Smartphone-Based Input as a Practical Alternative for Object Manipulation in HMD-Based Immersive Environments

Yotaro Sasaki[1], Yoshitsugu Manabe[1] and Noriko Yata[1]

[1]Chiba University, Chiba, Japan

Abstract

In virtual and mixed reality environments using head-mounted displays (HMDs), users often rely on dedicated controllers or hand tracking to select and manipulate virtual objects. Dedicated controllers provide stable tracking, physical buttons, and vibrotactile feedback, but they require users to carry an additional device. Hand tracking improves portability, but it can be less stable or more physically demanding for precise and repeated manipulation. This study investigates a commodity smartphone as a practical auxiliary input device for HMD-based object manipulation. We implemented two smartphone-based methods with contrasting interaction styles: Gesture, a discrete command-oriented method using touchscreen strokes for object selection and mode switching, and Pointer, a continuous pointer-oriented method using touch input to control a cursor in the HMD view and manipulate objects. We compared these methods with a Meta Quest Touch Plus controller and hand tracking in four tasks: Selection, Translation, Rotation, and Docking. The results showed that the dedicated controller retained a clear time-efficiency advantage in complex continuous manipulation tasks, whereas its advantage was less uniform across all interaction stages. Importantly, the smartphone methods demonstrated distinct, task-dependent potential: Gesture achieved object selection performance comparable to the controller, and Pointer effectively supported continuous rotation while achieving an overall subjective rating statistically comparable to the dedicated controller ($p_{holm} = .1875$). Furthermore, despite requiring longer completion times due to explicit mode-switching structures, both smartphone methods successfully reached final positioning and rotation accuracies close to the controller across several conditions, significantly outperforming bare-hand tracking in terms of workload and alignment stability. These findings suggest that smartphones should not be regarded as universal replacements for dedicated controllers, but as context-dependent, portable auxiliary surfaces that can effectively complement device-free input and support precise, reliable interaction in future MR and mobile AR-glasses environments.

Keywords

Virtual Reality, Mixed Reality, Head-Mounted Display, Smartphone Interaction, Input Device, Object Manipulation

1. Introduction

In VR and MR environments using HMDs, users frequently need to select, move, rotate, and align virtual objects in three-dimensional space. Current VR systems commonly use dedicated controllers because they provide stable 6-DoF tracking, physical buttons, triggers, and vibrotactile feedback. These properties make controllers effective for precise and repeated object manipulation. However, in everyday MR or AR-glasses scenarios, carrying and managing a dedicated controller may become a burden, especially when the interaction is occasional, mobile, or embedded in daily activities.

Device-free input methods such as hand tracking, gaze input, and voice input reduce this burden and improve portability. At the same time, they can be limited for continuous precise manipulation. Hand tracking can require users to maintain gestures such as pinches or grasp-like postures, gaze input is not always suitable for explicit manipulation, and voice input is sensitive to social and environmental context. Therefore, future HMD interfaces may benefit from an input option that is more portable than a dedicated controller while still providing explicit input, tactile feedback, and precise finger-based control.

A smartphone is a practical candidate for this role. Many users already own and carry a smartphone, which reduces the need to purchase or carry an additional device. Smartphones also provide high-

APMAR'26: The 18th Asia-Pacific Workshop on Mixed and Augmented Reality, Aug. 3–5, 2026, Taipei, Taiwan

25wm0225@student.gs.chiba-u.jp (Y. Sasaki)

resolution touchscreens, stroke and multi-touch input, inertial sensing, wireless communication, and vibrotactile feedback. These properties make smartphones different from both dedicated controllers and device-free input. They do not provide the same physically tracked 6-DoF interaction as a controller, but they offer a familiar touchscreen surface that can support discrete commands, continuous pointing, mode switching, and explicit confirmation.

This study evaluates whether these characteristics can be used for HMD-based object manipulation. Although the experiment is conducted in VR to maintain controlled and repeatable task conditions, the target problem is broader HMD-based spatial interaction, including future MR and AR-glasses scenarios where lightweight and readily available input devices may be important.

We compare four input methods under the same task conditions: a dedicated controller, hand tracking, smartphone Gesture input, and smartphone Pointer input. The two smartphone methods were designed to cover two contrasting directions of smartphone interaction. Gesture is a discrete, command-based method that uses touchscreen strokes to select objects and switch operation modes. Pointer is a continuous, pointer-based method that uses touch input to control a cursor in the HMD view and manipulate objects. By comparing these methods with both a controller and hand tracking, we examine not only whether a smartphone can substitute for common HMD input methods, but also which stages of object manipulation are suitable for each smartphone interaction style.

The aim of this study is not to demonstrate that smartphones universally outperform dedicated controllers. Rather, the study clarifies the practical role of smartphones in HMD interaction. In particular, it examines whether smartphones can complement device-free inputs such as hand tracking and gaze input, and whether touchscreen-centered interaction can provide advantages that differ from those of dedicated controllers.

The contributions of this work are as follows: (1) the implementation of two smartphone-based HMD input methods with contrasting characteristics, namely a discrete command-oriented method and a continuous pointer-oriented method; (2) a unified comparison with both a dedicated controller and hand tracking; (3) an evaluation using object-manipulation tasks consisting of Selection, Translation, Rotation, and Docking; and (4) a discussion of the substitutability, limitations, and suitable application range of smartphone input for future HMD, MR, and AR-glasses interaction.

2. Related Work

Smartphones and handheld devices have been studied as input devices for AR and VR environments. These studies show that smartphones can be used not only as display devices, but also as input media that combine touch input, gestures, device motion, sensing, and tactile feedback.

Darbar et al. investigated smartphone-enabled text selection in AR-HMDs and compared techniques including continuous touch, discrete touch, spatial movement, and raycasting [1]. Knierim et al. presented the SmARtphone Controller, which uses a smartphone as an input and output modality for mobile AR environments and evaluates speed, accuracy, and workload in object manipulation and GUI interaction tasks [2]. These studies demonstrate the usefulness of smartphones as input devices for AR and HMD interaction. However, they do not directly examine how smartphone input compares with both contemporary HMD controllers and hand tracking across multiple components of object manipulation.

Smartphone input has also been explored as a possible alternative to specialized controllers in VR. SmartVR Pointer uses smartphones and gaze orientation for selection and navigation in virtual reality and explicitly addresses the possibility of replacing specialized VR controllers [3]. The present study shares this motivation, but focuses on object manipulation beyond selection and navigation, including translation, rotation, and docking, and evaluates smartphone input against both controller-based and hand-tracking baselines.

Touch, gesture, and pointing techniques using smartphones are also relevant to this study. Touch-Move-Release compared surface and motion gestures for mobile AR and proposed a gesture pattern based on touching, moving the device, and releasing [4]. Plane-Casting proposed smartphone-based

3D cursor control for distant object manipulation and provides background for spatial pointing with smartphones [5]. Waldow et al. compared touch-based smartphone input with gesture-based input for constrained object manipulation in AR [6]. These works motivate our comparison between two contrasting smartphone interaction styles: a discrete gesture-command method and a continuous pointer-control method.

From the viewpoint of 3D object manipulation, techniques such as ray casting, virtual-hand interaction, arm-extension methods, and hybrid approaches have long been studied. The Go-Go technique and the remote-object manipulation study by Bowman and Hodges provide background for combining ray-based selection with pose-based manipulation [7, 8]. In this study, these 3D interaction techniques are not treated as the main research target, but as a basis for designing comparable baseline methods and task conditions.

