Projective Windows: Bringing Windows in Space to the Fingertip

Joon Hyub Lee, Sang-Gyun An, Yongkwan Kim, Seok-Hyung Bae
Department of Industrial Design, KAIST
joonhyub.lee | sang-gyun.an | yongkwan.kim | seokhyung.bae @ kaist.ac.kr

Figure 1. In a mock setup depicting the flow of Projective Windows, (a) the user wishing to adjust the position and scale of an AR window (b) grabs the window, (c) moves it, (d) makes it bigger by bringing it closer, and (e) projects it to the desired position.

ABSTRACT
In augmented and virtual reality (AR and VR), there may be many 3D planar windows with 2D texts, images, and videos on them. However, managing the position, orientation, and scale of such a window in an immersive 3D workspace can be difficult. Projective Windows strategically uses the absolute and apparent sizes of the window at various stages of the interaction to enable the grabbing, moving, scaling, and releasing of the window in one continuous hand gesture. With it, the user can quickly and intuitively manage and interact with windows in space without any controller hardware or dedicated widget. Through an evaluation, we demonstrate that our technique is performant and preferable, and that projective geometry plays an important role in the design of spatial user interfaces.

Author Keywords
Augmented reality; virtual reality; 3D window management

ACM Classification Keywords
H.5.2. Information interfaces and presentation (e.g., HCI): User Interfaces-Interaction styles, Windowing systems

INTRODUCTION
We imagine an immersive future of computing where minimal gear worn over the eyes brings virtual interactive elements to the space around the user. These virtual elements may merely enhance the user's experience with the reality (AR) or may completely overwrite it (VR).

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

ACM ISBN 978-1-4503-5620-6/18/04…$15.00
https://doi.org/10.1145/3173574.3173792

Although an entirely new user interface (UI) suited to spatial computing may emerge as did the graphical UI consisting of windows, icons, menus, and a pointer for desktop PCs in the 1980s [21], and the touch UI for smartphones in the 2000s [24], we expect some spillovers from these past paradigms.

In particular, we focus on windows in this study. Much of the content that humans produce and consume, such as texts, images, and videos, is and will remain 2D. As such, windows, simply defined as 3D panels encapsulating 2D content and controls, will likely remain essential building blocks of the new spatial UI.

However, compared with their 2D counterparts in a 2D workspace, manipulating the positions, orientations, and scales of 3D windows in space can be difficult due to the added dimension. This necessitates techniques for easily managing and interacting with the windows scattered across the user's immersive 3D workspace.

We propose one such technique called Projective Windows. Through the strategic use of the absolute and apparent sizes of a window at various stages of the interaction, our technique enables grabbing, moving, scaling, and releasing the window in one continuous hand gesture (Figure 1). With our technique, the user can quickly and intuitively manage and interact with windows in space without any controller hardware or dedicated UI widget.

In the following sections are a discussion of previous works on spatial window management, a proposal of our Projective Windows technique, a description of our experiment setup, an evaluation of the performance and preferences of our technique, our conclusion, and a discussion on future work.

RELATED WORK
In this section, we review existing studies related to spatial window management, focusing on interacting with windows bound to various spatial reference frames.

Windows in Space

First formalized at Xerox PARC, a window signifies the expansion of an icon, representing an open file or a program [21]. Since its inception, the window has been an integral part of the 2D GUI. Subsequently, Feiner et al. introduced 3D windows in a VR environment bound to three types of reference frames: the head, the surround, and the world [6]. Ens et al. surveyed various spatial window techniques [4].

Head-Bound Windows

Head-bound windows are anchored to the user's view frustum, always visible regardless of the viewing direction, and useful for displaying ambient information. Techniques involving touching the front side of the head-mounted display [8] can enable interaction with these windows.

Surround-Bound Windows

Surround-bound windows are anchored to the virtual scaffold centered on the user's body, usually within arm's reach. This scaffold serves as a mobile workspace where the user can use kinesthetic memory in interacting with the bounded windows using the hand [5, 14]. Although these windows are virtual, suitable visualizations [3] can assist in touch interactions with them.

World-Bound Windows

World-bound windows are anchored to specific positions within the physical or virtual environment that serves as a stationary workspace [20], or to movable objects within the environment, such as the user's palm or physical props in the user's vicinity [9], that serve as physical proxies. Although world-bound windows can facilitate contextual interactivity [10], working with windows bound to distant positions or objects calls for novel remedies.

Interacting with Distant Windows

One such remedy is the image plane interaction proposed by Pierce et al. that enables the user to directly interact with distant objects in a 3D space using the hand, just as the user can on a 2D screen using a mouse cursor [18]. However, in a stereoscopic setup, when the user tries to select a distant window with a fingertip, either the window or the fingertip can appear duplicated depending on the user's vergence, due to binocular parallax. Nevertheless, once the window is selected utilizing existing techniques that can mitigate the parallax [12, 13, 23] and brought to the fingertip with the apparent size unchanged [18], the user may further interact with it without binocular parallax.

