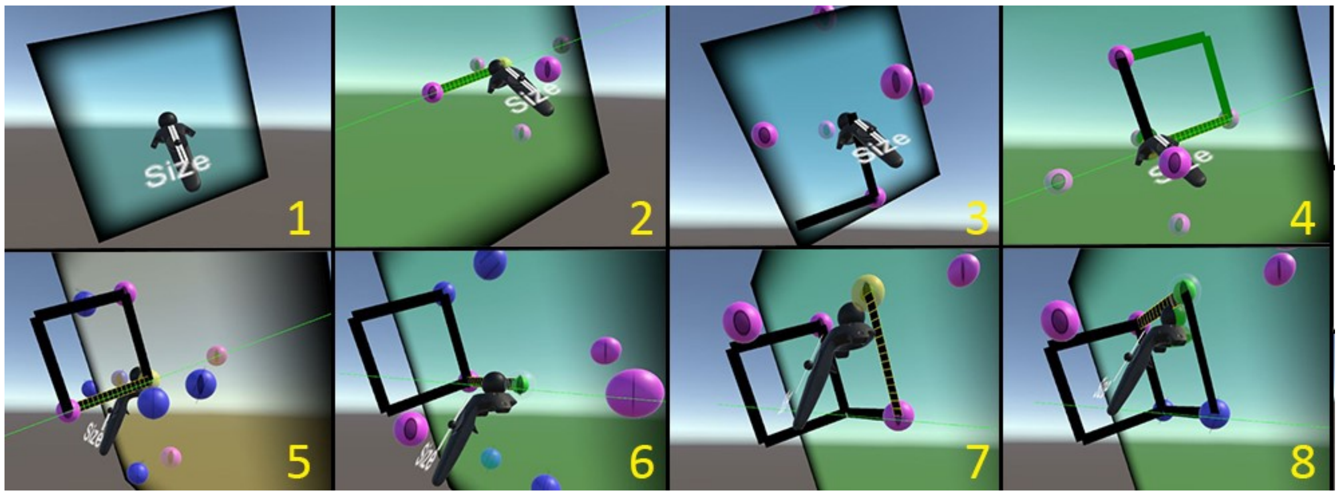

Figure 1: Step-by-step process of drawing two adjacent faces of a cube with Multiplanes. First the user draws a square on a plane in space (images 1-5) and then a second square (images 6-8) on an orthogonal plane that is suggested based on the controller orientation. Beautification trigger points help the user to keep stroke length uniform. Stroke beautification ensures straight lines.

ABSTRACT

The presence of a third dimension makes accurate drawing in virtual reality (VR) more challenging than 2D drawing. These challenges include higher demands on spatial cognition and motor skills, as well as the potential for mistakes caused by depth perception errors. We present Multiplanes, a VR drawing system that supports both the flexibility of freehand drawing and the ability to draw accurate shapes in 3D by affording both planar and beautified drawing. The system was designed to address the above-mentioned challenges. Multiplanes generates snapping planes and beautification trigger points based on previous and current strokes and the current controller pose. Based on geometrical relationships to previous strokes, beautification trigger points serve to guide the user to reach specific positions in space. The system also beautifies user's strokes based on the most probable intended shape while the user is drawing them. With Multiplanes, in contrast to other systems, users do not need to manually activate such guides, allowing them to focus on the creative process.

SUI '18, October 13–14, 2018, Berlin, Germany
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5708-1/18/10..$15.00
https://doi.org/10.1145/3267782.3267786

CCS CONCEPTS

• Human-centered computing: Virtual Reality, User Studies, User Interface Design

ADDITIONAL KEYWORDS AND PHRASES

Virtual Reality Drawing; 3D User Interfaces; Depth Perception; Spatial Cognition

1 INTRODUCTION

High quality Virtual Reality (VR) devices, such as the HTC Vive and the Oculus Rift, are now easily available to the public. These products have rekindled interest in using VR technologies in the design process by letting users draw directly in a 3D virtual environment. An example of a commercial product for 3D drawing is Tilt Brush [12]. Most of these tools build on the freehand drawing technique, in which the stroke follows the controller position. Although this technique provides an intuitive and effective method of conceptualizing new shapes, which helps in the creative process [32], prior work shows that the resulting drawings are less accurate than 2D sketches [2,25]. One possible explanation is that depth perception errors prevent the user from realizing the desired stroke, because they cannot see accurately enough where they are drawing in space – especially in relation to other content [2,30]. Another possible reason is the higher difficulty of the task, as the additional third dimension could result in higher cognitive and sensorimotor demands [33]. Finally, the absence of a physical surface also affects drawing accuracy [2].

In this paper, we present a system called Multiplanes, which assists freehand VR drawing a) by helping users identify the correct visual depth for drawing their strokes and b) by helping users draw perfect shapes (Figure 1). Our system aims to be used in the early stages of the design process for situations where accurate depictions of shapes and their spatial relationships matter, for example, to show that the shape is a cube or that edges of two shapes are aligned. Following Lim's [19] classification of sketches based on their visual features, such sketches are part of the perceptual level of the design process, with the other levels being functional and conceptual [19]. Multiplanes is positioned at an intermediate stage between "*pure*" sketching and "*pure*" computer-aided design. To achieve this our system automatically identifies drawing surfaces and guides based on previous and current strokes and the current controller pose. Our system also automatically beautifies circles and lines while the user is drawing them. Through the user interface of Multiplanes, designers can focus more on the creative process and worry less – or not at all – about finding the correct stroke position in space. Our contributions are the following:

- **Automatic snapping planes:** we present a new technique that automatically generates potential drawing planes based on the current controller pose and previous strokes.
- **Beautification trigger points:** based on the shape of the current stroke and its geometric relationship with the controller our system automatically creates a new kind of visible guides, called beautification trigger points. Moving the controller into such a trigger point while drawing triggers potential beautification of the current stroke.
- **Real-time beautification:** we introduce a new method for automatic identification and beautification of strokes *while* the user is drawing them.
- **Evaluation of all the above:** we evaluate the usability of Multiplanes in a first user study and compare it with freehand 2D and 3D drawing in a second study.

2 RELATED WORK

2.1 Challenges of 3D drawing in VR

Drawing is an important element of human expression since it is an efficient way to convey or record information and/or to start the design process [31]. Previous work has studied the challenges of 3D drawing and its usability for 3D content creation in VR. One example is Arora et al. [2], who analyzed the factors affecting the human ability to sketch in VR environments. Their goal was to identify the reasons behind the efficiency difference between 2D and 3D drawing. Compared to 2D drawing, drawing accuracy in VR decreased 148%, as measured via the mean overall deviation from a target stroke. They also found that visual guidance, in which users try to follow the guide with the controller; can improve mid-air sketching accuracy, but leads to worse aesthetic quality of strokes, as measured by curve fairness. They identify these problems to be a consequence of the lack of a physical surface and the depth perception issues associated with VR devices.

Wiese, et al. [33] investigated the learnability of freehand sketching in VR. They hypothesized that the difference between 2D and 3D drawing could correspond to higher manual effort and error proneness or to higher cognitive and sensorimotor demands. Yet, they were not able to identify which of the two reasons is the specific cause. However, they found that the users' sensorimotor skills for 3D drawing improve rapidly over time.

