Multimedia Tools and Applications (2024) 83:46625–46652
https://doi.org/10.1007/s11042-022-13637-x

1161: MULTIMEDIA ALTERNATE REALITIES

Incorporating real-world object into virtual reality: using mobile device input with augmented virtuality

Jongkyu Shin[1] · Kyogu Lee[1]

Received: 3 April 2020 / Revised: 13 March 2022 / Accepted: 1 August 2022 /
Published online: 30 August 2022

Abstract

Interactive virtual reality (VR) experience has become more common and widespread these days. The interaction is enabled by capturing inputs from users usually by mechanical hand-held controllers. Mainstream head-mounted display (HMD) based VR platforms come with these trackable multi-DOF controllers with a high level of versatility. This enables a tremendous diversity in input types and forms to provide more interactive and immersive VR experience. However, these devices lack in cross-platform compatibility as the protocols are non-standardized. Thus, it leads to a steep learning curve. The trouble in learning these devices results in non-intuitiveness and confusion to users. Also, the VR isolation may cause problems since the platforms are not designed to reflect any real environment around the user.

In this paper, we investigate an alternate input method for virtual reality by incorporating a real-world object with augmented virtuality. We set a mobile phone as the real-world object for this phase of the study. We import a mobile phone from the real world and evaluate the possibility of utilizing it as an input device. The objective of this method is to give VR users more flexibility in input methods for VR and to reduce the physical barrier between the reality and virtual environment by using augmented virtuality. We propose a method for VR that overlays both real texture of a mobile phone and user's hand onto a virtual scene in real-time by utilizing external cameras attached on the HMD. While being immersed in virtual environment with the HMD, users can manipulate the device naturally for entering inputs to alter the virtual environment without removing the headset.

We implement our prototype in two phases to explore its possibilities as a new input technique for VR. The evaluation was conducted by setting up different kinds of conditions and tasks in virtual environment. The effectiveness of the proposed method

✉ Kyogu Lee
kglee@snu.ac.kr

Jongkyu Shin
jqshin@snu.ac.kr

[1] Music and Audio Research Group, Seoul National University, 1 Gwanak-ro, Gwanak-gu, 151-742 Seoul, Republic of Korea

is verified experimentally in specific given tasks in comparison with conventional controllers. Furthermore, we address and improve important points for accurate real image display in the virtual environment which we have found in the process of the implementation and the experiment. The outcome of the study indicates that our approach based on augmented virtuality, which is a relatively new approach for mixed reality, could lead to a path for an effective and natural input method for VR.

Keywords Virtual reality · Augmented virtuality · Human computer interaction · User interfaces

1 Introduction

Virtual reality (VR) technology enables to deliver visual and aural computer-based synthetic simulated experiences that is similar to or completely different from the real world. The technology is applied in many areas including video games, engineering, entertainment, education or any areas that require experiences that is hard or impossible to archive in real life. Recent development of consumer-based Head Mounted Display (HMD) systems enabled users not only to easily access to the visual experiences of static visual scenes, but also to actively participate in the virtual reality (VR) environment with a great level of immersion. Combining with modern computer graphics, previous generation devices such as Oculus Rift DK[1] series, and the low-cost smartphone-based systems such as Samsung Gear VR focused on delivering stereoscopic imagery to users yet had limited ways to gather input entries from users and let them participate in VR scene. To interact with the virtual environment, indirect manipulation methods with low-resolution such as gaze input or gesture detection were utilized as there was no better way. As a workaround for manipulation and interaction in VR, there came current generation HMD platforms with tracking capability. The platform can track multiple objects including the HMD and dedicated controllers. Thus, they came out with packages including their dedicated trackable hand-held mechanical controllers which have multi degrees of freedom (DoF). Most controllers come with multiple buttons and touch pads, supporting different types of fingertip inputs. This technology fills the missing key to the definition of VR mentioned by Burdea G et al. [9], which is the communication.

The complex trackable controllers with multiple input ways create a great opportunity for diverse input-based interaction methods, and they work flawlessly in 3D-specific tasks or pre-mapped scenarios with rules to follow (e.g. ray-cast to point an object then click to grab [14]). However, those devices tend to have a steep learning curve in usability because their operating methods are not standardized and inconsistent throughout different platforms. Often these complexity results in lack of intuitiveness, confusion in control and unnatural interaction for users [5].

This direction of HMD platform development is the corresponding response to the proliferate demand for a more immersive virtual reality experience with a higher level of sensory deception. Moreover, this path leads to the right end point of the Mixed Reality spectrum introduced by Milgram and Kishino [30] shown in Fig. 1, which is the Virtual Environment. It is an artificially created world with capturing zero elements from the real world. The purpose of this environment is to give users a complete immersion towards virtual reality, trying to stimulate all human sensory at the same time without any unintended stimuli. On the other

[1] Oculus. https://www.oculus.com/

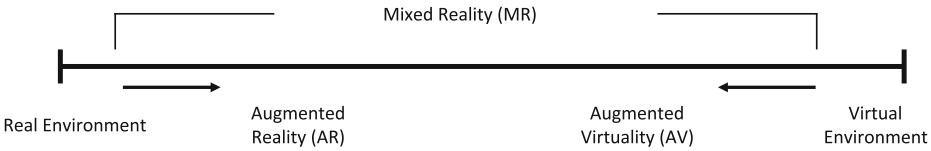

Fig. 1 The mixed reality continuum [30]

hand, the isolation problem inevitably arises as users get extremely difficult to realize the context of the real world, nor be able to communicate visually or aurally with outside in real-time. This concept doesn't allow users to see or hear things from outside while they are immersed in VR; Conversely speaking, they are isolated in the VR. Previously mentioned modern technology platforms usually target end-users, this could be a serious problem and considered as inconvenient or dangerous [29], as those platforms are likely to be installed and utilized in our everyday life environments such as home and office where multiple obstacles do exist.