Positioning and Scaling Windows

In a 2D workspace, windows are frequently moved and resized for referencing and multitasking. But specifying the desired 3D positions, orientations, and scales of windows in space remotely is a non-trivial task. Typically, ray-casting [11, 16] and multi-DOF widgets [22] are used for such a task, but Bukowski et al. showed that 3D objects can be positioned and oriented easily using the scene geometry as a reference, with only a mouse cursor and without any such widgets [2].

PROJECTIVE WINDOWS

In this section, we explain how fluid handovers between the absolute and apparent sizes of a window enable the grabbing, moving, scaling, and releasing of the window in one continuous hand gesture. Note: The following descriptions and figures assume the perspective of the dominant eye of the user, for better communication of the core concept.

Making an Area Cursor

First, the user makes an open pinch gesture to create a circular area cursor [7] that activates all of the windows crossing boundaries with it (Figure 2a). The user further specifies the selection by closing (Figure 2b) the fingers, and completes the selection by making the tips of the fingers touch (Figure 2c).

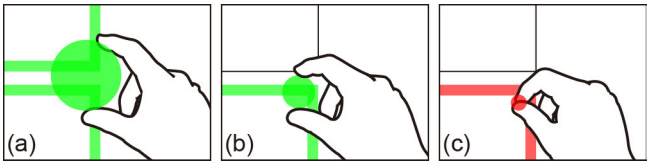

Figure 2. (a) The user creates a big area cursor, (b) specifies a window in a cluttered situation by closing the fingers and making the cursor smaller, and (c) selects it by pinching.

Grabbing Window

When the user makes a selection on a window (Figure 3a), i.e., a grab, the window is instantly brought to the hand (Figure 3b) while maintaining the same *apparent* size, thus *appearing* the same to the user (Figure 3a, b insets) as in the image plane interaction proposed by Pierce et al. [18]. This is done by reverse-projecting the window to a picture plane defined at the hand. Here, a window is akin to a flying touchscreen that snaps to the fingertip on demand, with which the user can then touch and interact [3] without binocular parallax [12].

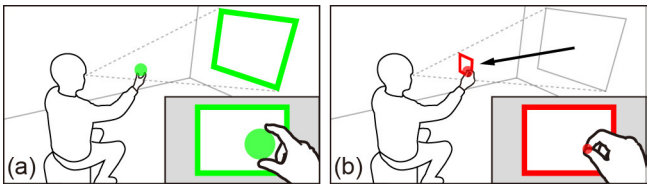

Figure 3. (a) The user makes a grab gesture on a window to (b) projectively bring it to the grabbed point.

Positioning and Scaling Window

Once grabbed, the window's *absolute* size remains the same, so the user can make the window *appear* bigger by bringing it closer to the face (Figure 4a), or smaller by taking it away from the face (Figure 4b). This apparent image essentially serves as a feedforward for the later projection. When the user releases the grab, the window is projected to the surface while maintaining the same *apparent* size.

At the same time, the user can move the grabbing hand to choose the surface onto which to project the window. The window is projected onto a vertical surface (Figure 4a, b), whereas the window is erected against a horizontal surface, perpendicular to the user's gaze, to enforce the best viewing angle (Figure 4c).

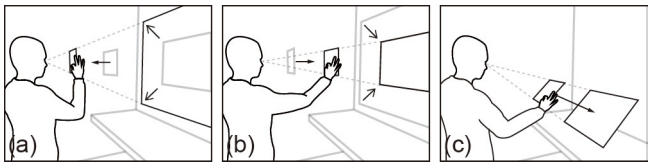

Figure 4. (a) The user makes the window appear bigger by bringing it closer to the face, and (b) smaller by putting it away. The user can project a window (a, b) onto a vertical surface, or (c) make it stand on a horizontal surface.

The behavior differs for vertical and horizontal surfaces so as to follow physical metaphors proposed by Bukowski et al. [2], as people tend to hang picture frames onto walls and erect them on desks. However, surfaces may enforce various projection policies as needed; it may make no sense to erect a drawing window against a horizontal digital tablet that accepts pen input.

Zoom Factor

We estimate how much scaling a single grab-move-release can produce. In a simplified case, the user directly facing a wall grabs a window of width W_1 attached to a wall at D_1 with the hand at d_1 (thereby reducing it to a fixed width w at the hand), moves it to d_2, and releases it to another wall at D_2 (Figure 5a). The zoom factor, defined as the final width W_2 divided by the initial width W_1, by similar triangles, can be expressed as:

$$\text{zoom} = \frac{W_2}{W_1} = \frac{wD_2/d_2}{wD_1/d_1} = \frac{D_2 d_1}{D_1 d_2} \qquad (1)$$

Typical values for D are 1 m for a wall just out of reach, and 4 m for a distant wall in a conference room. For d, they are 0.1 m for the closest comfortable distance from the face, and 0.4 m for the maximum comfortable arm extension. We substitute these values in the 9 possible cases in equation 1 (Table 1).

The width of a window can be increased and decreased by a factor of 16 through a single grab-move-release sequence, demonstrating the advantage of projective geometry.

