SmartVR Pointer: Using Smartphones and Gaze Orientation for Selection and Navigation in Virtual Reality

Brianna Rachel McDonald , Qingyu Zhang , Aiur Zambalovich Nanzatov , Lourdes Pena-Castillo * , Oscar Meruvia-Pastor *

Posted Date: 28 May 2024

doi: 10.20944/preprints202405.1815.v1

Keywords: Smartphones; Virtual Reality; Head-Mounted Displays; HMD Interaction Techniques; Mobile Devices in VR; Image-Based Tracking; Human-Computer Interaction; Mobile Computing; Touch Input

Article

SmartVR Pointer: Using Smartphones and Gaze Orientation for Selection and Navigation in Virtual Reality

Brianna McDonald [1], Qingyu Zhang [1], Aiur Nanzatov [1], Lourdes Peña-Castillo [1,2] and Oscar Meruvia-Pastor [1,*]

[1] Department of Computer Science, Memorial University of Newfoundland
[2] Department of Biology, Memorial University of Newfoundland
* Correspondence: oscar@mun.ca

Abstract: Some of the barriers preventing Virtual Reality (VR) from being widely adopted are the cost and unfamiliarity of VR systems. Here we propose that in many cases the specialized controllers shipped with most VR head-mounted displays can be replaced by a regular smartphone, cutting the cost of the system, and allowing users to interact in VR using a device they are already familiar with. To achieve this, we developed SmartVR Pointer, an approach that uses smartphones as a replacement for the specialized controllers for two essential operations in VR: selection and navigation by teleporting. In SmartVR Pointer a camera mounted on the head-mounted display (HMD) is tilted downwards so that it points to where the user will naturally be holding their phone in front of them. SmartVR Pointer supports three selection modalities: tracker-based,gaze-based, and combined/hybrid. In the tracker-based SmartVR pointer selection we use image-based tracking to track a QR code displayed on the phone screen and them map the phone's position to a pointer shown within the field of view of the camera in the Virtual Environment. In the gaze-based selection modality the user places the pointer using their gaze and taps on the phone for selection. The combined technique is a hybrid between head-gaze-based interaction in VR and smartphone-based Augmented Reality. It allows the user to control a VR pointer that behaves like a mouse pointer by moving their smartphone to select objects within the virtual environment, and to interact with the selected objects using the smartphone's touch screen. The touchscreen is used for selection and dragging. SmartVR Pointer is simple and requires no calibration and no complex hardware assembly or disassembly. We demonstrate successful interactive applications of SmartVR Pointer in a VR environment with a demo where the user navigates in the virtual environment using teleportation points on the floor and then solves a Tetris-style key-and-lock challenge.

Keywords: Smartphones; Virtual Reality; Head-Mounted Displays; HMD Interaction Techniques; Mobile Devices in VR; Image-Based Tracking; Human-Computer Interaction; Mobile Computing; Touch Input

1. Introduction

Virtual reality (VR) typically refers to immersive graphics systems where the user can navigate around computer-generated virtual environments and interact with 3D objects inside the environment. VR systems are usually accompanied by a set of specialized controllers held by the user in each hand and used to perform actions in the environment. In this work we show how the specialized controllers used for VR could be replaced by regular smartphones in particular applications. This would allow users to use a device they already own and are familiar with, which reduces the cost of the system and makes VR more accessible to the average consumer. Moreover, smartphones contain additional sensors and displays which can also be used to improve the interaction capabilities of regular VR systems, as suggested in by previous researchers [1–6].

To demonstrate the feasibility of SmartVR Pointer, we implemented two sample applications. The first of these applications is a navigation task where the user navigates around the VR environment by selecting footprint-shaped teleport markers placed on the floor using a pointer that looks like a mouse pointer, which we call a virtual pointer or simply "VR pointer". The user can teleport by

 doi:10.20944/preprints202405.1815.v1

aligning the VR pointer with the footprints and tapping the phone screen to jump to the desired location. The second application uses a key-and-lock mechanism inspired on the video game Tetris. Here, the user sees a sample Tetris-style scene where there are blocks and one or more empty spaces where additional blocks can fit in between the existing blocks. The user must select the correct blocks and adjust their position and orientation so that they fit inside the provided empty spaces. These applications demonstrate that the proposed technique can be used to conveniently perform many common tasks in VR, such as navigation, selecting, positioning, and determining the orientation of 3D objects.

2. Related Work

Early work in this area explored the idea of connecting a smartphone to a VR or Augmented Reality (AR) system and allowing the user to interact with 3D objects using a combination of HMD's and Hand-Held Devices (HHD's) [7,8]. A few different approaches using a smartphone or other smart devices for interaction in VR and AR have been proposed since, with and without some type of external tracking or hardware adaptations [1–6,9–15]. While some approaches focus on using built-in sensors in smart devices to control hand held ray casters or pointers [16–18], our approach is mainly image-based, like TrackCap and PAIR [4,5].

TrackCap, an image-based approach proposed by Mohr et al., involves mounting a cap on top of an HMD and using a smartphone camera to track an image printed on the underside of the cap [4]. This approach allowed them to track the position of the phone relative to the HMD and use it as an AR controller. One potential drawback of this approach is that strong back-lighting from the ceiling can cause poor visibility of the image on the cap, and users may have to compensate this by using an additional light source underneath the cap. Since in our approach the image target is displayed on the phone screen, it does not suffer from this issue since the brightness of the phone screen can be automatically adjusted to work well under different lighting conditions.

Another alternate approach for using a smartphone as a controller for VR was proposed by Hattori and Hirai, who proposed an image-based detection method based on "plane finding" [15]. This allows the phone to estimate its own position based on ambient information. However, their approach requires a specific synchronization process and their ray-casting mechanism may go out of alignment due to poor synchronization, whereas SmartVR Pointer does not require any form of synchronization.

Unlu and Xiao [5] proposed using a combination of a touch screen input along with internal and image-based tracking of the smartphone sensors to allow the use of the smartphone as a controller for several types of AR applications using the HoloLens. Their approach uses multiple smaller optical markers displayed around the edges of a phone screen along with built-in sensors in both phone and HoloLens. Multiple markers are used to provide redundancy to track the phone in case any of the markers get occluded by the user's hands during interaction. Our approach, on the other hand, implements a virtual pointer by placing a large marker at the center of the phone and casting a ray from the viewer towards the center of this marker. Most recently, HTC released the Vive Flow, a commercial smartphone-controlled VR headset aimed at the Mindfulness and Autonomous Sensory Meridian Response (ASMR) VR market [19]. It uses the phone sensors and its touchscreen to allow users to select content visible through the HMD. The Vive Flow uses a wand or laser-pointer paradigm, where the wand acts as a hand-held ray caster for object selection, whereas the touchscreen is used to refine the selection or to offer various menu options. This fundamentally differs from our approach that uses the smartphone utilizing optical, image-based tracking. In addition, HTC's implementation is proprietary, whereas SmartVR Pointer is distributed openly and can be downloaded here [20] under a creative commons (CC) license.