Finally, Tramper and Gielen [30] investigated the differences between eye-hand coordination in the frontal plane and in the depth plane while doing tracking and tracing tasks. They found differences in the coordination of gaze and finger position for both planes. For example, for tracking, gaze leads finger position in the frontal plane but lags behind the finger in the depth plane. Based on these results they conclude that the different lead times reflect differences in the dynamics of visuomotor control for coupled eye movements in the same lateral/vertical direction, known as *version*, and independent eye movements in opposite directions for depth accommodation, known as *vergence.*

2.2 3D Drawing

Irrespective of the challenges for 3D drawing, there has been a boom in VR drawing tools. Such systems rely on 3D input with VR controllers to let users draw 3D strokes. Some earlier systems such as 3DM [6], HoloSketch [7] or CavePainting [17], showed the possibilities of directly drawing in 3D, without promoting 2D lines into 3D space. Similar to them, newer commercial systems, for example Quill [10], FreeDrawer [32], and Tilt Brush [12], directly map the stroke position to user movement with a freehand drawing technique. Freehand drawing is easy to learn and everybody can operate these systems regardless of their experience level with VR systems [26,32]. Also, such systems are powerful enough that even an experienced user can be satisfied with the drawn results [27]. However, one problem with freehand drawing systems is that they are less accurate, in comparison to 2D sketches as well as 3D geometric drawings, which can affect the final product or outcome [2].

To solve this problem, Jackson and Keefe noted, "*new tools are needed that provide functionality that goes beyond straight one-to-one mappings of body movements to operations on digital geometry*" [15]. Examples of such systems are those that project the 3D position to a plane to remove a DOF from the stroke creation process. For example, Digital Tape Drawing [13] surrounds the drawing with a cubic volume so users can select an axis-aligned plane and position it in the desired place before drawing there. Users can also create non-planar surfaces onto which they can later draw. Another example is ImmersiveFiberMesh [23], which automatically fits a plane through the stroke. Yet another example is Kim et al. [18], where the users can create curved surfaces with 3D hand gestures. The user can then draw later onto these surfaces with a tablet and pen. Finally, SymbiosisSketch [3] utilizes AR to let users draw in 3D using a tablet. In this system the user positions the tablet manually in 3D space and uses the tablet screen as the current drawing surface. Users can create surfaces and then draw on them. In contrast, Multiplanes uses a mixed approach in which the user controls the general position of the stroke, but the system automatically selects an appropriate drawing surface based on the controller pose and the position of previous strokes. Another difference is that Multiplanes currently only supports drawing on planar surfaces. These surfaces can be positioned and rotated with 6DOF to create 3D drawings from planar strokes.

Other systems use novel metaphors to constrain user actions. For example, Drawing on Air [16] uses the metaphor of "*drawing*" by applying/rolling tape. This system also uses a force feedback device to emulate the resistance of a physical surface while the pen is moving through the air. Finally, Lift-Off [15] lets users select curves from a 2D image previously positioned in space and then rotate these curves with two hands/controllers until they are in the desired pose. As these curves already have their final shape, users only need to focus on positioning them. With Multiplanes we wanted to avoid introducing novel UI metaphors to keep the flexibility of freehand drawing for sketching [19]. Yet, for structured drawings, novel UI metaphors can produce visually better results than systems that support only freehand drawing. When appropriate, Multiplanes automatically beautifies the users' hand movement to create simple geometric shapes without requiring additional user interactions.

2.3 3D Drawing Beautification

Another area of research relevant to 3D drawing is beautification, which is the process of transforming informal and ambiguous freehand input to more formal and structured representations [21]. According to Wiese et al. beautification is useful in 3D drawings since "*creating a volumetric shape require users to make connections between several 2D objects, which is an error-prone action consequence of the depth perception problems that limit the user ability to exactly locate a previously drawn object point in space*" [33]. The main challenge in this area is correctly interpreting the users' intentions while they are making such freehand sketches, because, by nature, such drawings are informal and ambiguous.

An example of a beautification algorithm for 3D drawings is the work of Fiorentino et al. [11], which beautifies a user stroke as the user is creating it in an immersive freehand 3D sketching system. Their method first filters the position data to reduce noise, and then segments the stroke into sections using position, speed and curvature. Finally, they create spline points to create a smooth Bézier curve from the different segments. Rausch et al. proposed an approach [25], where they use a fuzzy logic algorithm for primitive shape recognition and a textual description language to define compound symbols to identify sketches done in a fully immersive freehand 3D sketching system. The first step of their algorithm is to separate each stroke into segments that are then recognized as primitives. Using those primitives, they analyze the whole sketch by matching it to previously learned symbols. This process happens after the user finishes the current stroke, and the system can group strokes together. In contrast to these algorithms, which smooth a curve but do not recognize the type of the shape [11], or recognize the shape and perform beautification only after the stroke is complete [25], Multiplanes' beautification algorithm automatically beautifies strokes in real-time as the user is drawing them.

3 Design Goals

Freehand (2D) sketching has an important role in the early phase of the design process [8,9,19]. However, such 2D design representations can be highly ambiguous and difficult to interpret, as they are composed of different symbols, including annotations, and diagrams, that represent the visual features and spatial relations between objects [19]. For example, a sketch of a room with its furniture needs to consider the orthogonality of the walls and the furniture, and maintain an appropriate proper size relationship between the furniture and the room. Moreover, the designer should be able to annotate the furniture to specify its color and material. Another problem with 2D design sketches is that designers need to understand the 2D representations of 3D objects, which requires strong visual intelligence and spatial skills [5]. Sketching in 3D can help simplify these problems since the drawings are done directly in a 3D environment. An advantage of sketching compared to 3D modeling is the short creation times, as 3D modeling require users to focus on the model geometry in addition to the design process.

However, 3D sketches are often more inaccurate than their 2D counterpart. Wiese, et al. [33] identified two possible causes for the differences in accuracy between 2D and 3D drawing. First, 3D drawing requires higher manual effort, making this activity more prone to errors than 2D drawing. Second, 3D drawing demands a higher cognitive and sensorimotor load. Tremper & Gielen [30] attribute this to the differences between the dynamics of visuomotor control for lateral/vertical (version) eye movements – which roughly speaking corresponds to drawing on a plane directly in front of a user – and vergence eye movements – which occur with visual depth changes.

Based on this research, our main design goal is to reduce the user effort to create accurate strokes, accurate angles, and accurate aligned or snapped vertices while drawing in VR systems by reducing the cognitive load and error proneness of 3D drawing. Our system assists at the perceptual level of the design process by helping designers draw sketches where the spatial relationship between features is accurate. To achieve this, we build on the guidelines proposed by Arora et al. [2] for 3D drawing and Stuerzlinger and Wingrave [28] for 3D user interfaces, and identify the following requirements of our system:

3.1 Show the most appropriate drawing surface: due to the limitations of depth perception in VR, positioning objects floating in free space accurately is hard. However, guiding the user with a drawing surface improves accuracy. Therefore, our system always shows the most appropriate drawing surface based on the current controller pose. Such a surface also highlights geometrical relevant relationships to previous strokes to help the designer to better sketch the spatial relationships between objects.