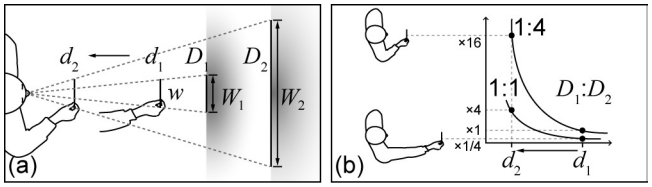

Figure 5. (a) When the user grabs a window from a wall, moves it relative to the face, and releases it to another wall, (b) the zoom (ratio of the final to initial widths) is inversely proportional to *d*.

zoom	$D_2 < D_1$	$D_2 = D_1$	$D_2 > D_1$
$d_2 < d_1$	1	4	16
$d_2 = d_1$	1/4	1	4
$d_2 > d_1$	1/16	1/4	1

Table 1. Zoom when attaching a window to a closer ($D_2 < D_1$), the same ($D_2 = D_1$), or a farther ($D_2 > D_1$) wall by bringing it closer to ($d_2 < d_1$), keeping the same distance ($d_2 = d_1$), or taking it farther from ($d_2 > d_1$) the face.

The same zoom would be more tedious in spatial interfaces that deal with absolute sizes. Moreover, because the zoom is inversely proportional to d (Figure 5b), when the zoom is large, the slope is steep, and when the zoom is small, the slope is gentle, so the user has finer control when the zoom is smaller. Some spatial techniques separately implement such a dynamic control resolution [19], but we obtain this favorable property as a byproduct of projective geometry.

IMPLEMENTATION

We used an Oculus VR headset for head tracking, a Leap Motion hand-tracking sensor attached to the front of the headset (Figure 6a) for tracking hand movement and posture within the user's view frustum, the Leap Motion sensor's front-facing camera for a video feed of the real world, the Unity 3D engine for integrating virtual and physical elements (Figure 6b), and an Intel quad-core i7 3.60 GHz PC with an Nvidia GTX 980 GPU. All calculations used the dominant eye position, so the visual continuity was maintained for the dominant eye but not for the non-dominant eye.

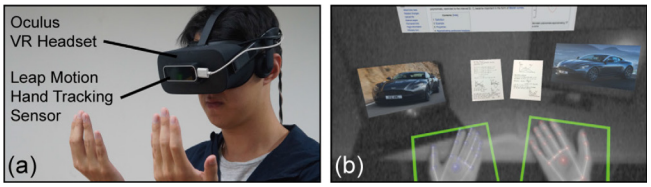

Figure 6. (a) Implementation hardware. (b) The hands, real and virtual objects in the user's view.

EVALUATION

The evaluation was carried out in two consecutive sessions: experiment and exploration; the same volunteers participated in both, in one sitting. The experiment compared Projective Windows using a hardware controller (PWC) (Figure 7) to a widget-based ray-casting technique (RC) using the same controller. The exploration probed the feasibility of using the hand for Projective Windows (PWH).

This decision to have participants perform PWH last was deliberate, as during a pilot test, those who tried PWH first developed behaviors that confounded with subsequent trials with PWC: The hand-tracking sensor's inability to recognize the pinch gesture of a hand moving at a high speed led to unwanted grabs and releases, and as a result, pilot testers avoided moving the hand too quickly and performed many broken-up operations within a limited region that they thought was the sensor's sweet spot. Those who tried PWC first did not develop such behaviors, precluding the possibility of counterbalancing.

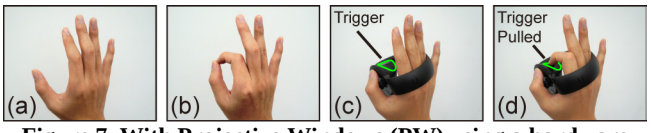

Figure 7. With Projective Windows (PW) using a hardware controller (PWC), (a-b) a finger pinch is substituted with (c-d) the pull of the trigger (highlighted green) on the hardware controller. Note the similarity between the bare hand gesture and the controller-based gesture.

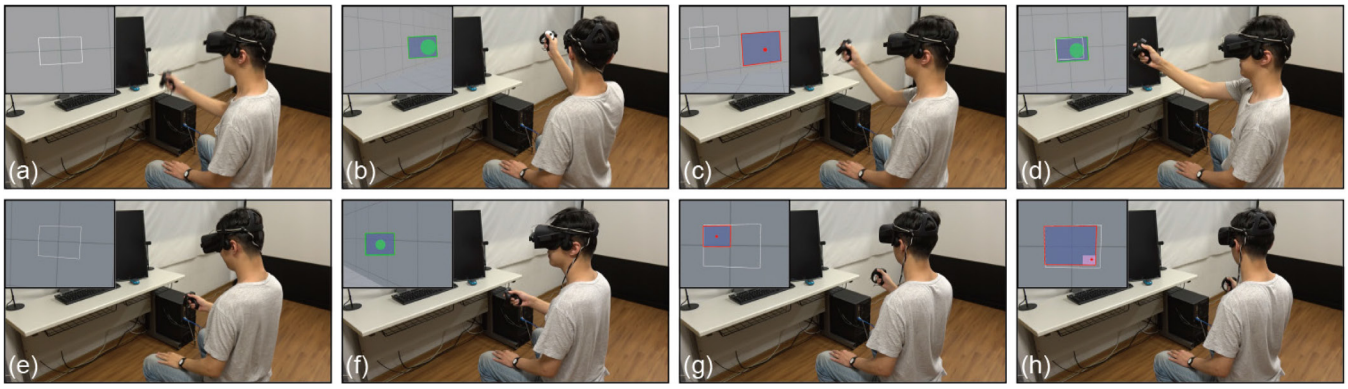

Figure 8. The experiment was conducted with PW using a controller (PWC) and ray-casting (RC). Using PWC, the participant (a) first searched for the target (white), (b) grabbed the window (blue), (c) moved the window while adjusting its apparent size, and (d) released it onto the target when the apparent sizes matched. Using RC, the participant (e) first searched for the target (white), (f) dragged the window (blue) to the target, (g) placed it in the target, and (h) scaled it so that the absolute sizes matched.