Researchers have also focused on using other smart devices such as smartwatches for interaction in VR [10,11,13,14]. One example is TickTockRay proposed by Kharlamov et al. [10] which uses the built-in sensors in a smartwatch to track the user's arm movements and control a ray-caster. Conversely,

 doi:10.20944/preprints202405.1815.v1

the solutions proposed by Park and Lee [14] as well as Kim and Woo [11] use hand tracking to allow users to interact with the virtual environment by performing hand gestures in front of depth sensors. A potential issue with using a smartwatch is arm fatigue caused by the user having to keep their arms raised while performing repetitive hand and wrist motions. Additionally, smartwatches are less widely adopted and less likely to be familiar to users than smartphones.

Other researchers have explored using devices that are not necessarily smart devices but contain Inertial Measurement Unit (IMU) sensors [9,12]. This includes the work by Young et al., where an arm-mounted input device was tested, and the work by Hincapié-Ramos et al., which proposed using a smartphone-like device as a controller. These methods are prone to some degree of error accumulation caused by drift and may need re-centering or re-calibration after prolonged use.

Researchers have also proposed using smartphones and smartwatches for pointing and interacting with large 3D displays [16–18]. These solutions use the gyroscopes and accelerometers built in these devices to control ray-casters for pointing and interacting in a way that could be applied to large displays within the VR. Like most VR controllers, they use a laser-pointer metaphor, where a ray is cast from the HHD or controller towards the target or selection region. By contrast, SmartVR Pointer casts a ray from the gaze of the user (determined by the HMD's camera position in world coordinates) towards the center of the QR code on the smartphone, allowing the user to refine the placement of a virtual pointer by displacing the smartphone itself.

3. Methods

To use the SmartVR Pointer, a camera is mounted on the HMD at the height of the user's face and tilted downwards to the region where the user would hold their smartphone comfortably, approximately at a 45 degree angle (see Figures 1 and 2). We propose three methods to define the position of the pointer used for target selection and dragging: 1) QR-code tracking; 2) Gaze-based tracking; and 3) a combination of both.

Figure 1. Setup for SmartVR Pointer with the camera mounted on top of the HTC Vive Cosmos Elite HMD.

 doi:10.20944/preprints202405.1815.v1

Figure 2. Setup for SmartVR Pointer with the camera mounted at the bottom of the Meta Quest 2 HMD.

3.1. QR Tracking-Based Mode

When the selection mode is activated, which can be done by having the user double-tap on the phone, we track an image of a QR code displayed on the smartphone screen using the Vuforia Engine [21]. PTC Vuforia provides an Augmented Reality (AR) Software Development Kit (SDK) in the form of a Unity plugin that is typically used for creating AR applications in mobile devices. Vuforia provides the functionality to easily track images and display virtual objects on top of them in AR, and we utilize Vuforia's tracking feature to track the QR code. As we track the QR code, we find out where the phone screen is located relative to the view of the camera on top of the HMD. We use this relative position of the phone to produce a pointer that the user can see inside the HMD, which is controlled by moving the smartphone within the field of view (FOV) of the camera and resembles a white mouse pointer, as shown in the accompanying video.

When the phone screen displaying the QR code displayed is within the camera's view, Vuforia is used to place a pointer on top of the center of the QR code. We can get the position of this pointer within the camera view (Figure 3 shows a red cube acting as a pointer). Additionally, there is a transparent 2D canvas object that covers the user's field of view inside the VR environment and is fixed to a camera plane in front of the user. We call this the VR canvas. The position of the pointer in screen coordinates is converted to the corresponding local coordinates within the canvas. These coordinates are then used to display a pointer (similar to a desktop mouse pointer) on the canvas, which the user can see within the VR (Figures 4 & 5).

 doi:10.20944/preprints202405.1815.v1

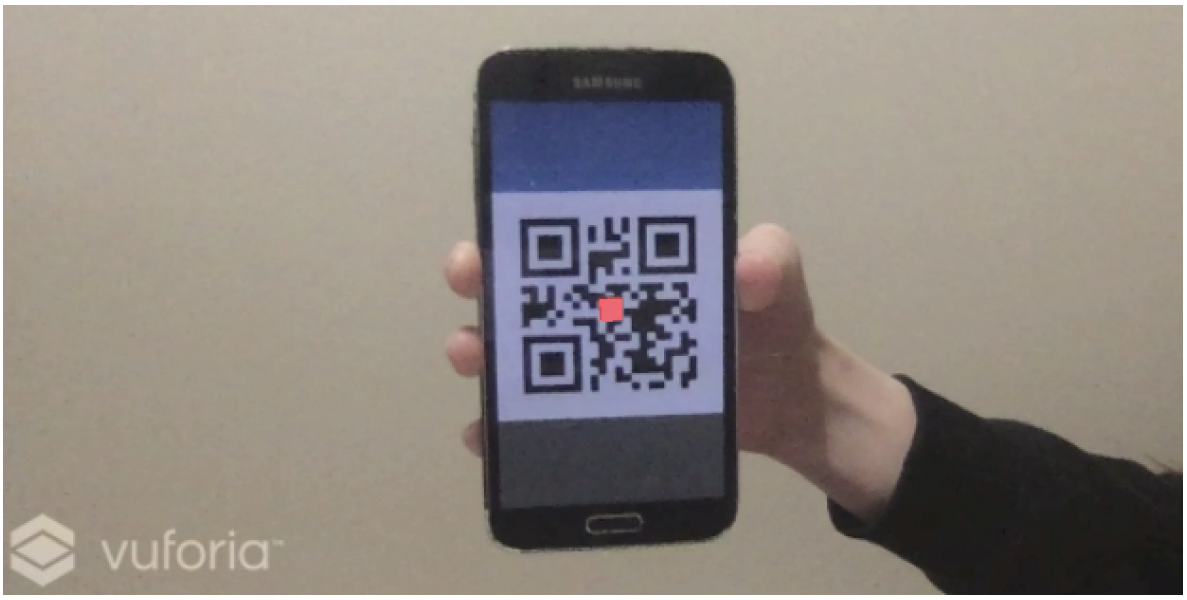

Figure 3. Using Vuforia Engine to display a red cube on top of the tracked QR code image displayed on the smartphone.