Prior work has therefore established that smartphones can be useful in AR/HMD input, VR controller replacement, touch and gesture input, 3D cursor control, and constrained object manipulation. The present study extends this line of work by asking a more specific practical question: under a unified set of object-manipulation tasks, when can a smartphone-based method approximate the performance of a controller, when is it closer to hand tracking, and when does it introduce additional cost? To answer this question, we implement two contrasting smartphone methods and compare them with both a Meta Quest Touch Plus controller and hand tracking in Selection, Translation, Rotation, and Docking tasks. This comparison clarifies not only whether smartphones can serve as alternative input devices, but also which stages of manipulation benefit from touchscreen-based discrete commands or continuous pointer control.

3. System and Input Methods

3.1. System Overview

The input methods are designed to make the four conditions comparable while preserving the characteristic affordances of each device. The controller and hand-tracking conditions use ray-based target acquisition and pose-based manipulation, which are common in HMD interaction [8, 7, 9]. The smartphone conditions apply touch input, stroke input, device orientation, and vibration feedback to HMD object manipulation, building on prior work on smartphone-based AR/VR interaction [1, 2, 3, 4, 5, 6].

The system consists of a VR application running on the HMD side and a smartphone application running on the input-device side. The HMD application manages the virtual scene, objects, selection state, pointer visualization, operation state, task progression, and logging. Smartphone input is transmitted to the HMD over a local Wi-Fi network using UDP. The HMD interprets the received events and performs selection, translation, rotation, and deselection. HMD-side events are also sent back to the smartphone and presented as vibrotactile feedback.

3.2. Input Methods

Dedicated controller method. The dedicated controller method emits a ray from a Meta Quest Touch Plus controller and displays a cursor at the hit position when the ray intersects a manipulable object. In Selection, the user selects an object with the index trigger. In Translation and Docking, holding the index trigger starts movement. At the start of movement, the system stores the distance from the controller ray origin to the hit point and the local offset between the hit point and the selected object. During movement, the object is positioned so that this local offset follows a point on the controller ray. Lateral movement is produced by changing the controller position or ray direction, while depth movement is controlled by moving the controller forward or backward along the initial ray direction. This depth displacement is amplified by a depth gain of 5.0. In Rotation and Docking, squeezing the hand trigger starts a grab-like rotation operation, and the relative controller orientation from the start of the operation is applied to the object. Controller vibration is presented for events such as selection, operation start, and task success.

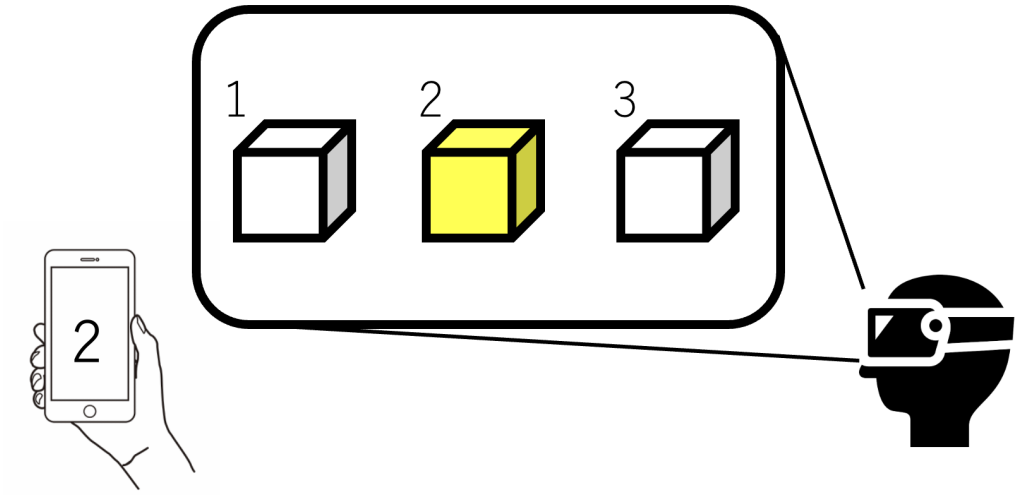

Figure 1: Schematic illustration of the gesture-command method. Users draw number gestures on the smartphone touchscreen to select objects, while predefined command gestures switch manipulation modes. The selected object can then be translated or rotated using smartphone and hand movements.

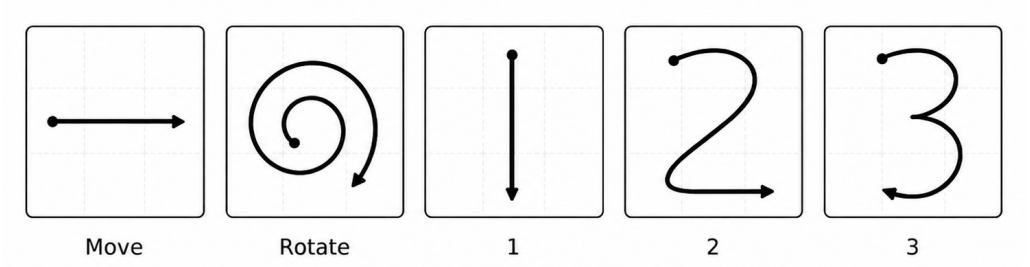

Figure 2: Gesture set used in the smartphone gesture-command method. The set includes command gestures for movement and rotation, and numeric gestures for selecting labeled objects. The gestures were selected based on a preliminary elicitation study with 16 participants.

Hand-tracking method. The hand-tracking method uses a ray from the right hand to acquire targets and displays a cursor at the hit position. In Selection, the user selects an object with an index-finger pinch. In Translation and Docking, holding the pinch starts movement. At the start of movement, the system stores the distance from the hand-ray origin to the hit point and the local offset between the hit point and the selected object. During movement, the object follows a point on the hand ray while preserving this local offset. Lateral movement is produced by changing the hand position or ray direction, while depth movement is controlled by moving the hand forward or backward along the initial ray direction. This depth displacement is amplified by a depth gain of 5.0. In Rotation, a fist strength is computed from the pinch strengths of multiple fingers, and a grasp-like hand posture is used to start rotation. During rotation, the relative change in hand orientation from the start of the operation is applied to the object.

Smartphone gesture-command method. Figure 1 presents a schematic illustration of the interaction flow in the gesture-command method. The gesture set was determined through a preliminary elicitation study with 16 participants by selecting gestures that were frequently proposed and easy to distinguish. At runtime, command recognition treats each input stroke as a two-dimensional touch-coordinate sequence. The sequence is resampled to 64 points based on cumulative path length and normalized while preserving its aspect ratio. The normalized sequence is then classified into a command label using a one-layer LSTM classifier with 64 hidden units.

The HMD manages three states, *Idle*, *Selected*, and *Operating*, and operation types such as *Move* and *Rotate*. Each object has a visible number label, and participants select an object by drawing the

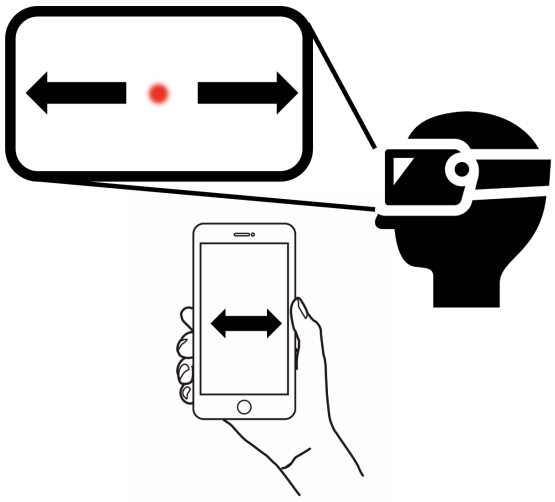

Figure 3: Schematic illustration of the pointer method. Smartphone touch input controls a cursor for object selection, after which the selected object can be translated and rotated.

corresponding number on the smartphone. After selection, predefined gestures start movement or rotation. Movement maps touch-position changes to the camera's right and up directions. To control movement along the depth axis, the system tracks the user's right hand and maps its displacement in the camera's depth direction to object movement. This function is active only while the user is touching the smartphone screen. A depth gain of 5.0 is applied to the detected displacement. For rotation, the user keeps touching the smartphone screen and tilts the device, which acts like a clutch for starting and maintaining rotation. The relative change in device orientation is then applied to the object. A double tap cancels the current operation or deselects the object.