EXPERIMENT SESSION

For the comparison between PWC and RC, to be fair, the sizes of widgets in RC were carefully considered; if the widgets were too small, the comparison would be unfair. Thus, the entire window was split into 4×4 regions, allowing for generous hit targets. The four corner regions were the scale (Figure 8h), and others were move widgets (Figure 8f, g). When the user fired the ray at a move widget by pulling the trigger on the controller, the window was translated with the same absolute size (i.e., not projected).

Participant

12 volunteers (6 female, 6 male, ages 20–27) participated. All but 1 were right handed. 5 were left eye, and 7 were right eye dominant [15]. All but 1 had previous VR experience.

Procedure

PWC was customized for each participant's dominant eye. During a warm-up session, participants could try the task a few times until they expressed that they felt confident. They sat on a comfortable stool and could take a break at any time. After completing all tasks, they completed a survey and were interviewed for comments. For counterbalancing, 6 of them used PWC first, and the others used RC first.

Task

In each task, participants had to move and scale a window of position (D_1, θ_1) and size (W_1) to match those of the target (D_2, θ_2, W_2) (Figure 8, 9). The windows and targets were placed on a fractal grid of equilateral triangles (side length = 1, 2 m) (Figure 9) to simultaneously compare the discrete levels of various factors in the later statistical analyses: For example, moving a window from A to a target at C requires a linear movement of 4 m and an angular movement of -60°, whereas A to D requires 2 m and 30° respectively. The widths of windows and targets were $W_{abs} = \sqrt{3}\times\{\ 0.4, 0.8\ \}$ m at B, D, F, H, I, & L, and 0.4 & 0.8 m at the other points so that the apparent width, defined as the projected width on a picture plane 1 m away from the eye, could be any one of $W_{app} = 0.4\times\{\ ¼, ½, 1, 2\ \}$ m for all windows and targets.

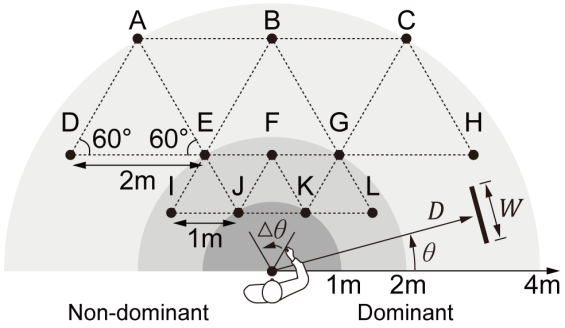

Figure 9. Windows and targets appeared on A~L. The user performed move and scale tasks using Projective Windows and a widget-based ray-casting baseline technique.

Both the window and the target had a fixed aspect ratio (4:3) and directly faced the participant. The target had a large wall around it and a size tolerance of ±10%, and each task ended when the window matched the target within the tolerance. The task completion time and operations count (one grab-release for PWC, and one trigger-untrigger for RC) were measured. The participant pressed a virtual button placed on the lap with the performing hand to begin the next task, which ensured that the hand always took off from the same position. Each participant performed 140×2 tasks.

Performance Results

In total, 3360 data points were gathered. The left-handed participant's angular distance was inverted so that a positive angular distance indicated a swing from the dominant hand side to the non-dominant one (Figure 9).

Task completion time and operations count by technique

A paired t-test indicated that the mean task completion time of PWC (4.5 s, standard error [SE]: 0.03 s) was significantly lower (22%, $t_{1679} = 26$, $p < 0.01$) than that of RC (5.8 s, SE: 0.05 s) (Figure 10a). Likewise, the operations count of PWC (1.5, SE: 0.02) was significantly lower (40%, $t_{1679} = 32$, $p < 0.01$) than that of RC (2.5, SE: 0.03) (Figure 10b).

Task completion time with PWC by factor

An analysis of variance (ANOVA) revealed a significant effect of the **linear distance** (Figure 11a■, $F_{4,1675} = 45$, $p < 0.01$) on the mean task completion time, and a post-hoc analysis revealed a significant difference among all distances except for between 3 & 4 m. For the **angular distance** (Figure 11b■, $F_{8,1671} = 11$, $p < 0.01$), the post-hoc analysis indicated no significant difference except for at 120°, at which it was significantly higher (6.0 s). For the **absolute size ratio** (Figure 11c■, $F_{8,1671} = 17$, $p < 0.01$), the post-hoc analysis indicated two distinct groups for the mean time, among 0.3, ½, 2, & 3.5, and among 0.6, 0.9, 1, 1.2, & 1.7. However, a more recognizable pattern was observed for the **apparent size ratio** (Figure 11d■, $F_{6,1673} = 54$, $p < 0.01$), where the post-hoc analysis revealed that the mean time at an apparent size ratio of 1 (3.9 s) was significantly lower compared with all of the others. Differences were significant between 1 & ½, and ½ & ¼, but not between ¼ & ⅛. Likewise, they were significant between 1 & 2, and 2 & 4, but not between 4 & 8. Furthermore, no significant difference was found between making the apparent size of the window bigger or smaller by a factor of 8.