Figure 4. Teleporting application enabled by this technique where the user can select footprint-shaped teleport points to navigate around the VR environment.

 doi:10.20944/preprints202405.1815.v1

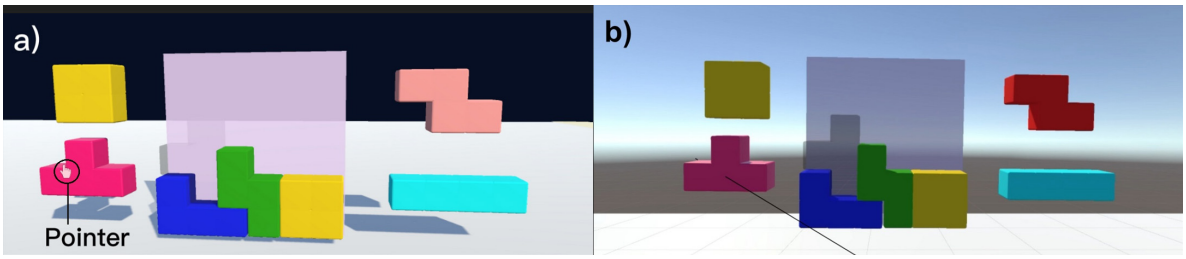

Figure 5. Illustration of how the ray-casting mechanism works. a) The view the player sees while wearing the HMD shows a pointer highlighting the Tetris block in the bottom left for selection. b) The developer view illustrates how a black ray is cast from the player towards the pointer, and through the Tetris block.

At the core of this technique is a mapping process that can be decomposed as two steps: in the first step, the Vuforia engine returns the coordinates of the QR code with respect to the real world camera as an X,Y,Z vector. We discard the Z coordinate (which is Vuforia's estimate of depth) and use the X,Y coordinates as the coordinates of the tracked object within the AR camera plane, which is a 2D plane that does not correspond to any location in the VR. We use the X,Y coordinates and the size of the AR camera plane to calculate the position of a pointer as a proportion of the width and height of the AR camera. These coordinates provide the relative displacement of the QR code within the field of view of the AR camera and are used in the second step. In step 2, we use the relative position of the tracked object to determine the actual position of the VR pointer within the VR canvas. This is done by determining the positions of the corners of the VR canvas in world coordinates, which vary according to the position and orientation of the HMD. Since we know the relative proportion of width and height of the tracked object in the AR camera, and we have the corners of the canvas in world coordinates, we can calculate the corresponding position of the VR Pointer in world coordinates. This way, the VR pointer is an object that always lies within the VR canvas.

As a result of this mapping process, the VR pointer is controlled by moving the phone and can be used as a pointer for interaction. This pointer function is achieved by creating a raycast from the user towards the pointer location on the canvas. The raycast extends indefinitely to hit objects in the background (Figure 5). It is invisible to the user and simply allows us to detect which 3D object is in line with the pointer and the user's point of view at any given time.

Lastly, there is a wireless transfer of data between the smartphone and the VR application running on a desktop computer. This allows the smartphone to communicate with the VR application and permits the user to perform various actions in VR by interacting with the phone screen. In particular, we can send a message to the VR application when the user presses their finger against the phone screen or releases their finger from it. We use this functionality to implement various types of clicking and hold-and-release interactions that are commonly used in VR. The proposed setup has the advantage that the system provides tactile/haptic feedback to the users, increasing their confidence during selection tasks. Touch is an important feature in systems that use smartphones in VR, as it has been shown that direct touch provides for faster and more precise interactions [22].

3.2. Gaze-Based Mode

In this mode, a red mouse pointer appears in the center of the field of view (FOV) of the user wearing the HMD. Users then move their head around to move the pointer around the scene. As the pointer is moved over eligible target locations, those locations become highlighted, indicating they are potential selection targets. The user then taps on the smartphone screen to confirm the target selection. In the case of a drag and drop operation, the object remains selected until the user stops touching the smartphone screen. Figure 6 illustrates, from a 3^{rd} person perspective, how this VR panel is attached to the HMD view and how a ray is cast from the camera to the pointer.

 doi:10.20944/preprints202405.1815.v1

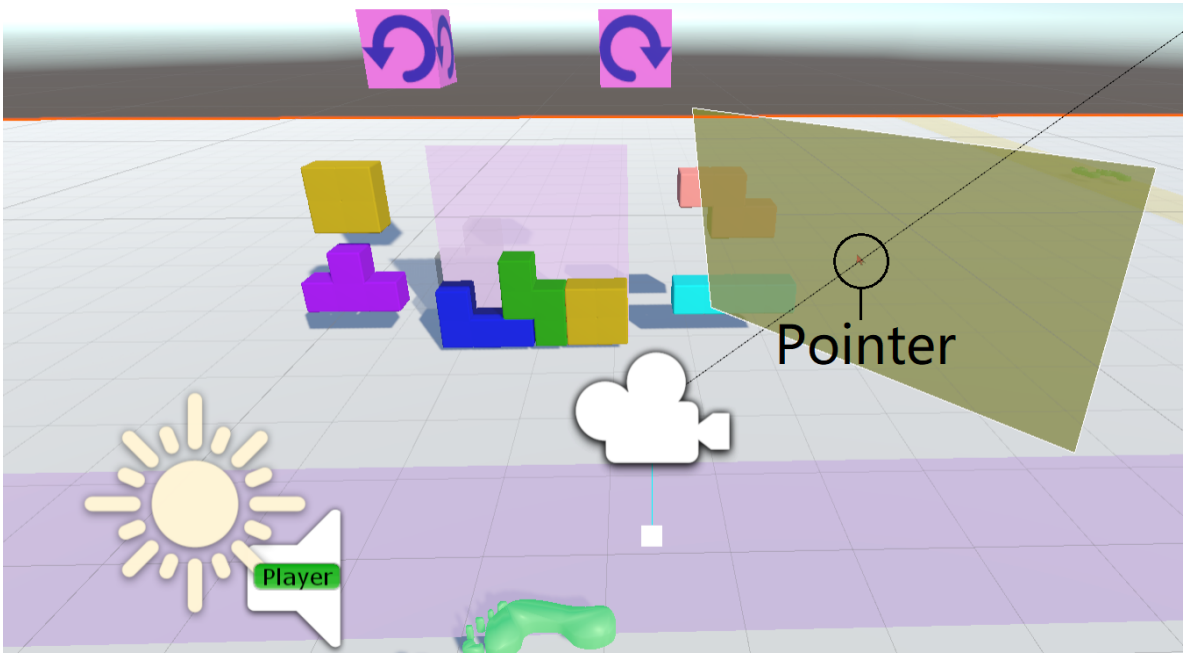

Figure 6. Screenshot of the Scene view in Unity, which illustrates the VR canvas (darkened here for emphasis). From the user's perspective the panel appears fixed in window coordinates.