3.2 Every stroke needs a parent surface: our system projects every stroke onto a surface to remove one degree of freedom from the inaccuracies introduced by the user's arm movement. Parenting all strokes to a surface makes the creation of orthogonal and parallel strokes easier. This method helps to mimic 2D pen and pencil sketching in 3D.

3.3 Incorporate visual guides into the 3D world: our system shows guide surfaces and points as 3D objects inside the composition, with shading and occlusion as depth cues, to make it easier for users to generate accurate results. Also, we use small spheres as guides, which have the smallest possible visual impact on the 3D content. This also avoids tracing actions in which the user follows a guide with their hand, which have been shown to decrease user accuracy.

3.4 Let users participate in the beautification process: most beautification systems infer a user's drawing intention after they finish the stroke. Doing the beautification in real-time during the stroke lets users adapt and correct the result before finishing the stroke by appropriate changes to their hand movement. This permits user to achieve their desired result, even if the initial beautification result is not the desired one, which reduces the need to re-draw a stroke. The current implementation detects circles, arcs, and lines.

3.5 Minimal use of GUI and gestures: our system bases its guides and plane predictions on the current position and rotation of the controller and information about the closest previous stroke, which reduces user's cognitive demands since they do not have to learn/remember gestures or GUI options.

4 Multiplanes

In Multiplanes, users directly draw freehand in the virtual environment. To diminish the freehand drawing problems like the inaccuracies produced by the changes in depth, every stroke is projected onto a surface. To further simplify the drawing, Multiplanes recognizes the type of stroke based upon the geometric relationship between controller position and previous strokes and automatically beautifies the user stroke to the identified type. If the geometric relationships do not suggest any known shape, the system leaves the stroke as a general curve that follows the hand movement, so users are still able to draw arbitrary shapes.

4.1 3D Snapping

Two of our main interactions, the drawing surface generation and the beautification trigger points (BTPs), require identification of the closest vertex of the closest stroke to the current controller position and rotation. To identify this vertex our algorithm proceeds as follows: first we calculate the angular distance, which we compute as a dot product between each stroke's parent plane normal and the direction the controller is pointing.

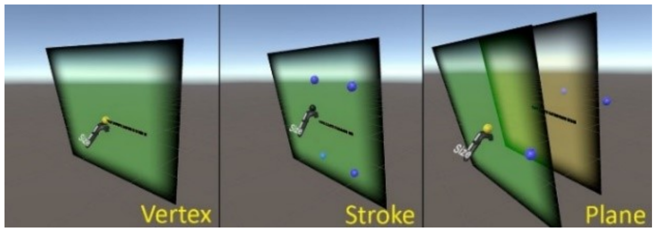

Figure 2: Illustration of the different guide types based on the distance to the closest vertex.

Then we add that value to the distance from the controller to the vertex. This gives us a weighted distance based on the controller rotation. Using this distance, we find the closest vertex and stroke. Once we find the closest stroke, the system visually outlines it to provide feedback to the user as to which stroke is influencing the system.

Our system also uses the distance to the closest vertex to show the correct guides. As explained before this distance is calculated with the current controller position. The potentially available guide types are vertex, stroke and plane, Figure 2. We empirically choose the appropriate distances to heuristically change among types based on human depth perception limitations [22]. For the vertex type, the controller needs to be closer than 5 cm to the stroke; for the stroke type, the controller needs to be closer than 5 cm in visual depth and inside a 50 cm radius of the stroke; and for the plane type, the controller needs to be between 5 cm and 50 cm in visual depth to the stroke. To change between guides and create BTPs users only need to move the controller closer to the stroke/feature they want the guide to follow from. For example, if they want to create a BTP based on the end vertex of a stroke, they would move the controller towards that vertex. Once the controller is less than 5 cm from the end, the BTP automatically appears.

4.2 Drawing Surface Generation

A main feature of our system is the generation of virtual surfaces, called drawing surfaces, onto which strokes are projected. Our approach removes the need for the user to press buttons or perform gestures to define such a surface. At the same time, it still leaves the user in control of the process. In contrast to our approach, manually activated systems suffer from the limits of human depth perception and the need to control 6DOF simultaneously, which make it difficult to accurately position such a plane [2,28]. This problem can further be aggravated in immersive virtual environments by the need to move around to find the best view to position such a plane. On the other hand, fully automatic systems may frustrate users by removing their sense of agency [20,23].

In Multiplanes, we implemented an enhanced 3D version of a previously presented surface creation approach by Igarashi and Hughes [14] called Suggestive Interface, in a desktop interface with a mouse as input device. However, their work only supports lines, while we also allow curves, arcs, and circles. We also use the 3D controller pose inside the virtual environment and previously created surfaces to define the position and rotation of the new surface. The surface generation involves two states. In the first state, the user creates a *feedback surface* while they move the controller in the virtual environment. Such visualizations have been shown to be a good approach to handle 3D drawing in immersive environments [13], where they improve user accuracy. In the second state, the user then creates a *drawing surface*, a static visual guide, upon which the user then can draw new strokes. This approach also helps users understand the assistance provided by the system and lets them make corrections in real time. In contrast, in other systems, such as ImmersiveFiberMesh [23], when the plane is created after the user draws a stroke, users need to do an additional interaction step to change the plane position.

4.2.1 Feedback Surface: in every frame, the 3D snapping algorithm returns the closest stroke parent surface. Together with the controller pose, the system then decides on the most appropriate feedback surface for that frame. Our algorithm first calculates the relationship between the controller rotation and the rotation of the closest stroke parent surface using the dot product of their normal vectors with a threshold. The possible relationships we consider are parallel, perpendicular, acute45° and no relationship. The angular tolerance for determining the controller relationship is 10° for parallel and acute45°, and 15° for perpendicular planes. These values were heuristically chosen through a pilot study, where participants preferred a higher angular tolerance for perpendicular surfaces than for parallel and acute45° surfaces. To avoid confusing users through the automatic change between planes, each plane type has a unique color. The second step uses this relationship to determine the drawing surface position and rotation. We rotate the plane based on the coordinate system of the stroke's parent surface. However, if the controller is inside a beautification trigger point (BTP, see below), we use the local coordinate system of the BTP instead. We calculate this local coordinate system from the stroke information and its parent plane. The first vector of the coordinate system, the normal vector, is the plane normal; the second, the right vector, depends on the type of geometry (the stored direction vector for lines and circles, or the right vector of the plane for general curves). The third vector, the forward vector, is defined by the cross product between those two vectors.

This extra information allows users to better visualize the orientation of a new stroke compared to previous strokes. Table 1 summarizes the relationship between these elements. Finally, our algorithm checks if the feedback surface is similar to a previously saved surface. To do this, we compare the new surface against all the stored surfaces by calculating the dot product of both surface normal vectors. If they are parallel, we then see if the new surface position is on the second surface by calculating the distance between the new surface and the old surface. To consider a surface similar to a previous one, the distance needs to be smaller than 5 cm [22]. When a new surface is in a similar position and orientation to a saved plane, we copy the saved surface values to the new surface.

Table 1: Drawing Surface Rotations

Relationship to closest plane	Used Values for new plane
Parallel	Keep controller position, copy plane rotation.
Perpendicular, Acute45°	Keep controller position, rotate accordingly to relationship.
No relationship	Keep controller position and rotation.