Task completion time with RC by factor

Although the ANOVA revealed significant effects of the **linear distance** (Figure 11a■, $F_{4,1675} = 19$, $p < 0.01$) and **angular distance** (Figure 11b■, $F_{8,1671} = 13$, $p < 0.01$) on the mean time, the post-hoc analysis revealed no recognizable pattern. On the other hand, for the **absolute size ratio** (Figure 11c■, $F_{8,1671} = 93$, $p < 0.01$), the post-hoc analysis revealed that the mean time at an absolute size ratio of 1 (3.8 s) was significantly lower compared with all of the others. No significant difference was found among 0.3, ½, 2, & 3.5, and among 0.6, 0.9, 1.2, & 1.7, revealing two distinct symmetric groups of the mean time centered on 1. In addition, for the **apparent size ratio** (Figure 11d■, $F_{6,1673} = 23$, $p < 0.01$), the post-hoc analysis revealed that the mean time at an apparent size ratio of 8 (11 s) was significantly higher compared with all of the others, and mean times were stable across ½, 1, & 2.

Operations count with PWC by factor

Both the **linear distance** (Figure 12a■, $F_{4,1675} = 8.9$, $p < 0.01$) and **angular distance** (Figure 12b■, $F_{8,1671} = 5.2$, $p < 0.01$) had significant effects on the mean operations count, but the post-hoc analysis indicated no recognizable pattern. For the **absolute size ratio** (Figure 12c■, $F_{8,1761} = 15$, $p < 0.01$), the post-hoc analysis revealed that the mean counts at 0.3 & ½ were significantly higher. For the **apparent size ratio** (Figure 12d■, $F_{6,1673} = 51$, $p < 0.01$), the post-hoc analysis revealed that the mean count differences were not significant between ⅛ & ¼, 1 & 2, and 4 & 8. There was no significant mean count difference between making the apparent size of the window bigger or smaller by a factor of 8.

Operations count with RC by factor

Although the ANOVA revealed a significant effect of the **linear distance** (Figure 12a■, $F_{4,1675} = 15$, $p < 0.01$) and **angular distance** (Figure 12b■, $F_{8,1671} = 17$, $p < 0.01$) on the mean operations count, the post-hoc analysis revealed no recognizable pattern. On the other hand, for the **absolute size ratio** (Figure 12c■, $F_{8,1671} = 59$, $p < 0.01$), the post-hoc analysis indicated that the mean operations count at an absolute size ratio of 1 (1.5) was significantly lower compared with all of the others. Moreover, for the **apparent size ratio** (Figure 12d■, $F_{6,1673} = 30$, $p < 0.01$), the post-hoc analysis revealed that the operations count was stable across ½, 1, & 2 and that it was significantly higher for ⅛, 4, & 8. The count was the highest at an apparent size ratio of 8 (5.8), significantly more so than that at ⅛ (4.1).

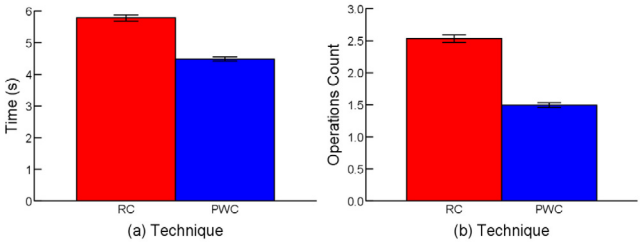

Figure 10. (a) Mean task completion time and (b) mean count of operations performed for each task by technique. PWC performed better for both (error bars: +/−2 SE).

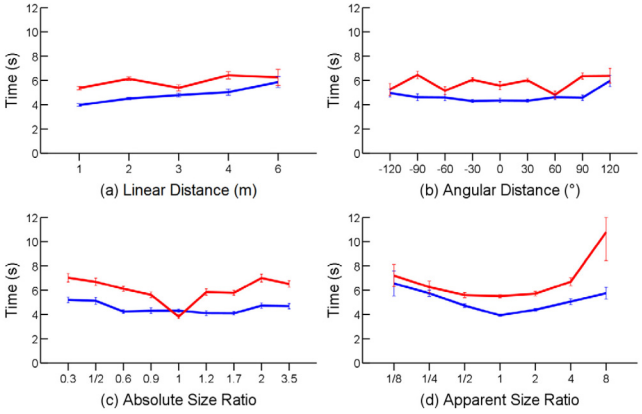

Figure 11. Mean task completion time by (a) linear and (b) angular distances, and (c) absolute and (d) apparent size ratios (■: RC, ■: PWC, error bars: +/−2 SE).