3.3. Combined Mode

The combined mode uses both methods alternatively. It defaults to QR-based tracking when the QR code is visible from the point of view of the camera and shows a white mouse pointer while the tracking is active. If either the QR code is not found within the FOV of the user or the Vuforia plugin is unable to track the smartphone for any reason, the system turns into head-gaze-based selection [23] and the mouse pointer turns red to indicate the gaze-based mode is active. Then, the user adjusts the direction of their head to control the position of the red pointer on the 2D canvas and either select a footprint for navigation or the Tetris pieces to solve the puzzle. The selection and release of Tetris pieces is done by the user pressing the smartphone's touchscreen to initiate a drag operation and later drop an item by ending the contact with the touchscreen. If at any point the QR code becomes visible again, the VR pointer turns white and the user can refine its positioning by moving the mouse around in front of the viewer.

3.4. Advantages of SmartVR Pointer

One of the advantages of SmartVR Pointer is that it only requires a regular camera mounted on top of the HMD. Potentially, the built-in cameras of some HMDs could also be used, with the drawback that the user may need to hold the Smartphone a bit higher than in our proposed configuration, potentially increasing arm fatigue. An alternative configuration for the system is to setup a wireless camera (e.g. a GoPro), which could stream to the desktop app running the main application. This would remove the need for the extra cable to the webcam, but inserts network delay that comes from the wireless transmission of the camera frames, making the method slow to react to user's actions. In our current setup, we use an Intel RealSense Depth-Sensing camera because this camera can be connected to longer USB cables and provides depth video for other applications. For the demo applications, we implemented our solution on two HMDs: a Vive Cosmos (Figure 1) and a Meta Quest 2 (Figure 2). In both cases, we use the RealSense camera as a regular RGB camera.

Another advantage is that the solution we present does not need calibration. Slight deviations of the camera from the suggested position, which might be caused by removing it and mounting it back in place, would not be an issue as long as the camera is fixed with respect to the HMD during operation. This is because the user will be guided by the position of the virtual pointer within the

 doi:10.20944/preprints202405.1815.v1

VR environment in a way that is analogous to the use of a mouse pointer in the desktop metaphor: similar to when a user finds a regular mouse pointer by moving the mouse around and finding it on the screen as the mouse pointer responds, a SmartVR user will find the pointer in the VR as long as there is detection of the QR code in the smartphone, or a red pointer in the gaze-based mode appears, and then refine the positioning to the desired target. To point to a particular location using QR-code based tracking, the user naturally moves the smartphone to the left, right, up or down of the virtual panel in front of them, and that will translate in the virtual pointer moving accordingly. When QR tracking is turned off or unavailable, we use gaze-based positioning to establish the selection target, and the user then presses the smartphone screen to indicate a selection, or to start a drag-and-drop sequence.

Additionally, users can regulate the angle of the camera with respect to the HMD. As mentioned above, if the camera used for tracking faces straight towards the front, the user will need to raise their hands higher up or move their gaze towards the floor. If the camera faced directly towards the floor, the user would have to move the phone on a quasi-horizontal plane (perpendicular to the camera). We have found that when the camera faces about 45 degrees down from the line of sight (or front axis) towards the vertical axis (for a user standing in an upright position), it is possible for the user to comfortably hold the phone, looking at a plane that is tilted roughly 45 degrees with respect to the upright position, which is typical when using the smartphone.

3.5. SmartVR Pointer Availability

The entire Unity project for SmartVR pointer, including its source code, is freely available by accessing the link provided here [20].

4. Sample Applications

The first application we implemented involves allowing the player to navigate around a virtual environment (Figure 4). The user is presented with multiple paths that have 3D footprints at various points along them that represent teleport points. To select a teleport point, the user first lines up the pointer with one of the footprints and taps the smartphone screen. This will teleport the player by moving them to the location of the footprint and updating their rotation to match the direction that the selected footprint is facing.

The second application we implemented is a key-and-lock mechanism based on the video game Tetris (Figure 5). In this demo, the user is presented with a Tetris puzzle with an empty goal space represented by a semi-transparent gray block showing where another block can fit. To complete the task, the user must select, position, and rotate the correct block into the correct position in the puzzle in a drag and drop fashion. The user can do this by selecting the correct block by aligning the pointer with it and then pressing-and-holding their finger against the phone screen until releasing at the right spot and in the orientation that matches the Tetris challenge. This causes the block to move with the pointer so the user can drag it into position by moving the smartphone. The user can also rotate the block by swiping the phone screen with their thumb by 25 degrees to the left or right, and this will rotate the block 90 degrees in the direction that they are swiping. Alternatively, users can rotate the item by moving it within the region of a clockwise rotation widget that is shown on top of the tetris challenge. Each time the dragged object aligns with the rotation icon, the item is rotated accordingly.

4.1. Reproducibility

The sample applications for this project were implemented using the Unity 3D game engine. In order to reproduce SmartVR pointer, a user needs the Unity engine with the Vuforia plugin installed on their computer, an HMD, a webcam with a form factor and weight suitable for attaching it to the HMD, and a velcro strip with adhesives to attach the webcam to the HMD. Alternatively, a built-in camera in the HMD could be used if it points roughly towards the region where users hold their smartphones and the HMD's SDK allows access to camera's video stream. As mentioned in section 3,

 doi:10.20944/preprints202405.1815.v1

SmartVR Pointer does not require any calibration, but the camera needs to point roughly towards the general region where the user wants to hold their smartphone in a comfortable way.

5. User Study

We have performed a pilot user study with five participants using a game-style scenario to find out if users of SmartVR Pointer perform as well as users of commercial VR controllers for the applications we have discussed in Section 4. Figure 7 illustrates a model of a Viking Village, which we employed in our study. After 3 trial rounds, each participant performed 5 measured rounds moving around the Viking Village under 4 different conditions: Using a regular VR controller (the baseline condition), using Gaze-based SmartVR pointer selection, using Tracker-based SmartVR pointer selection, and finally the combined or hybrid mode, where users could use either the Tracker-based SmartVR pointer selection or the Gaze-based SmartVR pointer selection whenever the smartphone is not visible from the camera's point of view. Figure 8 shows the average teleporting completion times for the 5 users and Figure 9 shows the teleporting performance variability, measured as the standard deviation of the completion times over the 5 runs. Figure 10 shows the average teleporting completion times for the 5 users and Figure 11 shows the teleporting performance variability, measured as the standard deviation of the completion times over the 5 runs. The results are summarized in Table 1.

Table 1. Summary of the average completion times and the standard deviations for teleporting and the Tetris challenge.