Smartphone pointer method. The pointer method maps touch movement on the smartphone to viewport coordinates of the HMD camera and continuously generates a camera ray from the corresponding viewport position to determine the cursor location in the HMD view. Figure 3 illustrates the interaction flow and mapping between smartphone inputs and object manipulations in the pointer method. The cursor is rendered in 3D space: when the ray intersects a selectable object, it is displayed on that object; otherwise, it is displayed on the background side of the scene. Its scale is adjusted according to distance so that its apparent size remains roughly constant from the user's viewpoint. A tap selects the object currently intersected by the ray. After selection, the user drags on the smartphone screen to move the object. Movement maps touch displacement to the camera's right and up directions. Depth movement is enabled only while the user is touching the smartphone screen and is controlled using the detected right-hand displacement along the depth direction. This depth displacement is amplified by a depth gain of 5.0. For rotation, the user performs a double tap and keeps the second touch on the screen, then tilts the smartphone while holding the touch. The relative change in smartphone orientation is then applied to the object.

Common features of the smartphone methods. In both of the smartphone conditions, users hold the device in their right hand and operate the touchscreen using their right thumb. Vibrotactile feedback is presented for events such as *Selected*, *Deselected*, *MoveStarted*, *RotateStarted*, and *PoseMatched*. This helps users understand state changes without continuously looking at the smartphone while wearing the HMD.

4. Experiment

4.1. Participants

Twelve participants took part in the experiment. Their ages ranged from 21 to 25 years, and all participants were right-handed. Their prior VR experience varied, ranging from participants with almost

no VR experience to participants who used VR regularly. The entire experiment took approximately one hour, including questionnaire responses and breaks.

4.2. Design

We conducted a within-subject experiment comparing four input conditions: **Hand**, **Controller**, **Pointer**, and **Gesture**. The experiment used a Meta Quest 3S HMD, a Meta Quest Touch Plus controller, and an iPhone 13 smartphone [10, 11, 12]. The detailed interaction design of each condition is described in the previous section.

Each participant experienced all four methods. Method order was counterbalanced using a Latin-square design with four order patterns: Hand–Controller–Pointer–Gesture, Controller–Pointer–Gesture–Hand, Pointer–Gesture–Hand–Controller, and Gesture–Hand–Controller–Pointer. Task order was fixed as Selection, Translation, Rotation, and Docking because each subsequent task was designed to incorporate the operations required in the preceding tasks.

For each method, participants first viewed an instruction screen and then performed task units for Selection, Translation, Rotation, and Docking. For each method, participants completed one practice trial for each of the four tasks before the main trials. The practice trials were configured so that participants could experience the main operations required by each method, including target selection, depth control, multi-axis translation, and multi-axis rotation, before the main trials.

The main trials used three conditions for each task, labeled Easy, Medium, and Hard. These labels indicate differences in movement amount, object size, or the number of controlled axes, and do not necessarily assume a strictly monotonic increase in subjective or empirical difficulty. Unless otherwise noted, the x-, y-, and z-axes refer to the Unity world coordinate axes used in the experimental scene. Trial conditions were generated from the task scenario asset. For each unit, the system extracted only the trials matching the current task type, and the order of main trials was shuffled using a seed derived from participant number, method order, and task type.

4.3. Tasks

The experiment used four tasks: Selection, Translation, Rotation, and Docking. These tasks were chosen to evaluate major components of HMD-based object manipulation: target acquisition, position control, orientation control, and compound manipulation that includes pose matching and deselection. In the manipulation tasks other than Selection, both the target object and the manipulable object were cubes with a side length of 0.80 m. Participants manipulated an opaque cube and matched it to a target cube shown as a wireframe, as illustrated in Figure 4. In Translation and Rotation, the target cube was placed 4.0 m in front of the participant (viewpoint-relative position: (0, 0, 4)), and the manipulable cube was initially placed with a position or orientation offset according to the task condition. Figure 5 shows an example experimental scene during object manipulation.

In the Selection task, three selectable cubes were placed around a point 4 m in front of the participant's initial viewpoint. In the implementation, the three objects were arranged on a circle with a radius of 1 m around this center point, forming a triangular layout from the participant's viewpoint. Each cube had a visible number label, and participants selected the object corresponding to the target number shown in the instruction. The Easy, Medium, and Hard conditions changed the cube size: 0.20 m, 0.12 m, and 0.07 m on a side, respectively. These sizes were not linearly spaced in physical size. Instead, the non-linear intervals were chosen to reflect the fact that target-acquisition difficulty changes approximately according to the logarithmic relationship between movement amplitude and target width [13]. A trial was successful when the selected object ID matched the target ID.

In the Translation task, participants selected the manipulable cube and moved it to the target position. The translation amount was defined as the initial offset between the manipulable cube and the target cube, using a Manhattan distance, that is, the sum of absolute displacements along the Unity coordinate axes, of 1.5 m. Easy was a one-axis condition with a 1.5 m displacement along the x-axis. Medium was a two-axis condition with 0.75 m displacement along both the x- and y-axes. Hard was a three-axis

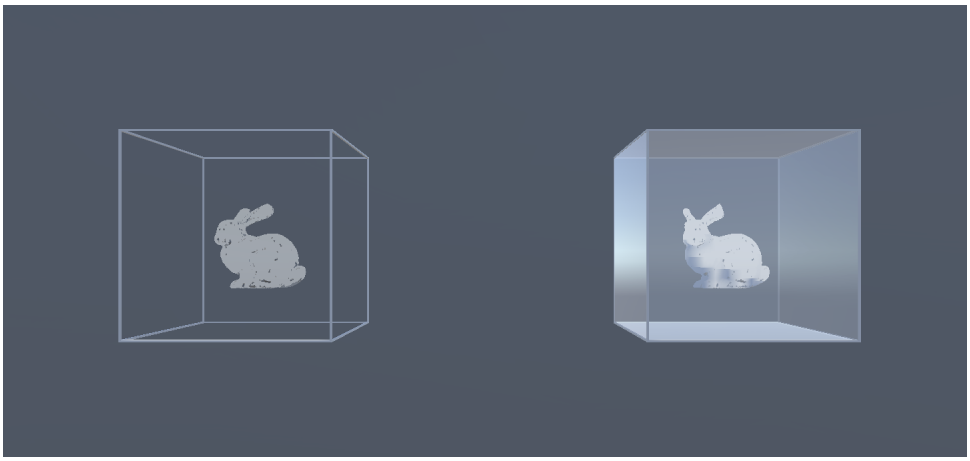

Figure 4: Objects used in the manipulation tasks. The left cube is the wireframe target object, and the right cube is the opaque manipulable object. Participants moved or rotated the manipulable object to match the target object's position and/or orientation.