Our feedback surface is a 5 × 5 cm square. We decided to highlight the surface edges as the user needs to perceive where the surface is in space and the orientation of infinite surfaces is harder to perceive without texture [29]. This size also helps the user perceive the surface as part of their hand/controller/drawing device. The color of the feedback surface enables the user to see the values used to create it. Green is for existing surfaces rotations (i.e., parallel planes), yellow for perpendicular surfaces, orange for acute45° surfaces, and blue for entirely new surfaces. Based on our experience in pilots, we expect that users quickly associate each color with a specific relationship.

4.2.1 Drawing Surfaces: once the user starts drawing, the system creates a static surface with the same rotation and position as the displayed feedback surface. The system stores the surface if it is a new one, which allows the system to refer later to it for the creation of feedback surfaces. If the surface is not new, we add the new stroke to the previously generated surface.

4.3 Beautification Trigger points (BTPs)

We created a new type of guide called a beautification trigger point (BTP). These points trigger actions when the user starts or ends a stroke inside them, or when they are crossed during drawing, see Figure 3. This way we can improve user accuracy and at the same time avoid tracing actions, which are detrimental to the aesthetic quality of strokes [2]. Previous work used points to show vertex positions like end points and intersections [4], but BTPs also show geometrical relationships like parallel strokes. In addition, by embedding the trigger points directly inside the 3D environment, we enable the user to benefit from multiple depth cues (the BTPs are rendered as textured spheres, which diminish in visual size with distance), which improves spatial perception.

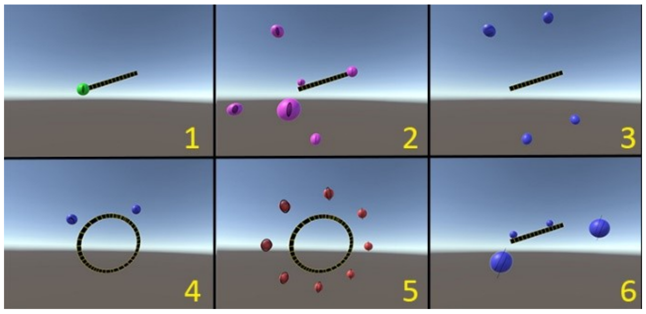

Figure 3: Different BTPs generated depending on stroke shape and the relationship between the plane and controller normal vectors. 1) Vertex; 2) Local coordinate system, created when another BTP is hit; 3) Vertices forming lines parallel to the snapping stroke; 4) Vertices forming tangent line to the circle; 5) 8 circle offset BTPs, to make it easier to perceive the shape of the offset circle; 6) Vertices forming lines perpendicular to the snapping stroke.

In Multiplanes, BTPs have multiple functionalities. First, *when the user is not drawing*, they work as normal snapping points and effectively help to define starting positions on vertices of previous strokes. They also show where to create a stroke relative to previous strokes with a given geometric relationship, such as parallelism, perpendicularity, or offset circles. The distance of the parallel and perpendicular BTPs to the stroke is the same as the stroke length. For the circle offset BTPs the distance to the stroke is the same as the controller distance to the stroke. Second, *while the user is drawing*, BTPs show position(s) where ending the current stroke will snap to a previous drawn stroke. BTPs also help to avoid user mistakes, such as selecting a wrong snapping plane or performing a careless hand movement, since hitting a BTP automatically triggers the 3D stroke beautification module (described below) and updates the drawing surface coordinate system with the BTP coordinate system. Triggering the 3D stroke beautification module can result in the stroke being beautified, based on criteria specific to the module.

Table 2: BTP Types

BTP Type	Guide Type	Stroke Type	Relationship
Vertex	Vertex	Any	Any
Center	Stroke	Circle	Any, less than 5 cm from circle center
Parallel	Stroke, Plane	Line, General Curve	Parallel
Perpendicular	Stroke, Plane	Line, General Curve	Perpendicular
Tangent	Stroke	Circle	Parallel
Circle Offset	Stroke	Circle	Any

4.3.1 BTP creation: in each frame, regardless if the user is drawing or not, the system creates BTPs when the 3D snapping algorithm returns a new closest vertex or changes the guide type. To do this our system uses the closest stroke type, the guide type and the current mathematical relationship to the controller calculated when creating the drawing surface. Table 2 explains how these values interact with each other. To avoid the generation of too many points inside the virtual environment, once we have created a list of potential BTPs, our system uses their position to compare them with existing BTPs. The system does not store such a duplicated BTP, except when the old point is a local coordinate system BTP, in which case the system deletes the old BTP. With this strategy, we avoid having old local coordinate systems affect the drawing plane when they do not match the current stroke. Then we add the new BTPs to the existing ones and calculate the distance and direction for each BTP to the controller. To further limit the number of points, we keep old BTPs generated by a different stroke only if the controller is getting closer and oriented towards that BTP. Finally, to reduce the visual clutter of showing all potential BTPs, we identify the fifteen BTPs that are closest to the controller and display only those.

4.3.2 BTP hit: in our system, users can interact with BTPs by "*colliding*" with them, i.e. intersecting them with the controller while drawing a stroke. This collision can be part of a longer movement, i.e. when users are drawing strokes with multiple sections, which makes it similar to the crossing-based interfaces from Accot and Zhai [1]. Or a BTP can be hit at the end of the stroke, to finish the stroke at that position. Regardless of the user intention, if the controller hits a BTP, we first change its color to yellow and send a vibration to the controller to give the user haptic feedback that they reached the correct point in space. Then, we create special BTPs called Local Coordinate System (LCS) BTPs based on the strokes' local coordinate system. With LCS BTPs, users can make use of a local reference frame, if desired, which helps, for example, with drawing right-angled geometry regardless of orientation of the "*base feature*". LCS BTPs are used to keep an implicit record of the orthogonality between two strokes. They are also useful creating strokes parallel to diagonal strokes.

If the user is drawing when they hit a BTP the drawing surface is oriented automatically to map the coordinate system to the local coordinate system of the BTP. Hitting a BTP triggers the 3D stroke beautification module, which may result in a beautification depending on the stroke. A stroke section is an independent segment of a stroke, created when the user hits a BTP. Once a new section is created, the stroke beautification module no longer uses stroke points from previous sections. This approach is computationally efficient and lets users create complicated shapes with a single multi-section stroke. Finally, if the user exits the BTP during drawing we delete that BTP to avoid creating multiple redundant stroke sections.

4.4 3D Stroke Beautification

Our 3D beautification algorithm identifies the type of stroke based upon the geometric relationship between controller positions (Figure 4) and automatically beautifies the user stroke to the identified type, either circle or line. Automatic identification of the stroke type while the stroke is being drawn gives users automatic feedback about the detected shape and lets them change their hand movement to approve or disapprove that identification. Because we focus only on simple shapes in the current prototype we only beautify lines and circles, with the objective to see how beautification affects the quality of the results.