Condition	Teleporting	StdDev	Tetris	StdDev
VR Controller	19.40	±3.04	**12.92**	±3.65
Gaze-Based SmartVR Pointer	**16.92**	±**2.53**	15.02	±**2.79**
QR Code Tracking SmartVR Pointer	50.33	±9.98	24.87	±5.27
Combined SmartVR Pointer	32.57	±7.22	22.15	±8.78

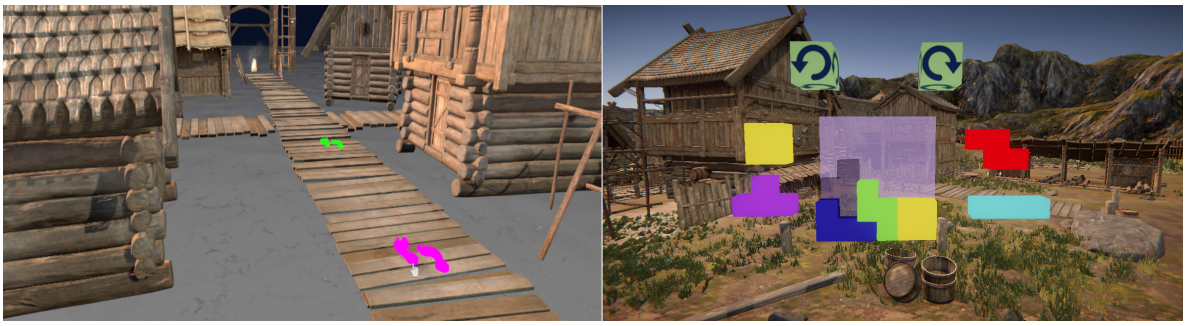

Figure 7. Illustration of SmartVR Pointer in a game-style scenario (an open source model of a Viking Village provided by Unity).

 doi:10.20944/preprints202405.1815.v1

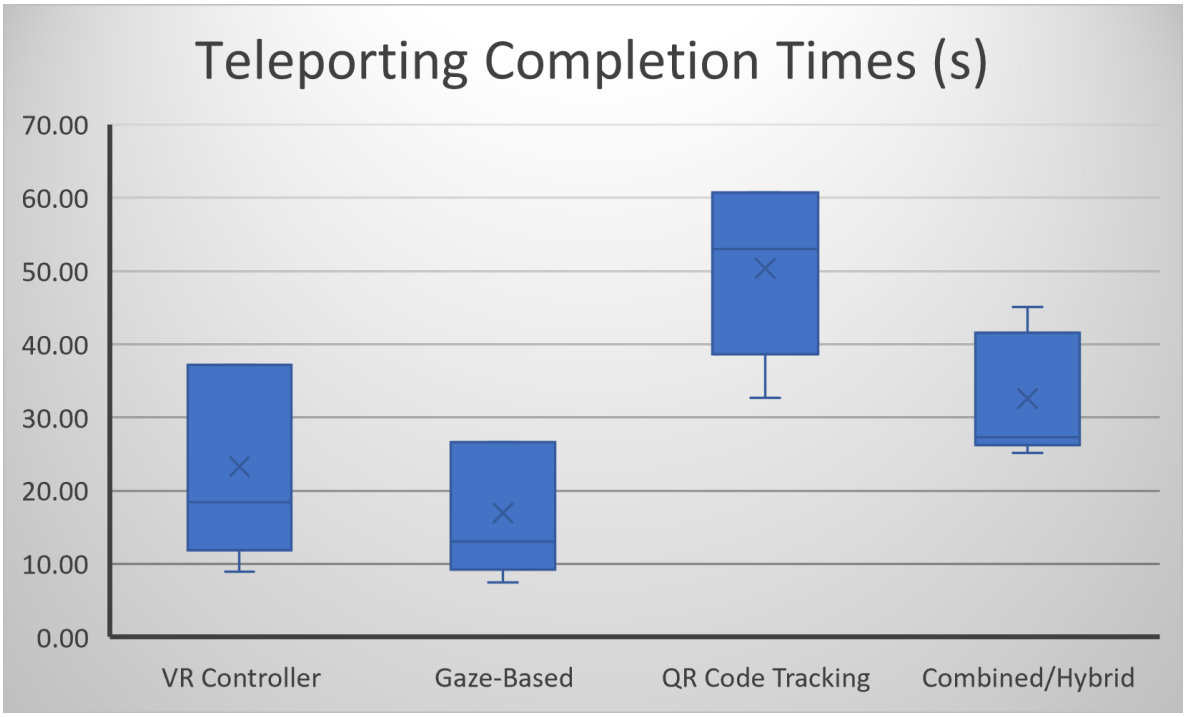

Figure 8. Average completion times for the 4 different conditions.

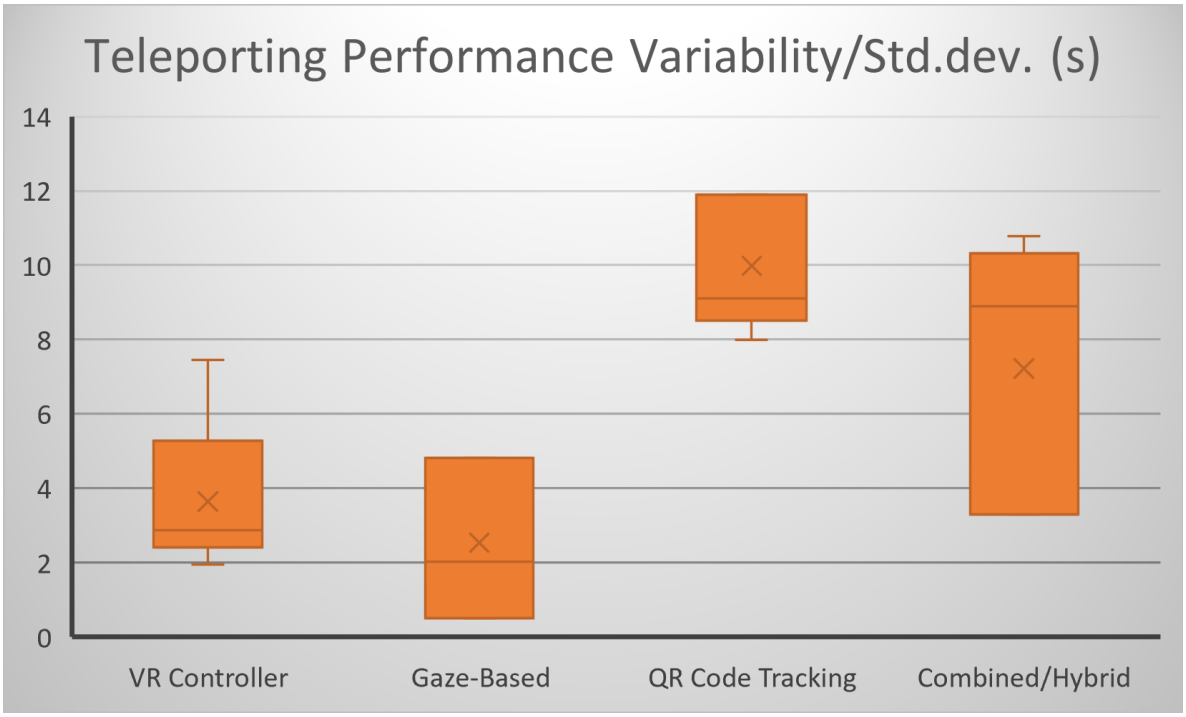

Figure 9. User performance variability in their completion times for the 4 different conditions.