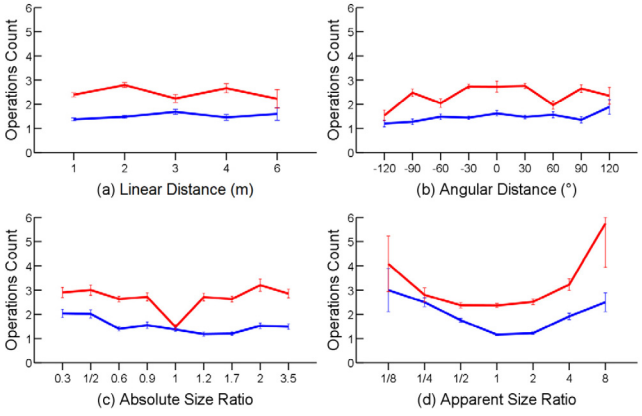

Figure 12. Mean operations count by (a) linear and (b) angular distances, and (c) absolute and (d) apparent size ratios (■: RC, ■: PWC, error bars: +/−2 SE).

Preferences Result

We performed a paired t-test to find the effect of the technique on the survey questions (Figure 13). A significant effect was found on **Q2** ($t_{11} = 3.2$, $p < 0.01$), where RC was considered to be more comfortable; **Q3** ($t_{11} = 3.0$, $p < 0.05$), where PWC was considered to be faster; **Q11** ($t_{11} = 4.4$, $p < 0.01$), where RC was considered to be more suitable for laborious tasks; and **Q12** ($t_{11} = 4.1$, $p < 0.01$), where PWC was considered to be more suitable for abrupt tasks.

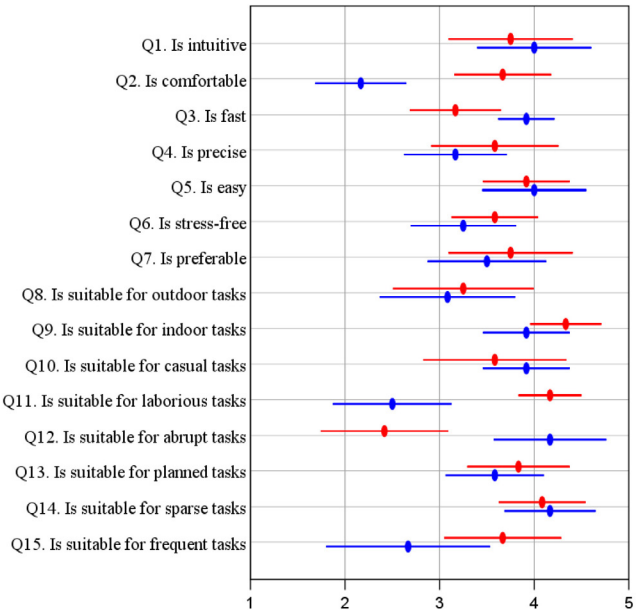

Figure 13. Survey result on five-point Likert scale (1: strongly disagree, 2: disagree, 3: neutral, 4: agree, 5: strongly agree, ■: RC, ■: PWC, ellipse: mean, error bars: +/−2 SE).

EXPLORATION SESSION

After the tasks, surveys, and interviews regarding PWC and RC in the experiment session, participants performed the same tasks with PWH, and were surveyed and interviewed.

For PWH, the mean task completion time was 4.7 s (SE: 0.04 s), and the mean operations count was 2.2 (SE: 0.03). In processing the data, operations that lasted less than 0.1 s or that moved less than 1 cm were considered to be sensor glitches, and they were disregarded from the calculation.

PWH scored above 3.0 for all questions, and 4.0 or above for **Q1** (intuitive, mean = 4.7), **Q3** (fast, 4.3), **Q4** (precise, 4.0), **Q5** (easy, 4.5), **Q7** (preferable, 4.3), **Q9** (suitable for indoor tasks, 4.3), **Q10** (suitable for casual tasks, 4.7), **Q12** (suitable for abrupt tasks, 4.4), **Q13** (suitable for planned task, 4.0), and **Q14** (suitable for sparse tasks, 4.6).

Due to the aforementioned confounds, a session was held separately for PWH, and thus, the above results were not directly comparable to those of PWC and RC. Yet, this separate session helped us explore the qualitative aspects of the user experience that can be expected when hand-tracking technology undoubtedly matures in the near future. The results from both the experiment and the exploration sessions are discussed in the next section.

DISCUSSION

In this section, we interpret the quantitative and qualitative results from the experiment and evaluation sessions.

PW is easy to learn

Performing Projective Windows for the first time, participants were a little thrown-off momentarily, but *"aha"* moments quickly followed. They were amused that they could easily control how big the window would appear when projected on a distant wall by directly holding and moving it to and away from their faces.

The pre-planning required for PW—for example, grabbing a window farther from the face first to bring it closer for enlarging, and the opposite for shrinking—however, took some practice, although most participants quickly became adept (< 10 tries). Some even discovered how to use PW before being told.