Our algorithm works as follows: first we identify the shape of the section by using a list of controller positions. To reduce noise, we only save controller positions that are separated by at least 1 cm in space. For this identification, the current section point list needs to have more than four positions – having fewer positions almost always results in a line. Once we have the required number of positions we compute the curvature vector. For this, we utilize the 1D version of the 2D surface Laplace-Beltrami operator [24] by applying the finite element method to diffusions along 1D curves in 3D space. The resulting 1D Laplace-Beltrami operator in 3D space is computed as follows: let the current vertex be p, and the previous and next vertices be $\mathbf{p}_{i-1}$ and $\mathbf{p}_{i+1}$. Let

$$L_{-1} = ||\mathbf{p} - \mathbf{p}_{i-1}||, \text{ and } L_1 = ||\mathbf{p}_{i+1} - \mathbf{p}|| \quad (1)$$

The curvature normal is then:

$$\mathbf{H} = ((\mathbf{p}_{i+1} - \mathbf{p}) / L_1 - (\mathbf{p} - \mathbf{p}_{i-1}) / L_{-1}) / ((L_1 + L_{-1}) / 2) \quad (2)$$

From $\mathbf{H}$, we then compute the curvature k = $||\mathbf{H}||$. We also compute the steering angle, the angle that indicates the rotation between $\mathbf{p}$, $\mathbf{p}_{i+1}$and $\mathbf{p}_{i-1}$, as

$$\cos^{-1}((\mathbf{p}-\mathbf{p}_{i-1}) / L_{-1} \bullet (\mathbf{p}_{i+1}-\mathbf{p}) / L_1) \quad (3)$$

From the curvature and the steering angle of each vertex, we calculate the average curvature, the mean angle, and the number of times that the angle changes direction (the zero-cross rate of the steering angle) for that section. Depending on these values we identify the section as a line, a circle, an arc or a general curve. We determined the threshold values heuristically by having users draw specific shapes multiple times in a pilot study and finding the optimal threshold parameters that works for most cases. The second step is to calculate and store the geometric properties of that section depending on its type. For general curves we store the start, middle and end vertices, for lines we store the start, middle and end vertices, plus line direction and length, and for circles we store the radius and the center position.

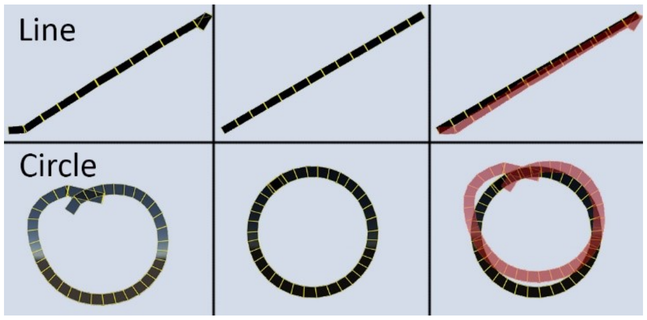

Figure 4: Line and circle beautification. The third column overlays both to facilitate comparisons.

The start, middle and end vertices are taken directly from the vertex list of the section. The line direction and line length are calculated using the start and end vertex positions. To calculate the radius and center we create a circumscribed circle for our points list. This involves creating a triangle for every 3 vertices by calculating its edges and then calculating the normal using the edges. The radius from those 3 vertices is the magnitude of the normal and the center is calculated using the barycentric coordinates equation. We avoid doing a regression, as we know that all our points are already in the same plane. Once we have the radius and center for every 3 vertices, we calculate the mean of those values to get the current section radius and center. The final step is to calculate the beautified stroke positions using the line or circle equation and the calculated values. Again, we use the 2D equation of a circle and line, as they are projected into a plane.

4.5 Other User Interactions

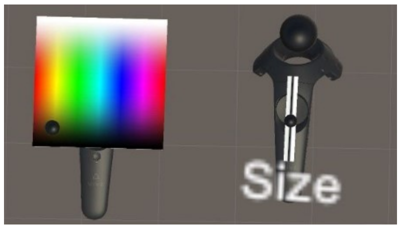

Figure 5: Controller user interface for Multiplanes

Our system supports two additional interactions methods. They permit the user to change the stroke size and color (Figure 5). Stroke size can be adjusted by moving the thumb vertically on the touchpad in the right-hand controller. Stroke color can be adjusted by moving the left thumb on the left-hand controller touchpad. The color space is an HSL plane with saturation on the vertical axis and hue on the horizontal one.

4.6 Implementation

We implemented this system using Unity (version 5.6.1f1) in C# on a 3.60 GHz PC with Windows 10 and an nVidia GTX1080. For the virtual reality headset, we used an HTC Vive with two standard HTC Vive controllers. Figure 6 show an example of a cube created with the system.

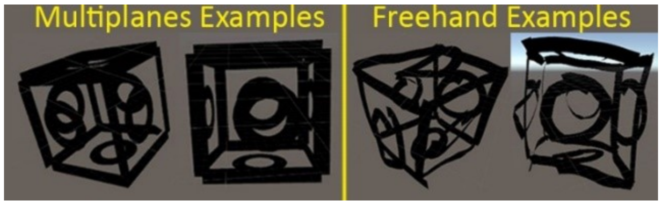

Figure 6: Example of a cube created with the implemented system. (Left) with Multiplanes and (right) freehand drawing.

5 EVALUATION

We evaluated Multiplanes with two user studies on novice users. One qualitative study evaluated the usability and functionality of our prototype. The second quantitative study compared the quality of 2D and 3D sketches done with our prototype to freehand 3D drawing. We choose to compare our system with freehand drawing to let users focus on the underlying strokes without the distractions of all other features available in commercial systems. Also, having a single software for running our study reduced the effect of small differences in interface methods and standardized the logging.

5.1 Usability Study 1

5.1.1 Participants: we recruited eight participants from the undergrad university community. Five were female. All of them were between 18 and 24 years old. Among all participants only 25% had drawing experience in VR.

5.1.2 Methodology: first, participants were introduced to the features of Multiplanes. Then participants used the system in a practice phase until they felt comfortable with drawing in 3D. Subsequently, they were asked to do a set of simple exercises to test their knowledge of the features. These exercise tasks were: create a line, create a circle, change color and size of a stroke, draw a cube, and draw a smiley face. Once participants could complete these tasks, they were instructed to perform the main task: to draw a 3D chair. Participants were shown a picture of a chair on a piece of paper and the experimenter explained its elements to the participants (Figure 7a). They were then instructed to draw the chair as accurately as possible, to make sure that all strokes touch each other and to avoid adding any extra elements to the sketch. Then participants drew the same chair 5 times: one time using only drawing surfaces, one time using only BTPs, one time using only beautification, one time using the full system and one time freehand. When only beautification was enabled, all strokes triggered the beautification module. While drawing, participants were asked to use the "*think-aloud*" method to explain their actions. Between each drawing interaction method, participants filled a questionnaire and were interviewed briefly about that interaction. We used the same order for all conditions for the first 3 drawings (drawing surface, BTPs, and beautification) to enable users to practice each functionality before trying the full system. For the last 2 drawings (full system vs freehand) the order of conditions was counter-balanced across participants. For the full system and freehand drawings participants only did the interview. We recorded a video of the participant's display and their voice while they were drawing, as well as their voice during the interview sections.

5.1.3 Design: participants answered a multipart questionnaire about ease of interaction, perceived speed, perceived accuracy and overall opinion for each interaction method. In the interview participants answered questions about their experience while drawing the chair and their opinion about the system and the interactions.