 doi:10.20944/preprints202405.1815.v1

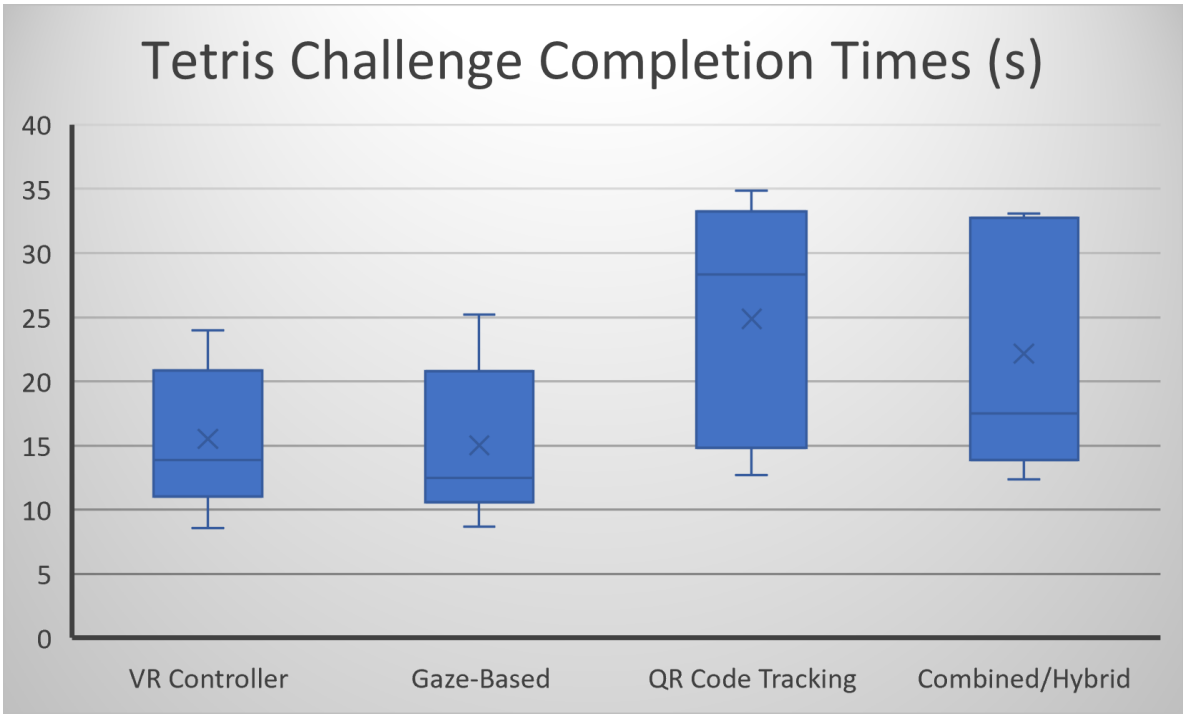

Figure 10. Tetris challenge completion times for the 4 different conditions.

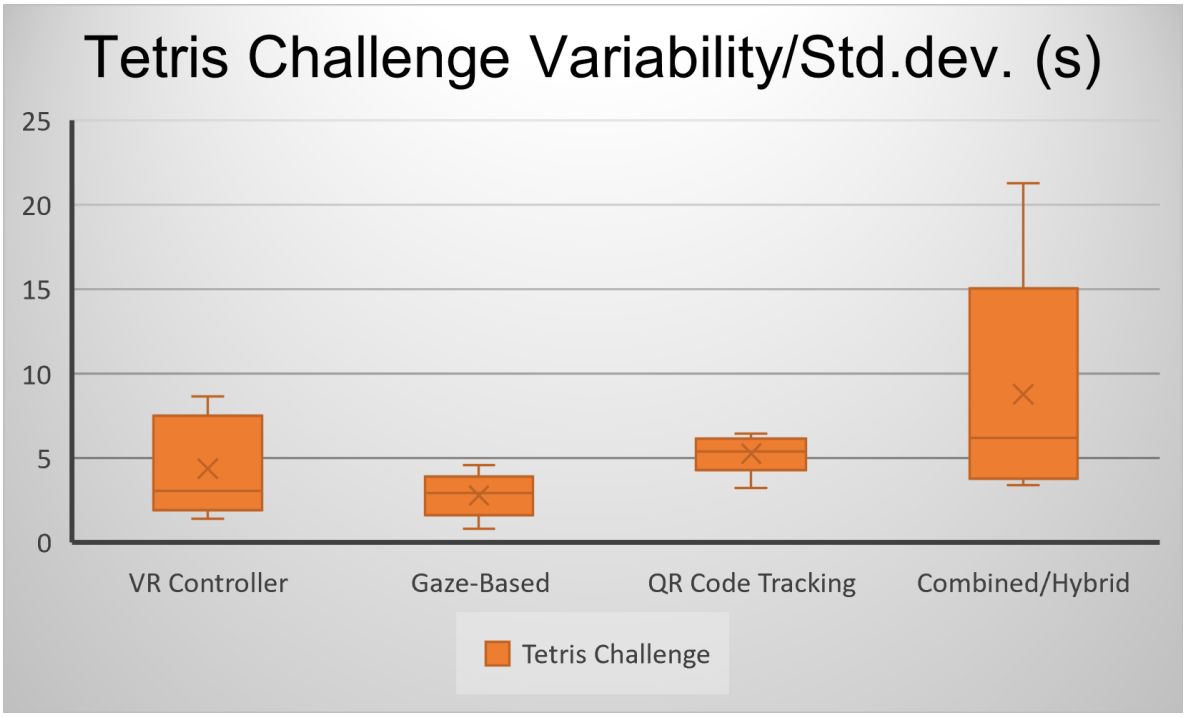

Figure 11. Performance variability for solving the Tetris challenge for the 4 different conditions.

Results suggest that, among the four conditions explored, users were most efficient using Gaze-based SmartVR Pointer, where they performed with comparable efficiency as when using regular VR controllers and with less variability overall. Some users even mentioned they preferred gaze-based selection over the regular VR controllers. All users performed slower in the combined condition and were slowest in the method where they could only navigate with the tracker-based option. In terms of task completion variability, the gaze-based condition seems to be the most consistent technique, with even slightly smaller standard deviation than the VR controllers (Table 1). On the other hand, the

 doi:10.20944/preprints202405.1815.v1

condition with highest task completion variability overall was the hybrid combination, where some users experienced as little variability as with the VR controllers and some had as much as with the QR code tracking (Figures 9 and 11).