PW is to RC as a touch is to a mouse click

Participants found both PWC and RC to be generally intuitive, precise, easy, stress free, preferable, and suitable for outdoor, indoor, casual, planned, sparse, and frequent tasks (Q1, 4-10, 13-15). Regarding the overall impression, participants noted that RC *"felt like the familiar desktop mouse cursor,"* whereas PW *"felt like touch interaction on a smartphone"* (P8, 10). Participants found themselves *"naturally reaching out to grab the window"* (P11). Also, PW's fluid handover between absolute and apparent sizes allowed them to *"focus on how it would appear"* (P11).

PW is and feels fast

Participants answered that they felt they could perform tasks more quickly with PWC (Q3) because it allowed them to *"move and scale at the same time"* (P10), as supported by the significantly lower mean time and operations count of PWC compared with those of RC. With RC, participants *"had to look for and aim at widgets"* (P1), and during the controlling of the windows across large distances, *"the window moved too fast"* and *"hand tremor was amplified"* (P4, 5, 6, 7, 10), to the detriment of the performance of RC.

PW is better without a hardware controller

PWC was less comfortable compared with RC (Q2), as it required considerable arm movement while one was carrying the weight of the controller (P3, 4, 6, 7, 8). Participants found the wrist orientation and the pulling of the controller trigger to be *"unnatural"* for PW (P1, 5, 7).

PW is better with the hand

When using the hand for PWH, participants noted that they felt like *"physically touching the window like a real object"* (P1, 3, 8, 9, 10) and could better plan their hand motions: *"When I could see my own hand holding the window, I knew exactly how much closer or farther to move it"* (P1). This hints at the fact that they could use kinesthetic memory to anticipate the amount of motion required to make the hand appear a certain size and consequently the window appear a desired size, relative to the hand holding it.

PW is advantageous for greater scaling operations

Participants performed trivial scaling operations quickly, e.g., with an apparent size ratio of 1 for PWC (Figure 11d) and with an absolute size ratio of 1 for RC (Figure 11c), irrespective of the technique. However, as they had to make the window considerably bigger or smaller, the two performance metrics diverged between the techniques, showing that the greater the scaling, the more advantageous PW is over RC.

PW is more suited to abrupt, unintensive tasks

Participants found PWC to be more suitable for abrupt tasks (Q12), such as reacting to a pop-up window. However, they found RC to be more suitable for laborious tasks (Q11), such as computer graphics, where the position and size of an object may need to be meticulously adjusted, as they could *"put it down first, and then adjust the size, focusing on each step"* (P3, 5, 10). This points out the need for a clutching mechanism, whereby users can suspend a window in a temporary position for further actions.

Apparent size matters in AR and VR

For both PWC and RC and both the mean task completion time and operations count, the performance patterns were the most recognizable for the apparent size ratios (Figure 11d, 12d). This indicates that the apparent sizes of the visual elements within a spatial user interface may be a critical factor in spatial task performances to which the designers of future AR and VR interfaces should pay close attention.

USER SCENARIOS

We prototyped everyday computing scenarios of Projective Windows (Figure 14) using our implementation (Figure 6).

- *Scale and position anywhere*: In a design studio scenario (Figure 14a), the user can pull picture windows out of a laptop screen and easily scale and place them anywhere on nearby walls for visual reference, just as he or she would sticky notes, but with the ability to freely change the size.
- *Cross-device jumps*: In the design studio (Figure 14a), the user can pick up a window from a laptop screen and place it on a tablet device to quickly change the input from typing to drawing, without having to swap applications.
- *Cloning physical objects*: In a study scenario (Figure 14b), the user can perform the grab gesture to instantly scan a notebook page and then generate a projective window from it, to scale and place it anywhere for reference.
- *Using proximity and geometry*: In a living room scenario (Figure 14c), the user can pick up a small movie window from a nearby table, play the preview of the movie by bringing it closer to the face [1], and then start playing the movie by projecting it onto a vertical wall.
- *AR and VR compatibility*: In VR (Figure 14d), the user can use the entire unbounded scene as a workspace, even projecting windows across large distances.

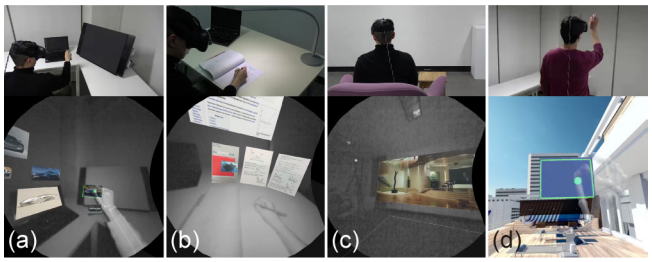

Figure 14. User scenarios of Projective Windows in a (a) design studio, (b) study, (c) living room, and (d) VR scene.

CONCLUSION & FUTURE WORK

We proposed Projective Windows, a technique for arranging 3D planar windows in AR and VR workspaces, which is, thanks to its strategic use of the absolute and apparent sizes of a window at various stages of the interaction, minimal, direct, and intuitive, requiring only a single continuous flow of hand gesture and no other controllers and UI widgets.

We evaluated our technique quantitatively and qualitatively, and found that it performed significantly better against a baseline technique for two key metrics: task completion time (22%) and operations count (40%). We furthermore demonstrated the relevance and usefulness of exploiting projective geometry in designing spatial UIs.