5.1.4 Results, Observations & Discussion: all participants could complete the tasks, see Figure 7b for several examples. Results from the questionnaire were lower than expected (average of 4.5 on the 7-point Likert scale), see Table 3 for individual scores. However, it was particularly interesting that, although the scores rated Multiplanes as "*medium*", in the interview six participants talked positively and were excited about the system and its features. For example, participant 4 said "*I liked when you combined all [methods] together, because when I was drawing the chair, I could match the points. I took this plane here and this beautification points here. Make sure that it matches up, then I draw them together.*" On a similar note, participant 5 responded to the same question in a similar way, "*I loved the full system, like it helps with [drawing] flat [content]. And I can use it to see if [things are] perpendicular or parallel*". Finally, participant 1 said that "*it gives a good idea where you are drawing, because it helps you see how the planes are correlated in 3D*".

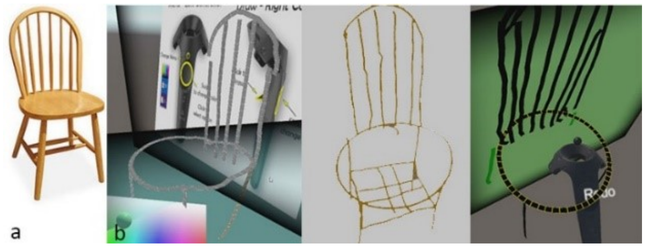

Figure 7: a) The picture of a chair shown to participants b) Three examples of chairs created with the implemented system.

The two participants that did not like the full system both experienced problems activating the features, especially the BTPs and the beautification. For example, participant 2 said that *"the system doesn't detect the beautification"*. This, together with the "*think-aloud*" transcripts, leads us to believe that the scores correspond more to problems with the prototype than real problems with the proposed interactions. Figure 7b illustrates some of these problems, with some strokes not being beautified and some not snapping to previous strokes.

When asked about the BTPs, most participants liked their functionality. For example, participant 4 said that "*when I was looking at the other side, it kind of helped me to double check to see if everything was good. And it was easy because it kind of [popped up] and show[ed] me this [BTP] point here to draw the line here*". However, they complained that sometimes the BTPs got in their way, especially the circle offset BTPs. For example, participant 3 said that *"[I] wanted to draw one line freely, but the points were in the way. This was frustrating*". When asked about the plane creation and the automatic snapping algorithm, participants used the change in color to identify the different relationships, for example participant 2 said that "*works fine, I understand the [plane] colors. Look for it [the color] and get it fast*". Some participants found the automatic snapping algorithm too sensitive but were still able to use it. For example, participant 4 said that *"[the plane was] a little bit jerky at times when I was trying to switch to a different plane. But when I was trying to draw to a new plane, it worked really well*". Finally, when asked about the beautification module, participants also had positive words about it. For example, participant 6 said that *"[beautification] will create the lines perfectly straight and circles into proper circles instead of distorting them with different starting points. So that was a good one*". One problem that participants identified was that the algorithm was not sensitive enough and sometimes they felt that it was not beautifying their stroke as expected, for example, participant 1 said that "*I think [it] was trying to, but it keep creating a line automatically*".

We also analyzed the participant's opinion about drawing freehand. Most participants complained about having problems when they tried to connect strokes and when trying to draw in the same plane without assistance. These results agree with previous work [2]. More important, four participants said they preferred Multiplanes as it helped them solve these problems. For example, participant 6 stated that *"I will add the plane because you can draw in a particular dimension or axis, rather than beginning in a certain axis and ending in another"*. And participant 8 stated that *"drawing freehand is hard because it has no plane and no points"*. Therefore, based on the interview answers we believe that Multiplanes fulfills its design goal to simplify 3D drawing by making it easier to draw shapes accurately, particularly by removing the need to correctly judge the depth of a stroke while drawing.

Table 3: Study 1 Likert Scores

Interaction	Ease	Speed	Accuracy	Overall
Drawing Surface	4.9	4.5	4.6	5.5
BTPs	4.8	4.6	3.9	4.6
Beautification	4.9	4.6	4.3	5

5.2 Study 2: Comparison with freehand drawing

In this study we aimed to compare the quality of Multiplane drawings with freehand 3D sketches. We used a similar version of Multiplanes, with fixes for some of the usability problems that might have affected the results of the first study.

5.2.1 Participants: we recruited six paid participants from the university community. 50% were female. 50% of the participants were between 18 and 24 years old, 34% were between 25-34 years old, and 16% were between 35 and 44 years old. Among all participants only 33% had drawing experience in VR.

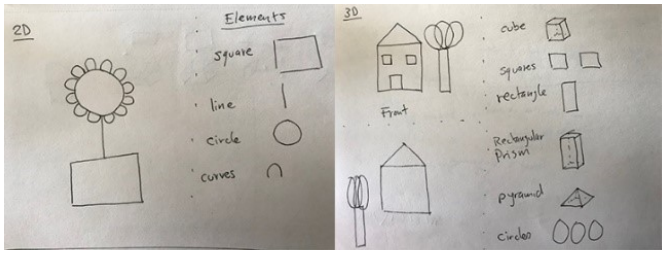

Figure 8: a) The sketch of a flower in a pot shown to participants. B) The sketch of a 3D house with a tree shown to participants

5.2.2 Methodology: participants experienced the same introduction, training and explanation phase as in study 1. Figure 8 shows the sketches shown to participants to explain the task. However, in this experiment participants performed two drawing tasks, each with Freehand and Multiplanes, for a total of four drawings. The first drawing was a 2D sketch of a flower in a pot and the second one was a 3D house with a tree (Figures 9 and 10). For the 2D sketching tasks, participants were not permitted to move away from their starting position. For the 3D sketch tasks participants were encouraged to move around the sketch. We evaluated both 2D and 3D sketches, because we wanted to see if the accuracy problems in VR drawing are caused only by depth perception errors or also by the user spatial perception of the 3D environment and their navigation within it. Such issues may become more pronounced when the user moves around in the virtual environment than when they stay in the same place. Between drawing tasks, participants were permitted to rest for up to five minutes. Once participants finished the 3D task, they were asked to answer a questionnaire. Screen recordings of the participants were done to later evaluate the quality of the drawings.

5.2.3 Design: the study used a 2x2 within-subjects design. The independent variables were interaction technique (freehand and multiplanes), and sketch type (2D and 3D). The order of conditions across both dimensions was counter-balanced across participants. We coded their final drawings using the method from Wiese et al. [33] to evaluate the quality of the drawings based on the strokes. This coding method evaluates the quality of a drawing using four categories: a) line straightness, b) matching of line pairs, c) degree of deviation, and c) corrective movements. Drawings can get up to 3 points in each category and the total score is the sum of these values (maximum 12 points). Finally, participants answered a usability questionnaire similar to study 1.

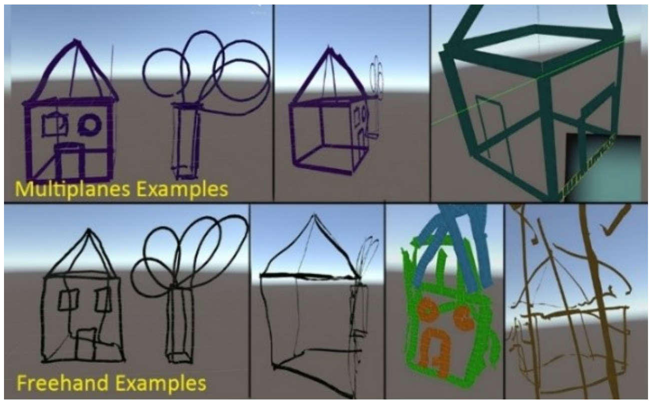

Figure 9: Examples of 3D drawings created by participants (top Multiplanes, bottom Freehand).