6. Limitations

A limitation of the QR code tracking approach is the fact that, as in any optical-based sensing system, if there were objects blocking the line of sight between the smartphone and the HMD, the QR code would not be readable and tracking would be lost. This is a common limitation for many of the related research we have mentioned, and even optical trackers of actual VR controllers must remain visible to the base stations or optical sensors. For instance, tracking would be lost if a finger or a hand was placed between the phone's camera and the QR code in the TrackCap system [4]. When placing the QR code, we made sure it is highly visible and large, and left room on the sides to allow users to hold the phone and tap the screen for interaction. In this way, the tapping operation does not interfere with the phone tracking. A different proposal to address this issue is presented in the PAIR system, where an array of smaller QR codes is placed around the screen of the smartphone, ensuring one of the codes can be tracked at any time. [5].

A limitation of the Gaze-based mode is that users are required to center their view in their selection by actively positioning their head accordingly, whereas in the QR code tracking system users have the possibility to keep their head positioning stable or semi-static while moving the phone in front of them to displace the pointer to the desired target position.

A final limitation is that users are not able to directly see the smartphone screen inside the VR. This could be problematic under some circumstances, such as when the user is trying to pick the phone from a table or a desk. To counter this, some HMD's allow the rotation of the HMD towards the top, without removing it. An alternative could be to use an Augmented Reality HMD, such as the Meta Quest 3 or the Apple Vision Pro, provided the corresponding SDK gives developers access to the built-in front camera video stream, or to use Near-Range Augmented Virtuality, as proposed by Alaee et al. [1], to allow VR users to see the smartphone and other items in close proximity.

7. Future Work

Several researchers have attempted to combine AR and Smartphones. We believe that SmartVR pointer would be ideal for see-through AR HMD's. When wearing a see-through HMD, such as Microsoft's HoloLens or the Magic Leap, a user would be able to see the smartphone directly and use the virtual pointer to select objects within the Augmented Reality environment, similar to what is shown in Figure 3 and in the supporting video. The viability of such an approach has been shown by Unlu et al., who used the HoloLens' front camera in the PAIR system [5].

A proposed improvement for SmartVR is to use the built-in cameras in non-see-through HMDs, which would eliminate the need for an external camera as well as the possibility of the camera being accidentally detached or disconnected. A related challenge with some HMD's, such as the HTC Vice Cosmos, is that the built-in cameras are pointing towards the front, which forces users to either raise the phone higher than in the proposed setup, or to look down to find the smartphone. More recently, some HMD's such as the Valve Index and the Meta Quest 3, have built-in pass-through cameras that capture users hands and surroundings without forcing users to raise their hands in front of them. These pass-through cameras use ultra-wide-angle lenses, also known as fish-eye lenses, which cause large perspective distortions in the captured frames. These distortions make it very challenging for the Vuforia engine to track the QR code, as the plugin is designed to process images from cameras with regular lenses and views. An optical tracking system designed to compensate for such wide-angle lens distortions could be used to track the QR code in these HMD's. A potential problem with this approach is that currently, not all HMD manufacturers provide game engine developers with SDK access to the video streams from the pass-through cameras.

 doi:10.20944/preprints202405.1815.v1

Finally, it is worthwhile noting that the Vuforia plugin provides a measure of the tracked target's depth, so that it could be possible to assess the approximate distance from the user's viewpoint to the phone. When the user moves closer to or away from the camera, a 3D pointer could use this depth information. Depending on the application, it might be useful to access the depth information for spatial tasks such as 3D sculpting. Such 3D positioning would require that the distances detected by the Vuforia plugin be mapped to distances in the Virtual Environment. In this work, we found that keeping the pointer on a 2D projection plane in front of the user provides a robust and intuitive solution for selection and interaction with virtual objects, and that the depth channel processing would introduce a confounding factor for the operations we have implemented.

We are exploring additional essential operations in VR and AR that can be performed as quickly and efficiently as when using a custom VR controller. We are thus interested in finding out whether more the functions afforded by current VR controllers can be replaced by smartphones and look forward to develop additional applications of this technique for Mixed and Augmented Reality.

8. Conclusions

We have presented SmartVR Pointer, a technique that allows the use of the smartphone, a device already in use by most people, for intuitive, one-handed tactile interaction in VR, that eliminates the need for specialized controllers for the demonstrated applications. The proposed approach offers the following advantages:

1. It allows for the use of the smartphone in Virtual, Mixed and Augmented Reality and supports the sense of touch for object selection, dragging and navigation.
2. It eliminates the need to learn how to use custom VR controllers for completing these tasks.
3. It works with a regular camera and is compatible with HMD's that already come with a front camera.
4. It is simple to setup and requires no calibration, synchronization, or complex hardware assembly or disassembly.

The approach we have described has the potential to make VR more accessible to the average consumer through a reduction both in the learning curve associated with using most VR controllers and in the need of specialized controllers for VR systems. Through the application demos, we found that SmartVR Pointer can be reliably used for navigation as well as for selection, positioning, and rotation of 3D objects within VR. This is promising, as many interactions in VR consists of some combination of these basic tasks.

Author Contributions: Conceptualization, Brianna McDonald, Qingyu Zhang and Oscar Meruvia-Pastor; Data curation, Brianna McDonald and Oscar Meruvia-Pastor; Formal analysis, Aiur Nanzatov, Lourdes Pena-Castillo and OscarConceptualization, Brianna McDonald, Qingyu Zhang and Oscar Meruvia-Pastor; Data curation, Aiur Nanzatov and Oscar Meruvia-Pastor; Formal analysis, Aiur Nanzatov, Lourdes Pena-Castillo and Oscar Meruvia-Pastor; Funding acquisition, Lourdes Pena-Castillo and Oscar Meruvia-Pastor; Investigation, Brianna McDonald, Qingyu Zhang, Aiur Nanzatov and Oscar Meruvia-Pastor; Methodology, Brianna McDonald, Qingyu Zhang and Oscar Meruvia-Pastor; Project administration, Oscar Meruvia-Pastor; Resources, Brianna McDonald, Qingyu Zhang, Aiur Nanzatov and Oscar Meruvia-Pastor; Software, Brianna McDonald, Qingyu Zhang and Aiur Nanzatov; Supervision, Oscar Meruvia-Pastor; Validation, Brianna McDonald, Qingyu Zhang, Aiur Nanzatov and Oscar Meruvia-Pastor; Visualization, Brianna McDonald, Qingyu Zhang, Aiur Nanzatov and Oscar Meruvia-Pastor; Writing – original draft, Brianna McDonald, Qingyu Zhang and Oscar Meruvia-Pastor; Writing – review & editing, Aiur Nanzatov, Lourdes Pena-Castillo and Oscar Meruvia-Pastor.