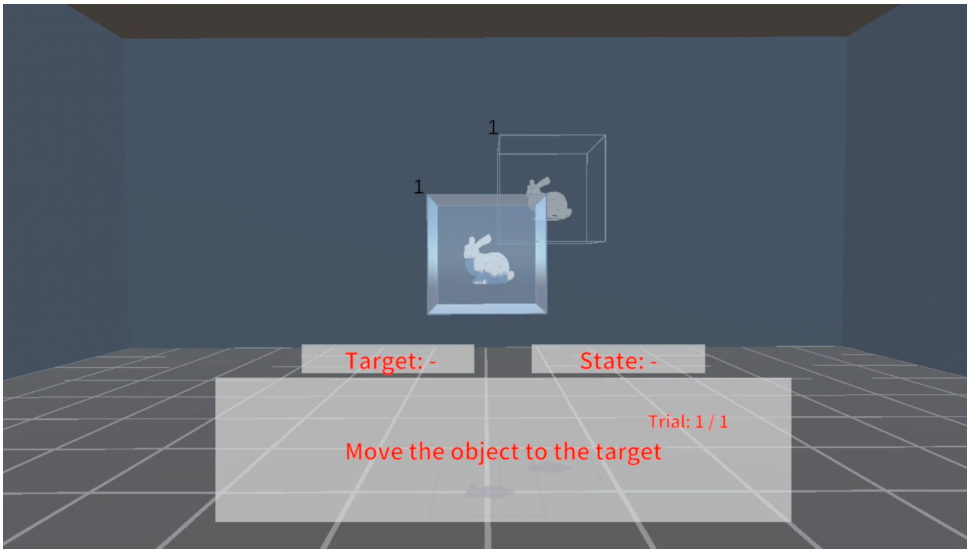

Figure 5: Example experimental scene during the object-manipulation experiment.

condition with 0.5 m displacement along each of the x-, y-, and z-axes. This condition design reflects the smartphone methods, in which movement along the x/y directions is controlled by two-dimensional touch input on the screen, whereas movement along the z direction is controlled by hand displacement in depth.

In the Rotation task, participants rotated the manipulable cube to match the orientation of the semi-transparent target cube. The target cube was placed at the viewpoint-relative position (0, 0, 4), and the manipulable cube was initially presented at the same position with an orientation offset. The total rotation amount was 60 degrees. Easy was a one-axis condition with a 60-degree roll rotation. Medium was a two-axis condition with 30 degrees of rotation around both roll and pitch. Hard was a three-axis condition with 20 degrees of rotation around roll, pitch, and yaw. This axis configuration was designed to gradually increase the number of controlled rotational axes while keeping the total nominal rotation amount constant. The choice of axes was also informed by practical differences in wrist and forearm motion during hand- or device-based rotation. For example, general range-of-motion references indicate that forearm pronation/supination and wrist flexion/extension allow larger motion ranges than radial/ulnar deviation [14].

In the Docking task, participants matched both position and orientation in three sequential steps. In

each trial, three pairs of opaque manipulable cubes and semi-transparent target cubes were arranged approximately 4 m in front of the participant and numbered 1, 2, and 3 from left to right. Participants selected each object in numerical order and matched it to the position and orientation of the corresponding target. After successfully matching each pair, participants had to deselect the object before proceeding to the next pair. This design was intended to evaluate not only pose matching, but also the complete operation process consisting of selection, alignment, and deselection.

Success criteria were defined for each task type. In Translation, a trial was successful when the position error between the manipulated object and the target was no greater than 0.08 m. In Rotation, a trial was successful when the angular error was no greater than 10 degrees. In Docking, both criteria had to be satisfied simultaneously. The success state had to be maintained for 0.5 s before a trial or Docking step was considered successful. After a normal trial, selection and operation states were reset, and the next trial began after a 1.0 s inter-trial interval. In Docking, a step was completed only after the participant deselected the successfully aligned object.

4.4. Procedure

At the beginning of each method, the HMD displayed an instruction screen explaining the current method. After the participant gave a confirmation input, the system moved to a waiting screen for the next task unit. Each task unit was also started by a confirmation input. After all trials in a unit were completed, the system advanced to the next unit. When all units for a method were completed, a method-completion screen was displayed, and the next method started after participant confirmation. After each method, participants answered a questionnaire and took breaks when needed. The experiment ended after all four methods had been completed.

At the start of each trial, selection state, pointer state, operation state, and temporary objects for sequential Docking were reset. Selection trials displayed three selectable objects, while Translation, Rotation, and Docking trials displayed the manipulable object and target object. During a trial, the HMD presented the task type, trial number, task instruction, and runtime feedback such as position and rotation errors. When a trial or Docking step was successful, the system displayed the success state in the instruction or result text area and changed the target cube color to green as visual feedback.

4.5. Measures

The system recorded trial-level information including participant ID, input method, method order, task type, condition label, trial order, sequential-Docking step index, shuffle seed, target ID, start position, target position, target rotation, and a sign-coded representation of the movement or rotation condition.

Objective measures included total task completion time, time from task start to first selection, and time from first selection to success. For sequential Docking, the system also recorded the time from success to deselection. Accuracy measures included final position error, final rotation error, minimum position error during the trial, and minimum rotation error during the trial. Operation-count measures included the numbers of selection, movement, and rotation actions.

Subjective measures were collected using a seven-point Likert-scale questionnaire after each method. In this paper, we analyze physical load, mental load, task-specific ease of use, helpfulness of vibrotactile feedback, and overall evaluation. Workload-related items were coded so that higher scores represented lower perceived load, making higher values consistently indicate more positive ratings. Task-specific ease of use was rated separately for Selection, Translation, Rotation, and Docking.

5. Results

We analyzed task completion time, final accuracy, and subjective ratings. Because all four methods were tested by the same participants, we used Friedman tests for omnibus comparisons across methods. When the omnibus test was significant, pairwise Wilcoxon signed-rank tests with Holm correction were used for post-hoc comparisons. Unless otherwise stated, values are reported as medians.

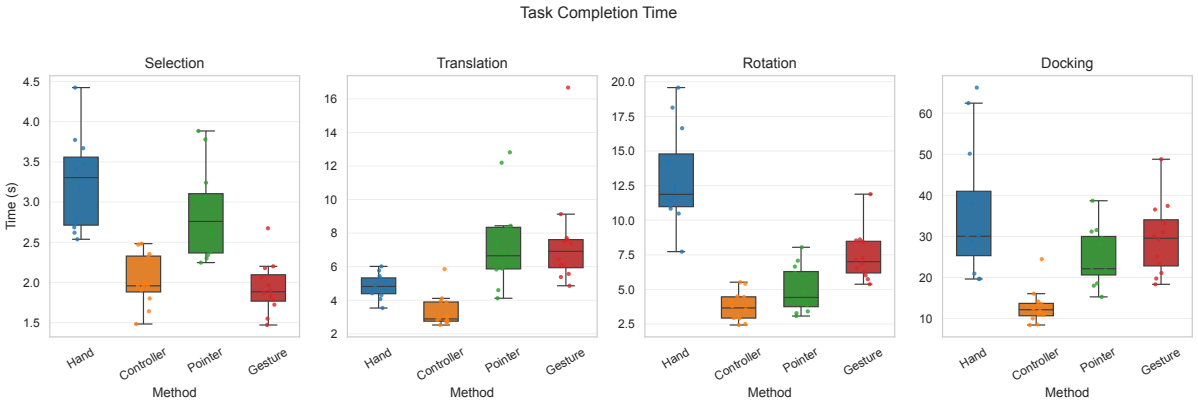

Figure 6: Task completion time for Selection, Translation, Rotation, and Docking. The tasks are plotted in separate panels to account for differences in completion-time scales across tasks. Lower values indicate faster performance.

5.1. Task Completion Time

Task completion time differed significantly among methods for all four tasks: Selection ($\chi^2(3) = 29.30$, $p < .001$, Kendall's $W = .814$), Translation ($\chi^2(3) = 23.70$, $p < .001$, $W = .658$), Rotation ($\chi^2(3) = 28.70$, $p < .001$, $W = .797$), and Docking ($\chi^2(3) = 23.70$, $p < .001$, $W = .658$).