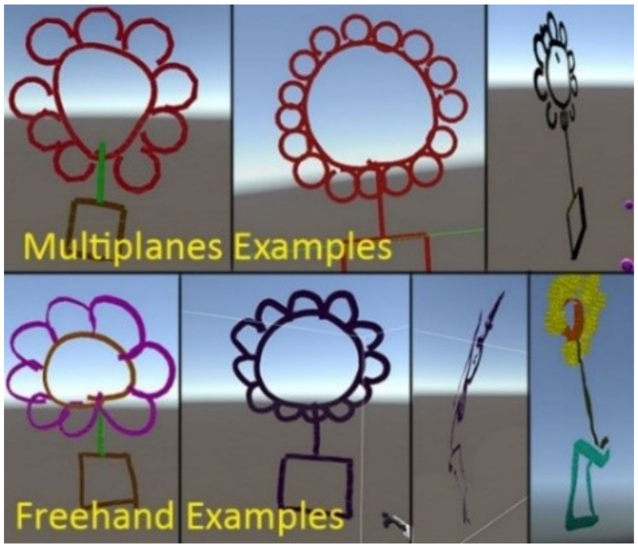

Figure 10: Examples of 2D drawings created by participants (top Multiplanes, bottom Freehand).

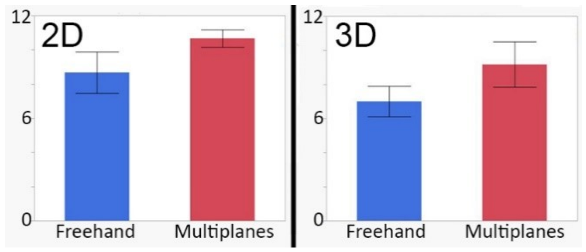

Figure 11: Drawing Scores for each interaction technique divided by sketch type.

5.2.4 Results & Discussion: all participants could complete the tasks, see Figure 9 & 10 for several examples. We analyzed the total scores for each drawing using repeated measures ANOVA with α = 0.05. The data was normally distributed. Table 4 shows the average scores for each condition. Statistical results are reported below:

Interaction Technique ($F_{1, 5} = 107.8$, $p < 0.001$): overall, there was a significant main effect of score on interaction technique, see Table 4 and Figure 11. Average score for drawings made with Multiplanes (m = 9.9, SD = 1.24) were significantly higher than for freehand drawings (m = 7.8, SD = 1.34). Cohen's d = 1.6 identifies a large effect size.

Sketch Type ($F_{1, 5} = 12.11$, $p < 0.05$): overall, there was a significant main effect of score on sketch type, see Table 4 and Figure 11. The average score for 2D drawings (m = 9.7, SD = 1.37) was significantly higher than for 3D drawings (m = 8.1, SD = 1.56). The effect size was again large (d = 1.1).

Interaction Technique × Sketch Type ($F_{1, 5} = 0.05$, $p = 0.83$): there was no significant main effect of interaction technique on sketch type.

Table 4: Study 2 Drawing Accuracy Scores, higher scores are bolded

Drawing	2D Freehand	2D Multiplanes	3D Freehand	3D Multiplanes
Line Straightness	2.3	**3.0**	1.7	**2.3**
Match Line	2.5	2.5	1.8	**2.3**
Degree Deviation	1.7	**2.7**	1.5	**2.0**
Corrective Movement	2.2	**2.5**	2.0	**2.5**
Total	8.7	**10.7**	7	**9.2**

Although our study had a small number of participants, the effect sizes were large, and the mean difference between groups is larger than one standard deviation. Together with the visually more appealing drawings created with Multiplanes, this motivates us to believe that Multiplanes is better than freehand drawing for VR sketching.

When analyzing the questionnaire results, most participants liked the freedom of freehand drawing. Yet, the results also show that they appreciated the benefit of higher accuracy achievable with Multiplanes. Moreover, they rated the ease of interaction, automatic and unobtrusive beautification, and the BTP functionality high (average of 5.5 on the 7-point Likert scale or better). The results of study 2 together with those from study 1, confirm that Multiplanes help increase accuracy when drawing in VR by providing interactions that help reduce depth perception and visuomotor errors.

An interesting finding was that for both systems 2D drawings had higher scores than 3D drawings. This finding is also reflected in the questionnaire results, where the ratings for the 2D sketches were in general higher. These results lead us to hypothesize that the higher cognitive load when drawing in VR is not only a consequence of depth perception issues and higher visuomotor skill requirements, but also reflects challenges with the user's spatial perception of the environment.

6 CONCLUSION

We presented Multiplanes, a VR freehand drawing assistant that incorporates novel interaction techniques, which help users be more accurate. Our work aims to help users, especially non-artists, in creating sketches of concepts or ideas that rely on geometrical relationships between strokes, such as parallel features. For the current controller pose and stroke, Multiplanes automatically identifies an appropriate drawing surface. The system then also displays guides beautification trigger point guides, called BTPs, based on previous strokes. These guides show geometrical relationships to previous strokes and snapping points. Multiplanes also automatically beautifies a stroke in real-time while the user is drawing it or when users hit a BTP. Our two studies identified that participants liked the system and appreciated the increased accuracy they could achieve with it. Not only that, but an analysis of the stroke quality show that Multiplanes drawings had a better quality than freehand drawings. We believe that this difference is a consequence of our design since Multiplanes addresses some of the depth perception and visuomotor errors present in VR systems, which cause problems, for example, when joining two lines or correctly identifying the drawing surface of a stroke. In the future we plan to extend the functionality of Multiplanes.

6.1 Limitations

In our current system we only implemented planar surfaces as we wanted to focus on evaluating our automatically generated surfaces in a reasonably simple context. This approach limits the variety of sketches that can be created in our system to planar shapes. We believe the design of the Multiplanes interaction techniques can be generalized to non-planar surfaces, perhaps based on arcs or freeform curves, which will let users draw shapes such as animals and humanoids. We plan to include such types of surfaces in a future version of our system. Another limitation is that the beautification system only beautifies to arcs, circles and lines. In the future, we plan to further improve our system to use a larger variety of shapes, including spline curves.

ACKNOWLEDGMENTS

This research began during an internship at Adobe Research, and was sponsored in part by Adobe Research, CONACYT and NSERC.