Some speculate that AR and VR devices might come to replace all screen-based devices in the future [17]. For Projective Windows to facilitate seamless interactions with 2D content in future immersive 3D experiences, more work is needed on the systematic use of ad-hoc surfaces, such as the palm or a handheld plane prop, and surrounding geometries, such as wall edges or ceilings. In addition, it should better integrate with other relevant 3D windows techniques [5, 9, 10, 14].

REFERENCES

1. Till Ballendat, Nicolai Marquardt, and Saul Greenberg. 2010. Proxemic interaction: designing for a proximity and orientation-aware environment. In *Proc. ITS '10*, 121-130.
2. Richard W. Bukowski and Carlo H. Séquin. 1995. Object associations: a simple and practical approach to virtual 3D manipulation. In *Proc. I3D '95*, 131-ff.
3. Li-Wei Chan, Hui-Shan Kao, Mike Y. Chen, Ming-Sui Lee, Jane Hsu, and Yi-Ping Hung. 2010. Touching the void: direct-touch interaction for intangible displays. In *Proc. CHI '10*, 2625-2634.
4. Barrett Ens, Juan David Hincapié-Ramos, and Pourang Irani. 2014. Ethereal planes: a design framework for 2D information space in 3D mixed reality environments. In *Proc. SUI '14*, 2-12.
5. Barrett M. Ens, Rory Finnegan, and Pourang P. Irani. 2014. The personal cockpit: a spatial interface for effective task switching on head-worn displays. In *Proc. CHI '14*, 3171-3180.

6. Steven Feiner, Blair MacIntyre, Marcus Haupt, and Eliot Solomon. 1993. Windows on the world: 2D windows for 3D augmented reality. In *Proc. UIST '93*,145-155.
7. Tovi Grossman and Ravin Balakrishnan. 2005. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proc. CHI '05*, 281-290.
8. Jan Gugenheimer, David Dobbelstein, Christian Winkler, Gabriel Haas, and Enrico Rukzio. 2016. FaceTouch: enabling touch interaction in display fixed UIs for mobile virtual reality. In *Proc. UIST '16*, 49-60.
9. Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. 2011. OmniTouch: wearable multitouch interaction everywhere. In *Proc. UIST '11*, 441-450.
10. Valentin Heun, James Hobin, and Pattie Maes. 2013. Reality editor: programming smarter objects. In *Adj. Proc. UbiComp '13*, 307-310.
11. Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. 1994. A survey of design issues in spatial input. In *Proc. UIST '94*, 213-222.
12. Joon Hyub Lee, Seok-Hyung Bae, Jinyung Jung, and Hayan Choi. 2012. Transparent display interaction without binocular parallax. In *Adj. Proc. UIST '12*, 97-98.
13. Joon Hyub Lee and Seok-Hyung Bae. 2013. Binocular cursor: enabling selection on transparent displays troubled by binocular parallax. In *Proc. CHI '13*, 3169-3172.
14. Frank Chun Yat Li, David Dearman, and Khai N. Truong. 2009. Virtual shelves: interactions with orientation aware devices. In *Proc. UIST '09*, 125-128.
15. Alistair P. Mapp, Hiroshi Ono, and Raphael Barbeito. 2003. What does the dominant eye dominate? A brief and somewhat contentious review. *Attention, Perception, & Psychophysics*, 65(2), 310-317.
16. Mark R. Mine. 1995. Virtual Environment Interaction Techniques. *UNC Chapel Hill CS Dept*.
17. Kyle Orland. 2014. Will VR make flat panels obsolete? Oculus' founder gives it 20 years. *Ars Technica*. Retrieved September 19, 2017 from https://arstechnica.com/gaming/2014/04/will-vr-make-flat-panels-obsolete-oculus-founder-gives-it-20-years/
18. Jeffrey S. Pierce, Andrew S. Forsberg, Matthew J. Conway, Seung Hong, Robert C. Zeleznik, and Mark R. Mine. 1997. Image plane interaction techniques in 3D immersive environments. In *Proc. I3D '97*, 39-ff.
19. Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. 1996. The go-go interaction technique: non-linear mapping for direct manipulation in VR. In *Proc. UIST '96*, 79-80.
20. George Robertson, Maarten van Dantzich, Daniel Robbins, Mary Czerwinski, Ken Hinckley, Kirsten Risden, David Thiel, and Vadim Gorokhovsky. 2000. The Task Gallery: a 3D window manager. In *Proc. CHI '00*, 494-501.
21. David Canfield Smith, Charles Irby, Ralph Kimball, Bill Verplank, and Eric Harslem. 1987. Designing the Star user interface. *Byte*, 7, 242-282.
22. Paul S. Strauss and Rikk Carey. 1992. An object-oriented 3D graphics toolkit. In *Proc. SIGGRAPH '92*, 341-349.
23. Dimitar Valkov, Frank Steinicke, Gerd Bruder, and Klaus Hinrichs. 2011. 2D touching of 3D stereoscopic objects. In *Proc. CHI '11*, 1353-1362.
24. Mark Weiser. 1999. The computer for the 21st century. *Scientific American*, 265(3), 94-104.