In Selection, Gesture was the fastest method, with a median completion time of 1.88 s, followed closely by Controller at 1.96 s. Pointer and Hand were slower, with medians of 2.76 s and 3.30 s, respectively. Post-hoc comparisons showed that Gesture was significantly faster than Hand and Pointer, and Controller was significantly faster than Hand and Pointer. There was no significant difference between Gesture and Controller. These results indicate that the label-based discrete selection used in Gesture can achieve selection performance comparable to the dedicated controller, likely because it allows users to specify the target directly rather than precisely aligning a spatial cursor.

In Translation, Controller was the fastest method, with a median completion time of 2.89 s. Hand followed at 4.82 s, while Pointer and Gesture were slower, with medians of 6.65 s and 6.91 s. Controller was significantly faster than both smartphone methods, and Hand was also significantly faster than Pointer and Gesture. This difference is likely attributable not only to the additional cost of depth control in the smartphone methods, but also to their staged interaction structure, in which selection and movement are separated more explicitly. In Controller and Hand, users can start translation immediately after acquiring the target by holding the trigger or pinch. In contrast, Pointer and Gesture require users to transition from selection to a movement phase through dragging or gesture commands, and this select-to-manipulate transition may have increased completion time.

In Rotation, Controller was again the fastest method, with a median completion time of 3.66 s. Pointer was next at 4.42 s, followed by Gesture at 7.00 s and Hand at 11.87 s. Post-hoc tests showed that Hand was significantly slower than all other methods, Gesture was significantly slower than Controller and Pointer, and Controller and Pointer did not differ significantly. This result is important because it shows that smartphone orientation input, especially in the Pointer condition, can support rotation more effectively than maintaining bare-hand grasp-like gestures.

In Docking, Controller showed the clearest advantage, with a median completion time of 12.15 s. Pointer followed at 22.13 s, while Gesture and Hand were slower, with medians of 29.56 s and 30.03 s. Controller was significantly faster than Hand, Pointer, and Gesture. Pointer was significantly faster than Hand, but the difference between Pointer and Gesture was not significant. Docking required repeated selection, alignment, and deselection across three object-target pairs, so the advantage of Controller appears to come from its stable tracking and clear physical state transitions through trigger and grip input.

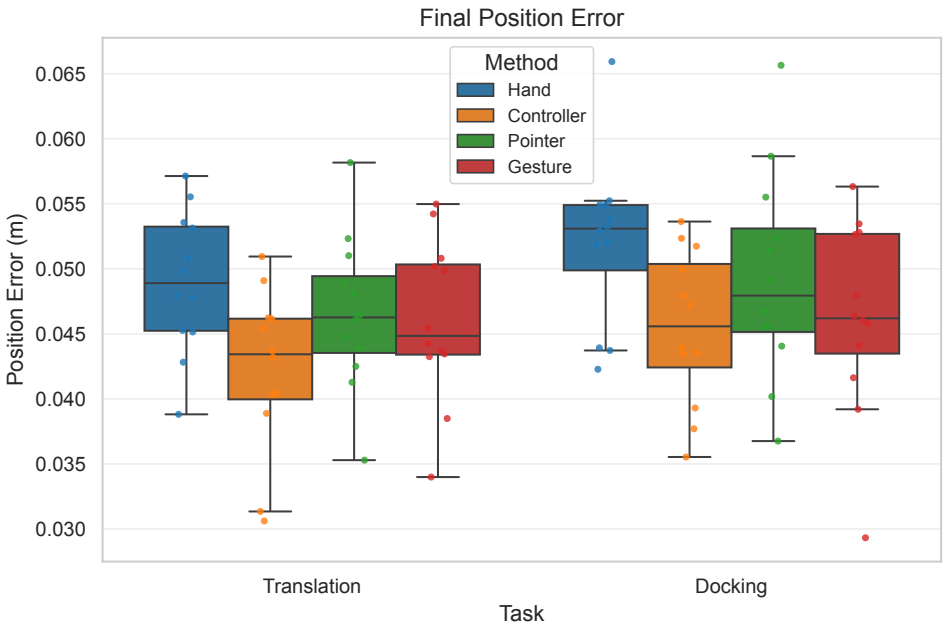

Figure 7: Final position error for Translation and Docking. Lower values indicate higher positional accuracy.

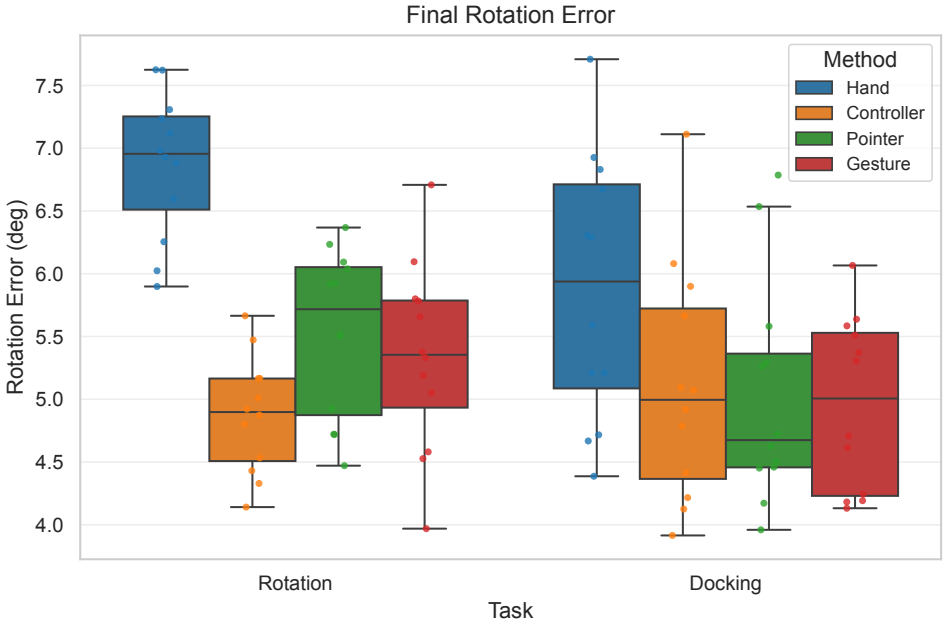

Figure 8: Final rotation error for Rotation and Docking. Lower values indicate higher rotational accuracy.

5.2. Accuracy

For final position error, the method effect was not significant in Translation ($\chi^2(3) = 5.00$, $p = .172$, $W = .139$). Median errors were 0.043 m for Controller, 0.045 m for Gesture, 0.046 m for Pointer, and 0.049 m for Hand. Thus, although Controller showed the smallest median error, the methods reached broadly comparable final positional accuracy in the simple Translation task.