REFERENCES

1. Johnny Accot and Shumin Zhai. 2002. More than dotting the i's --- foundations for crossing-based interfaces. *Proceedings of the SIGCHI conference on Human factors in computing systems Changing our world, changing ourselves - CHI '02*, 4: 73. https://doi.org/10.1145/503376.503390

2. Rahul Arora, Rubaiat Habib Kazi, Fraser Anderson, Tovi Grossman, Karan Singh, and George Fitzmaurice. 2017. Experimental Evaluation of Sketching on Surfaces in VR. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '17)*, 5643–5654. https://doi.org/10.1145/3025453.3025474

3. Rahul Arora, Rubaiat Habib Kazi, Tovi Grossman, George Fitzmaurice, and Karan Singh. 2018. SymbiosisSketch: Combining 2D & 3D Sketching for Designing Detailed 3D Objects in Situ. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '18)*: 1–15. https://doi.org/10.1145/3173574.3173759

4. Eric A. Bier. 1990. Snap-dragging in three dimensions. *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '90)* 24, 2: 193–204. https://doi.org/10.1145/91394.91446

5. Ernesto Bueno and Benamy Turkienicz. 2014. Supporting Tools for Early Stages of Architectural Design. *International Journal of Architectural Computing* 12, 4: 495–512. https://doi.org/10.1260/1478-0771.12.4.495

6. Jeff Butterworth, Andrew Davidson, Stephen Hench, and Marc. T. Olano. 1992. 3DM: A Three Dimensional Modeler Using a Head-Mounted Display. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D'92)*, 135–138. https://doi.org/10.1145/147156.147182

7. Michael F. Deering. 1996. The HoloSketch VR sketching system. *Communications of the ACM* 39, 5: 54–61. https://doi.org/10.1145/229459.229466

8. Catherine Elsen, Jean Noël Demaret, Maria C. Yang, and Pierre Leclercq. 2012. Sketch-based interfaces for modeling and users' needs: Redefining connections. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM* 26, 3: 281–301. https://doi.org/10.1017/S0890060412000157

9. Daniela Faas, Qifang Bao, and Maria C. Yang. 2014. Preliminary Sketching and Prototyping: Comparisons in Exploratory Design-and-Build Activities. *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* 7, January 2016: V007T07A018. https://doi.org/10.1115/DETC2014-34928

10. Facebook. 2018. Quill. Retrieved from https://www.facebook.com/QuillApp/

11. Michele Fiorentino, Giuseppe Monno, Pietro A. Renzulli, and Antonio E. Uva. 2003. 3D Sketch Stroke Segmentation and Fitting in Virtual Reality. In *International Conference on the Computer Graphics and Vision*, 188–191.

12. Google. 2016. Tilt Brush. Retrieved from https://www.tiltbrush.com/

13. Tovi Grossman, Ravin Balakrishnan, Gordon Kurtenbach, George Fitzmaurice, Azam Khan, and Bill Buxton. 2002. Creating principal 3D curves with digital tape drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '02)*, 121–128. https://doi.org/10.1145/503376.503398

14. Takeo Igarashi and John F. Hughes. 2001. A suggestive interface for 3D drawing. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '01)*, 173–181. https://doi.org/10.1145/502348.502379

15. Bret Jackson and Daniel F. Keefe. 2016. Lift-Off: Using Reference Imagery and Freehand Sketching to Create 3D Models in VR. *IEEE Transactions on Visualization and Computer Graphics* 22, 4: 1442–1451. https://doi.org/10.1109/TVCG.2016.2518099

16. Daniel F. Keefe, Robert C. Zeleznik, and David H. Laidlaw. 2007. Drawing on Air: Input Techniques for Controlled 3D Line Illustration. *IEEE Transactions on Visualization and Computer Graphics* 13, 5: 1067–1081. https://doi.org/10.1109/TVCG.2007.1060

17. Daniel F Keefe, Daniel Acevedo, Tomer Moscovich, David H Laidlaw, and Joseph LaViola. 2001. CavePainting: A Fully Immersive 3D Artistic Medium and Interactive Experience. *Proceedings of the ACM Symposium on Interactive 3D Graphics (I3D 2001)*: 85–93. https://doi.org/http://doi.acm.org/10.1145/364338.364370

18. Yongkwan Kim, Sang-Gyun An, Joon Hyub Lee, and Seok-Hyung Bae. 2018. Agile 3D Sketching with Air Scaffolding. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '18)*: 1–12. https://doi.org/10.1145/3173574.3173812

19. Chor-kheng Lim. 2003. An insight into the freedom of using a pen: Pen-based system and pen-and-paper. *Proceedings of Asian Design International Conference.* Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.759

20. Hannah Limerick, David Coyle, and James W. Moore. 2014. The experience of agency in human-computer interactions: a review. *Frontiers in Human Neuroscience* 8, August: 1–10. https://doi.org/10.3389/fnhum.2014.00643

21. S. Murugappan, S. Sellamani, and K. Ramani. 2009. Towards beautification of freehand sketches using suggestions. In *Proceedings of the Eurographics Symposium on Sketch-Based Interfaces and Modeling (Expressive'09)*, 69. https://doi.org/10.1145/1572741.1572754

22. Robert Patterson and Wayne L. Martin. 1992. Human stereopsis. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 34, 6: 669–92. https://doi.org/10.1177/001872089203400603

23. Helen Perkunder, Johann Habakuk Israel, and Marc Alexa. 2010. Shape Modeling with Sketched Feature Lines in Immersive 3D Environments. In *Proceedings of the Eurographics Symposium on Sketch-Based Interfaces and Modeling (Expressive'10)*, 127–134.

24. Ulrich Pinkall and Konrad Polthier. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics* 2, 1: 15–36.

25. Dominik Rausch, I Assenmacher, and Torsten W. Kuhlen. 2010. *3D Sketch Recognition for Interaction in Virtual Environments.* The Eurographics Association. https://doi.org/10.2312/PE/vriphys/vriphys10/115-124

26. Steven Schkolne, Michael Pruett, and Peter Schröder. 2001. Surface drawing: Creating Organic 3D Shapes with the Hand and Tangible Tools. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '01)*, 261–268. https://doi.org/10.1145/365024.365114

27. Ryan Schmidt, Azam Khan, Gord Kurtenbach, and Karan Singh. 2009. On expert performance in 3D curve-drawing tasks. In *Proceedings of the Eurographics Symposium on Sketch-Based Interfaces and Modeling (Expressive'09)*, 133–140. https://doi.org/10.1145/1572741.1572765

28. Wolfgang Stuerzlinger and Chadwick A. Wingrave. 2008. The Value of Constraints for 3D User Interfaces. In *Virtual Realities: Dagstuhl Seminar*, Sabine Coquillart, Guido Brunnett and Greg Welch (eds.). Springer Vienna, Dagstuhl, 203–223. https://doi.org/10.1007/978-3-211-99178-7_11

29. Geb Thomas, Joseph H Goldberg, David J Cannon, and Steven L Hillis. 2002. Surface textures improve the robustness of stereoscopic depth cues. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 44, 1: 157–170. https://doi.org/10.1518/0018720024494766

30. Julian J. Tramper and Stan Gielen. 2011. Visuomotor coordination is different for different directions in three-dimensional space. *The Journal of Neuroscience* 31, 21: 7857–7866. https://doi.org/10.1523/JNEUROSCI.0486-11.2011

31. David. G. Ullman, Stephen Wood, and David Craig. 1990. The importance of drawing in mechanical design process. *Computer & Graphics* 14, 2: 263–274.

32. Gerold Wesche and Hans-Peter Seidel. 2001. FreeDrawer: A Free-Form Sketching System on the ResponsiveWorkbench. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST '01)*, 167. https://doi.org/10.1145/505008.505041

33. Eva Wiese, Johann Habakuk Israel, A. Meyer, and S. Bongartz. 2010. Investigating the learnability of immersive free-hand sketching. *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium (SBIM'10)*, October 2017: 135–142.