References

1. Alaee, G.; Deasi, A.P.; Pena-Castillo, L.; Brown, E.; Meruvia-Pastor, O. A User Study on Augmented Virtuality Using Depth Sensing Cameras for Near-Range Awareness in Immersive VR. Institute of Electrical and Electronics Engineers Inc., 2018.
2. Bai, H.; Zhang, L.; Yang, J.; Billinghurst, M. Bringing full-featured mobile phone interaction into virtual reality. *Computers and Graphics (Pergamon)* **2021**, *97*, 42–53. doi:10.1016/j.cag.2021.04.004.

 doi:10.20944/preprints202405.1815.v1

3. Desai, A.P.; Pena-Castillo, L.; Meruvia-Pastor, O. A Window to your Smartphone: Exploring Interaction and Communication in Immersive VR with Augmented Virtuality. 2017 Computer And Robot Vision (CRV) Conf. Proc.; , 2015.
4. Mohr, P.; Tatzgern, M.; Langlotz, T.; Lang, A.; Schmalstieg, D.; Kalkofen, D. TrackCap: Enabling smartphones for 3D interaction on mobile head-mounted displays. Proc. CHI; Association for Computing Machinery, Inc: NY, USA, 2019. doi:10.1145/3290605.3300815.
5. Unlu, A.E.; Xiao, R. PAIR: Phone as an Augmented Immersive Reality Controller. Proc. VRST; Association for Computing Machinery, Inc: NY, USA, 2021; pp. 1–6. doi:10.1145/3489849.3489878.
6. Zhang, L.; He, W.; Bai, H.; He, J.; Qiao, Y.; Billinghurst, M. A Hybrid 2D-3D Tangible Interface for Virtual Reality. Proc. SIGGRAPH; ACM: NY, USA, 2021. doi:10.1145/3450618.3469166.
7. Aseeri, S.A.; Acevedo-Feliz, D.; Schulze, J. Poster: Virtual reality interaction using mobile devices. 2013, pp. 127–128. https://doi.org/10.1109/3DUI.2013.6550211.
8. Budhiraja, R.; Lee, G.A.; Billinghurst, M. Interaction techniques for HMD-HHD hybrid AR systems. 2013, pp. 243–244. https://doi.org/10.1109/ISMAR.2013.6671786.
9. Hincapié-Ramos, J.D.; Özacar, K.; Irani, P.P.; Kitamura, Y. GyroWand: IMU-based Raycasting for Augmented Reality Head-Mounted Displays. Association for Computing Machinery, Inc, 2015, pp. 89–98. https://doi.org/10.1145/2788940.2788947.
10. Kharlamov, D.; Woodard, B.; Tahai, L.; Krzysztof, P. TickTockRay: Smartwatch-based 3D pointing for smartphone-based virtual reality. Proc. VRST; Association for Computing Machinery, Inc: NY, USA, 2016; pp. 363–364. doi:10.1145/2993369.2996311.
11. Kim, H.I.; Woo, W. Smartwatch-assisted robust 6-DOF hand tracker for object manipulation in HMD-based augmented reality. Institute of Electrical and Electronics Engineers Inc., 2016, pp. 251–252. https://doi.org/10.1109/3DUI.2016.7460065.
12. Young, T.S.; Teather, R.J.; Mackenzie, I.S. An arm-mounted inertial controller for 6DOF input: Design and evaluation. Institute of Electrical and Electronics Engineers Inc., 2017, pp. 26–35. https://doi.org/10.1109/3DUI.2017.7893314.
13. Hirzle, T.; Gugenheimer, J.; Rixen, J.; Rukzio, E. WatchVR: Exploring the Usage of a Smartwatch for Interaction in Mobile Virtual Reality. Association for Computing Machinery, 2018. https://doi.org/10.1145/3170427.3188629.
14. Park, K.B.; Lee, J.Y. New design and comparative analysis of smartwatch metaphor-based hand gestures for 3D navigation in mobile virtual reality. *Multimedia Tools and Applications* **2019**, *78*, 6211–6231. https://doi.org/10.1007/s11042-018-6403-9.
15. Hattori, K.; Hirai, T. Inside-out Tracking Controller for VR/AR HMD using Image Recognition with Smartphones. Association for Computing Machinery, 2020. https://doi.org/10.1145/3388770.3407430.
16. Pietroszek, K.; Kuzminykh, A.; Wallace, J.R.; Lank, E. Smartcasting: A discount 3D interaction technique for public displays. Proc. OzCHI; Association for Computing Machinery, Inc: NY, USA, 2014; pp. 119–128. doi:10.1145/2686612.2686629.
17. Pietroszek, K.; Tahai, L.; Wallace, J.R.; Lank, E. Watchcasting: Freehand 3D interaction with off-the-shelf smartwatch. Institute of Electrical and Electronics Engineers Inc., 2017, pp. 172–175. https://doi.org/10.1109/3DUI.2017.7893335.
18. Siddhpuria, S.; Malacria, S.; Nancel, M.; Lank, E. Pointing at a Distance with Everyday Smart Devices. Proc. CHI; Association for Computing Machinery, Inc: NY, USA, 2018; pp. 1–11. doi:10.1145/3173574.3173747.
19. HTC. VIVE Flow. https://www.vive.com/ca/product/vive-flow/overview/.
20. SmartVR Pointer: Using Smartphones and Gaze Orientation for Selection and Navigation in Virtual Reality - Project Download. https://drive.google.com/drive/folders/1aZy3rgfaQnb1dKR1UosWUP0r5ryAKLiw?usp=sharing.
21. PTC Vuforia. Vuforia Engine. https://developer.vuforia.com/downloads/sdk.

 doi:10.20944/preprints202405.1815.v1

22. Zhu, F.; Sousa, M.; Sidenmark, L.; Grossman, T. PhoneInVR: An Evaluation of Spatial Anchoring and Interaction Techniques for Smartphone Usage in Virtual Reality. Proceedings of the CHI Conference on Human Factors in Computing Systems; Association for Computing Machinery: New York, NY, USA, 2024; CHI '24. doi:10.1145/3613904.3642582.
23. Blattgerste, J.; Renner, P.; Pfeiffer, T. Advantages of Eye-Gaze over Head-Gaze-Based Selection in Virtual and Augmented Reality under Varying Field of Views. Proceedings of the Workshop on Communication by Gaze Interaction; Association for Computing Machinery: New York, NY, USA, 2018; COGAIN '18. 0.1145/3206343.3206349, doi:10.1145/3206343.3206349.