In Docking, final position error differed significantly among methods ($\chi^2(3) = 10.30$, $p = .016$, $W = .286$). The only significant post-hoc difference was between Hand and Controller. Controller had a median error of 0.046 m, compared with 0.053 m for Hand. Pointer and Gesture had medians of 0.048 m and 0.046 m, respectively, and did not differ significantly from Controller. This suggests that once

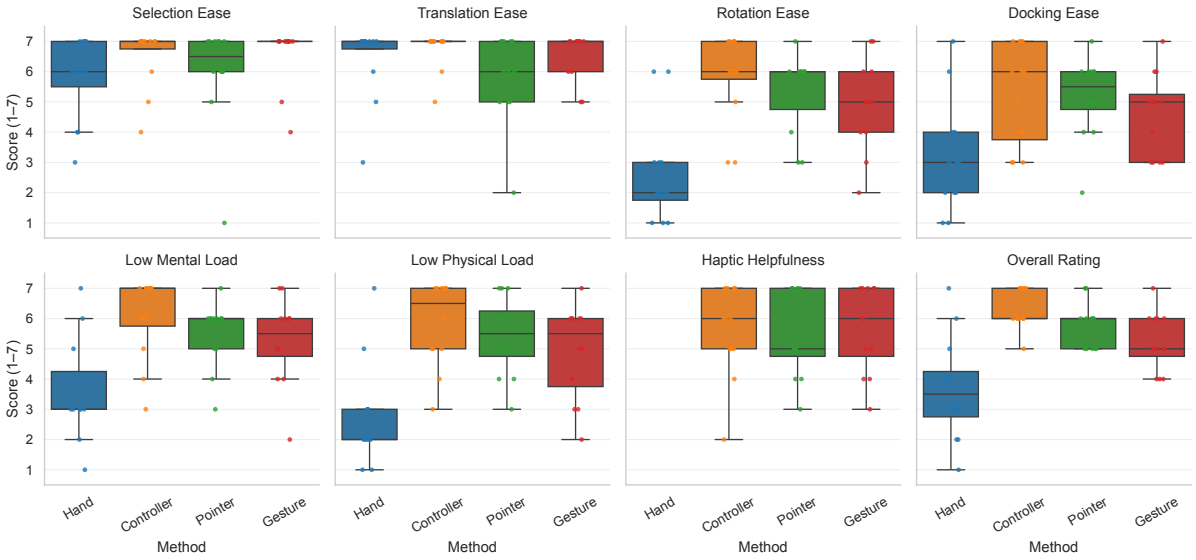

Figure 9: Subjective ratings on a seven-point scale. Higher values indicate more positive ratings, including the workload items, where higher scores indicate lower perceived workload.

participants reached the target, the smartphone methods could achieve final positional accuracy close to the controller, even though they required more time.

For final rotation error in Rotation, there was a significant method effect ($\chi^2(3) = 23.40$, $p < .001$, $W = .650$). Controller produced the smallest median error at 4.90 degrees. Gesture and Pointer followed at 5.35 and 5.72 degrees, while Hand had the largest error at 6.96 degrees. Post-hoc tests showed that Hand had significantly larger errors than Controller, Pointer, and Gesture, and Pointer had significantly larger errors than Controller. The smartphone methods therefore did not match the controller's rotational accuracy, but they were significantly more accurate than Hand in the Rotation task.

For final rotation error in Docking, the method effect was also significant ($\chi^2(3) = 10.00$, $p = .019$, $W = .278$). Pointer and Gesture had medians of 4.67 and 5.01 degrees, Controller had a median of 5.00 degrees, and Hand had a median of 5.94 degrees. Post-hoc tests showed that Hand had significantly larger errors than Pointer and Gesture. Unlike in the isolated Rotation task, Controller did not show a significant advantage in final Docking rotation error. This indicates that the main weakness of the smartphone methods in Docking was not final attainable accuracy, but the time and effort required to reach and confirm the aligned state.

5.3. Subjective Ratings

Subjective ratings showed significant method effects for Rotation ease ($\chi^2(3) = 16.34$, $p = .001$, $W = .454$), Docking ease ($\chi^2(3) = 11.68$, $p = .009$, $W = .324$), low mental load ($\chi^2(3) = 19.89$, $p < .001$, $W = .552$), low physical load ($\chi^2(3) = 22.09$, $p < .001$, $W = .613$), and overall rating ($\chi^2(3) = 19.57, p < .001, W = .544$). In contrast, no significant method effects were found for Selection ease ($\chi^2(3) = 4.32$, $p = .229$, $W = .120$) and Translation ease ($\chi^2(3) = 5.13$, $p = .163$, $W = .142$), indicating that participants evaluated all methods relatively favorably during these fundamentally simpler interaction phases.

For Rotation ease, Hand received the lowest evaluation with a median score of 2.0, whereas Controller and Pointer both achieved medians of 6.0, and Gesture followed with a median of 5.0. Post-hoc tests revealed that Hand was rated significantly lower than all other three methods ($p_{holm} < .05$). This outcome strongly aligns with the objective performance metrics, confirming that bare-hand rotation was highly challenging, while the smartphone's orientation input (Pointer and Gesture) was perceived as comparably usable to the dedicated controller. For Docking ease, although the overall main effect

was significant, the conservative post-hoc pairwise comparisons with Holm correction did not detect any specific significantly different pairs ($p_{holm} \geq .0703$ for Hand vs. Pointer).

Regarding workload-related parameters, Hand consistently imposed the heaviest burdens. The median score for low physical load was 2.0 for Hand, compared to 6.5 for Controller, 5.5 for Pointer, and 5.5 for Gesture. Similarly, the median score for low mental load was 3.0 for Hand, 7.0 for Controller, 6.0 for Pointer, and 5.5 for Gesture. Post-hoc analysis showed that Hand caused significantly greater physical and mental workload than all other three methods ($p_{holm} < .05$). Notably, Controller was rated significantly better than Gesture for low physical load ($p_{holm} = .0391$), suggesting that Gesture's explicit command input and sequential mode-switching structure induced not only cognitive friction but also non-negligible physical strains.

Vibrotactile feedback helpfulness was evaluated positively across all supported techniques, showing no significant difference among Controller (Median = 6.0), Pointer (Median = 5.0), and Gesture (Median = 6.0) ($\chi^2(2) = 0.64, p = .728, W = .027$). This result demonstrates that haptic cues were consistently valued regardless of the interaction metaphor, validating our design implementation of transferring HMD-side event cues back to the smartphone as vibrotactile feedback, which mitigates the issue that the smartphone's screen is visually unavailable while wearing an HMD.

For the Overall rating, Controller achieved the highest praise with a median of 6.0, followed by Pointer at 6.0, Gesture at 5.0, and Hand lowest at 3.5. Post-hoc comparisons confirmed that Hand was evaluated significantly lower than Controller ($p_{holm} = .0088$) and Pointer ($p_{holm} = .0098$). Furthermore, Controller was rated significantly higher than Gesture ($p_{holm} = .0156$). Crucially, no statistical difference was observed between Pointer and Controller ($p_{holm} = .1875$). This distinct pattern emphasizes that the spatial pointer approach (Pointer) can achieve user acceptance comparable to that of a conventional dedicated controller, effectively presenting a more viable and efficient alternative than gesture-based interfaces or bare-hand interaction.

6. Discussion

The results clarify both the strengths and limitations of using a smartphone as an input device for HMD-based object manipulation. Overall, the dedicated controller remained the most time-efficient method for several tasks, especially Translation, Rotation, and Docking. This result confirms the effectiveness of stable 6-DoF tracking, physical triggers, grip input, and vibrotactile feedback for spatial manipulation. However, the controller did not dominate all aspects of performance. The smartphone methods achieved final accuracy comparable to the controller in several conditions, and crucially, Pointer did not differ significantly from Controller in overall subjective rating ($p_{holm} = .1875$). Moreover, although the differences were not statistically significant, smartphone input numerically outperformed the controller in some task-specific measures: Gesture showed a slightly shorter median completion time than Controller in Selection, and Pointer showed a smaller median final rotation error than Controller in Docking. These results suggest that smartphones should not be dismissed simply because they are slower in some manipulation tasks. Rather, they can serve as practical alternatives when maximum time efficiency is not the primary requirement, particularly in situations where portability, availability, final accuracy, and subjective acceptability are important.

This interpretation is important for future MR and AR-glasses scenarios, especially in outdoor or mobile use. In such settings, users may not always carry a dedicated controller, even if a controller remains the best option for high-speed or highly repetitive manipulation. A smartphone, by contrast, is commonly available and already carried by many users. Outdoor AR-glasses interaction may involve intermittent tasks such as selecting annotations, adjusting virtual objects, confirming commands, or aligning spatial content with real-world locations. In these cases, the relevant question is not whether a smartphone universally replaces a controller in every task, but whether it can provide sufficiently good interaction when the alternative is no controller, hand tracking alone, or a lightweight everyday input setting. The present results support this more practical view from two directions. Compared with the controller, smartphone input was generally slower in complex tasks, but it reached comparable

final alignment accuracy in several cases and was subjectively acceptable, particularly in the Pointer condition. Compared with hand tracking, the smartphone methods showed clearer advantages in tasks where bare-hand input was difficult: Pointer was significantly faster than Hand in Rotation and Docking, both smartphone methods produced significantly smaller final rotation errors than Hand in Rotation, and Hand received lower ratings for rotation ease, workload, and overall evaluation ($p_{holm} < .05$). These results suggest that smartphone input is valuable not only because it can approximate some aspects of controller performance, but also because it can compensate for weaknesses of device-free input. Thus, smartphone input is better understood as a context-dependent auxiliary controller: it can complement hand tracking or gaze input and provide a practical substitute for a dedicated controller in tasks where portability and availability matter more than maximum speed.

The two smartphone methods also revealed different design opportunities. Gesture was most effective in Selection, where it achieved the shortest median completion time and did not differ significantly from Controller. This suggests that a smartphone touchscreen can be useful when it is used for direct symbolic or command-based input rather than for imitating spatial controller motion. In the Selection task, drawing a target number allowed users to specify the intended object directly, reducing the need for precise cursor alignment. Pointer, on the other hand, was more suitable for continuous manipulation. In Rotation, Pointer was significantly faster than Gesture and Hand, and its completion time did not differ significantly from Controller. These results indicate that smartphone-based interaction should not be treated as a single technique. Discrete touchscreen commands and continuous pointer control support different stages of object manipulation, and a future system may benefit from combining them adaptively.

The touchscreen is therefore central to the value of smartphone input. A controller is strong because it provides tracked spatial input and clear physical state transitions through buttons and triggers. A smartphone is strong for different reasons: it provides a familiar high-resolution surface that can support touch positions, strokes, multi-touch, screen-region commands, visual feedback, and vibrotactile feedback. The results of Gesture in Selection are especially relevant to this point. They show that the smartphone can exploit interaction styles that are not available, or are less natural, on a conventional controller. This suggests that smartphone-based HMD input should not simply copy controller interaction. Instead, future designs should more fully exploit the touchscreen, for example through symbolic commands, mode palettes, precision sliders, snapping controls, or combined touch-and-motion techniques.

At the same time, the smartphone methods still introduced interaction costs. Translation and Docking were slower with Pointer and Gesture than with Controller. This delay appears to come mainly from the staged interaction structure of the smartphone methods. Compared with Controller and Hand, the smartphone methods separated selection and manipulation more explicitly: users had to move from selecting an object to issuing a movement or rotation command through dragging, gestures, or touch states. These transition costs are especially visible in Docking, where users repeatedly selected, aligned, and deselected three object-target pairs. For Docking ease, although the overall Friedman test showed a significant main effect ($\chi^2(3) = 11.68, p = .0086$), the strict Holm post-hoc correction did not find any specific significantly different pairs ($p_{holm} \geq .0703$ for Hand vs. Pointer). This suggests that while there was a general trend showing that device-based inputs were preferred over bare hands, the continuous selection and state transitions across multiple objects introduced high variance and descriptive ties in scores, blending the pairwise boundaries under a conservative multi-comparison penalty. Importantly, however, the final accuracy results suggest that the smartphone methods were able to reach the target once users spent enough time. The remaining challenge is therefore not sensing accuracy alone, but interaction efficiency: simplifying mode transitions, reducing state-management cost, and helping users converge quickly near the target.

The results also clarify the limitations of hand tracking in this experiment. Hand was conceptually direct and device-free, but it was slower in Rotation and Docking and received lower workload and overall ratings. It also produced larger final rotation errors than the other methods in Rotation. This tendency is consistent with prior work showing that hand-based input can provide naturalness and immersion, but may be disadvantaged in time, accuracy, and physical exertion compared with controller-

based input for precise object manipulation. Kangas et al. reported that hand-only input was significantly slower than controller-based input in direct rigid-object manipulation in VR, and that there is a trade-off among naturalness, task completion time, and accuracy [15]. Luong et al. showed that controller input was faster and more accurate than free-hand input with lower physical exertion, especially in raycast interaction [16]. Hameed et al. also showed that although hand tracking can provide a natural experience, handheld controllers can provide more reliable input in VR reach-grab-place tasks [17]. In the present study, maintaining pinch or grasp-like gestures, keeping the hand stable, and switching input states without physical confirmation appear to have imposed workload. In addition, hand tracking required participants to keep the hand within a visible and trackable region in front of the body, which may have forced them to keep the arm raised during manipulation. This posture may have contributed to arm fatigue similar to the gorilla-arm effect reported in mid-air interaction research [18]. By contrast, the smartphone methods allowed users to perform most touch-based operations without holding the smartphone itself at a specific position in the HMD view, except when depth movement required right-hand displacement. Crucially, the post-hoc tests revealed that Controller significantly outperformed Gesture in low physical load ($p_{holm} = .0391$). This indicates that Gesture's explicit gesture inputs and repetitive command strokes over the touchscreen induced not only cognitive demands but also non-negligible physical exhaustion over the user's wrists and fingers compared to a well-balanced ergonomic hardware controller.

These findings support a hybrid view of input for future HMD, MR, and AR-glasses environments. For high-speed, repeated, or professionally demanding manipulation, dedicated controllers remain highly effective. For lightweight everyday interaction, especially outdoors, users may prefer not to carry an additional controller or may need to interact while standing, walking, or switching attention between the physical environment and digital content. In such situations, hands or gaze may handle quick and coarse operations, while a smartphone can be introduced as an auxiliary controller when explicit selection, precise adjustment, prolonged manipulation, command input, or reliable state confirmation is needed. The task-dependent advantages observed in this study support this role: the smartphone was not uniformly superior, but it approximated the controller in selected measures (particularly in the Pointer layout) and outperformed hand tracking in several rotation- and workload-related measures. This framing positions the smartphone not only as a fallback device, but also as an input surface that can add capabilities that hands, gaze, and controllers do not individually provide.

Future smartphone-based HMD input should therefore focus on reducing the costs identified in this study while further exploiting the touchscreen. Mode transitions should be reduced or made more continuous, because select-to-manipulate transitions increased time and effort. Near-target assistance, such as dynamic gain adjustment, axis constraints, snapping, target attraction, and automatic refinement, may help users converge quickly without reducing their sense of control. Finally, visual and vibrotactile state feedback should be carefully designed, because smartphone users wearing an HMD cannot always look at the phone screen. Improving these aspects could make smartphone input more competitive with controllers not only in final accuracy and acceptability, but also in interaction speed.

This study has several limitations. First, the experiment was conducted in VR rather than MR. VR allowed controlled object placement and repeatable task conditions, but it does not fully reflect real-world spatial context, visual clutter, walking use, or environmental constraints. Second, the tasks were not designed as strict precision-measurement tasks. The success thresholds for position and rotation were used to define practical task completion, and the recorded final errors indicate the alignment accuracy achieved within these task constraints. Therefore, the accuracy results should be interpreted as evidence that the methods can reach comparable practical alignment, rather than as a direct measurement of the maximum attainable precision of each input method. Third, the sample size was limited to 12 participants. Although the within-subject design revealed clear effects for task completion time and several subjective ratings, future work should test a larger and more diverse participant group. Fourth, the smartphone methods are initial implementations. The present results should therefore be interpreted as evidence of practical potential and design trade-offs, not as a final optimized smartphone interaction technique.

7. Conclusion

This paper presented and evaluated smartphone-based input methods for HMD-based object manipulation. We implemented Gesture, a discrete command-oriented method using numbers and stroke gestures on a touchscreen, and Pointer, a continuous pointer-oriented method using smartphone touch input and device orientation. We compared these methods with a Meta Quest Touch Plus controller and hand tracking in Selection, Translation, Rotation, and Docking tasks.

The results showed that the dedicated controller remained the most time-efficient method for several manipulation tasks, especially Translation, Rotation, and Docking. However, smartphone input showed practical potential in specific operation stages and did not simply underperform the controller across all measures. Gesture achieved selection times comparable to the controller, demonstrating the effectiveness of label-based discrete target specification. Pointer supported continuous rotation effectively, and crucially, did not differ significantly from the controller in the overall subjective rating ($p_{holm} = .1875$). In addition, the smartphone-based methods achieved final position and rotation accuracy close to the controller in several conditions, and in some task-specific measures they numerically exceeded the controller despite the absence of statistically significant differences.

These findings indicate that smartphones should not be viewed as universal replacements for dedicated controllers in all HMD manipulation tasks. Nevertheless, they can serve as practical alternatives in situations where carrying a dedicated controller is undesirable and where the task does not require maximum time efficiency, allowing slightly longer completion time in exchange for portability and availability. This role is particularly relevant to outdoor or mobile AR-glasses use, where users are likely to have a smartphone but may not carry a dedicated controller. In such contexts, a smartphone can function as an auxiliary controller that complements device-free inputs such as hand tracking or gaze input for intermittent tasks. This claim is supported by the comparison with Hand: smartphone input was significantly more effective than bare-hand input in rotation- and workload-related results ($p_{holm} < .05$), suggesting that it can compensate for limitations of device-free manipulation while preserving portability. Crucially, our precise post-hoc analysis also highlighted method-specific interaction costs: Gesture was rated significantly lower than Controller in overall subjective acceptability ($p_{holm} = .0156$) and induced higher physical exhaustion ($p_{holm} = .0391$ for low physical load), indicating that mimicking discrete strokes over a touchscreen introduces notable user strains compared to a continuous pointer or an ergonomic hardware controller. The main value of smartphones lies not only in their availability, but also in their touchscreen, which enables interaction styles that are different from controller imitation. By combining touchscreen input, device orientation, and vibrotactile feedback, smartphone-based HMD interfaces may support explicit, practical, and portable interaction in future MR and AR-glasses environments. Future work should focus on reducing mode-transition latencies, mitigating finger/wrist fatigue in stroke inputs, and exploring adaptive snapping or target-proximity assistance under actual outdoor and mobile MR scenarios.

Declaration on Generative AI

During the preparation of this work, the author used ChatGPT to support the drafting and revision of the English manuscript structure and wording. The author also used a generative AI tool to create one schematic figure illustrating the gesture-command method. The author reviewed and edited all generated text and figure content as needed and takes full responsibility for the final content of the paper.

References

[1] R. Darbar, A. Dey, M. Billinghurst, Exploring smartphone-enabled text selection in ar-hmd, in: Proceedings of Graphics Interface 2021, 2021. doi:10.20380/GI2021.06.

[2] P. Knierim, D. Hein, A. Schmidt, T. Kosch, The smartphone controller: Leveraging smartphones as input and output modality for improved interaction within mobile augmented reality environments, i-com 20 (2021) 49–61. doi:10.1515/icom-2021-0004.

[3] B. McDonald, Q. Zhang, A. Nanzatov, L. P. na Castillo, O. Meruvia-Pastor, Smartvr pointer: Using smartphones and gaze orientation for selection and navigation in virtual reality, Sensors 24 (2024) 5168. doi:10.3390/s24165168.

[4] Z. Dong, J. Zhang, X. Bai, A. Clark, R. W. Lindeman, W. He, T. Piumsomboon, Touch-move-release: Studies of surface and motion gestures for mobile augmented reality, Frontiers in Virtual Reality 3 (2022) 927258. doi:10.3389/frvir.2022.927258.

[5] N. Katzakis, K. Kiyokawa, M. Hori, H. Takemura, Plane-casting: 3d cursor control with a smartphone, in: Proceedings of the 3rd Dimension of CHI: Touching and Designing 3D User Interfaces, 2012, pp. 13–21. doi:10.1109/3DUI.2012.6184179.

[6] K. Waldow, M. Misiak, U. Derichs, O. Clausen, A. Fuhrmann, An evaluation of smartphone-based interaction in ar for constrained object manipulation, in: Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology, VRST '18, 2018. doi:10.1145/3281505.3281541.

[7] I. Poupyrev, M. Billinghurst, S. Weghorst, T. Ichikawa, The go-go interaction technique: non-linear mapping for direct manipulation in vr, in: Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology, UIST '96, 1996, pp. 79–80. doi:10.1145/237091.237102.

[8] D. A. Bowman, L. F. Hodges, An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments, in: Proceedings of the 1997 Symposium on Interactive 3D Graphics, I3D '97, 1997, pp. 35–42. doi:10.1145/253284.253301.

[9] Meta, Create ray interactions, Meta Horizon OS Developers Documentation, 2026. URL: https://developer.oculus.com/documentation/unity/unity-isdk-ray-interaction/, [Online; accessed 3-June-2026].

[10] Meta, Meta quest 3s: Next-gen virtual reality headset, Meta Quest product information, 2026. URL: https://www.meta.com/quest/quest-3s/, [Online; accessed 3-June-2026].

[11] Meta, How to use meta quest touch plus controllers, Meta Quest Help Center, 2026. URL: https://www.meta.com/help/quest/articles/getting-started/getting-started-with-quest-3/touch-plus-controllers/, [Online; accessed 3-June-2026].

[12] Apple, iphone 13: Technical specifications, Apple Support, 2026. URL: https://support.apple.com/en-us/111883, [Online; accessed 3-June-2026].

[13] P. M. Fitts, The information capacity of the human motor system in controlling the amplitude of movement, Journal of Experimental Psychology 47 (1954) 381–391. doi:10.1037/h0055392.

[14] Eaton Hand, Normal range of motion reference values, Eaton Hand, Anatomy and Hand Surgery Education, 2026. URL: http://www.eatonhand.com/hw/hw003.htm, [Online; accessed 3-June-2026].

[15] J. Kangas, S. K. Kumar, H. Mehtonen, J. Järnstedt, R. Raisamo, Trade-off between task accuracy, task completion time and naturalness for direct object manipulation in virtual reality, Multimodal Technologies and Interaction 6 (2022) 6. doi:10.3390/mti6010006.

[16] T. Luong, Y. F. Cheng, M. Moebus, A. Fender, C. Holz, Controllers or bare hands? a controlled evaluation of input techniques on interaction performance and exertion in virtual reality, IEEE Transactions on Visualization and Computer Graphics 29 (2023) 4633–4645. doi:10.1109/TVCG.2023.3320235.

[17] A. Hameed, S. Möller, A. Perkis, How good are virtual hands? influences of input modality on motor tasks in virtual reality, Journal of Environmental Psychology 92 (2023) 102137. doi:10.1016/j.jenvp.2023.102137.

[18] J. D. Hincapié-Ramos, X. Guo, P. Moghadasian, P. Irani, Consumed endurance: A metric to quantify arm fatigue of mid-air interactions, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '14, 2014, pp. 1063–1072. doi:10.1145/2556288.2557130.