To implement a natural, intuitive input technique for virtual reality with a proper visual representation for minimized virtual isolation and physical barrier, we focus on the research field of the augmented virtuality (AV). AV is a subset of VR technologies which is placed next to AR in the Mixed Reality Continuum. AV is referred to as to have a "window" inside the virtual environment to see the outside real world and mostly implemented by using at least one external cameras on HMDs or outside environment.

In this paper, we investigate a novel AV-based input method for VR other than methods that involve dedicated controllers. We propose an approach that enables users to directly interact with an object placed outside VR yet maintaining the level of immersion for the virtual environment by blending well the visual components in two different worlds. For this phase of the study, we utilize a mobile phone as a medium to link the virtual world and the real world with the form of AV. This approach has the advantage of establishing a flexible virtual environment yet considering the relevant data of the real environment for filling the gap between different worlds. Blending both worlds can benefit for more expandability and ensure a more intuitive platform by using user-friendly input devices for virtual reality.

The feasibility of the prototype is validated through series of user studies. Within the implementation, we first compare the proposed method with conventional platform with hand-held controllers to quantify how the system affect the overall virtual reality experience as well as the usability with a variety of scenarios and interaction types. After investigating the existing limitations of the method, we rearrange and modify implementations for improvement. The final results demonstrate that the proposed input method significantly improves performance in certain tasks that involves 2D-based UI in VR, yet does not harm immersive VR experience, and minimizes the sense of difference between the reality and the VR. Thus, our AV-based input method with real-world mobile phone has a promising potential to be utilized in virtual environment in future studies.

The contributions of this work are summarized below:

– **Expanding the option to perform an input-based interaction in virtual environment, which is currently limited to dedicated hand-held controllers:**

  Numerous approaches have been conducted using see-through VR (AV) and mobile phone, but none of them considered integrating those devices with the virtual

environment. As we call the device as a "controller", we implement the prototype that the outside object could be used to alter the virtual environment. As this is our initial step for the study, we mainly focus importing the visual image of real-world mobile phone into the virtual environment, and investigate on feasibility of using it as a VR controller.

– **Designing tasks and validating the system for comparison against the conventional input method in VR, investigating how AV-based input technique affects input performance and usability:**

  Since the common and the widespread input device for VR platform is the mechanical controller, we set the condition as the baseline and evaluate our method with a novel set of tasks in VR, and conduct user study including SSQ and SUS.

– **Finding a key factor when visually incorporating a live element into VR, address them, and conduct experiments to see how our method maintains the level of cognition to VR users.**

  Our initial results from the first experiment showed that the stereoscopy of the imagery is a crucial factor when establishing an AV environment. Thus, we compare our proposed method with a bare-eye condition in order to measure the level of user confusion.

The remainder of this paper is organized as follows. In the next section, we describe previous methods and studies in this area related to our study. In the following section, we describe our system in detail, including the visual representation of the configuration and implementation of the object overlay technique and prototype. We then present our experimental setups followed by the results section, and discuss the key findings and limitations with our attempt to improve our first approach. Final suggestions and findings are given in the next section, followed by the conclusion of this study.

2 Related work

In this section, we describe input methods that were previously conducted in attempts to provide immersive virtual reality experience to users in various forms and shapes. In addition, we focus on previous studies on the combination of reality and virtual reality, especially on the augmented virtuality term.

2.1 Input techniques for VR

The input techniques in virtual environment can be categorized into two groups; passive expression methods that captures users' body motions as input signals and methods which involves external physical devices.

Passive expressions for input in virtual environment involve implementations to monitor users simultaneously for capturing pre-defined motion change that would be utilized as trigger/input signal for manipulation. Maggioni [28] introduced hand gesture input system by using RGB camera to monitor user's hands and segment hands image, then estimate the gesture to be utilized in the 3D environment. This hand tracking approach was followed by adding depth to

the awareness [33, 42] and allowing a more precise tracking by tracking fingers. Finger touch input was also used as input signals in mixed reality (MR) scenarios [25, 43, 44] by enabling fingertip gestures when a user approaches a surface. Not only applied to hand but extending the range to arm approach was also used [32]. Expanding the tracking methods from single body parts to full body were also explored by Latoschik [26] and Caserman [13]. Under tracking full-body platforms, human articular surface tracking works as the basis of all motion recognition.

The methods to track body parts (i.e., hands, fingers) were not able to be used in scenarios which required a high precision level, as they generally only respond to distinguishable big movements or large delta values between the image frames for posture detection. Also, vision-based tracking approaches had latency problems as they required long processes for posture classification. Ranging from seven to approximately 300 ms [15, 22, 23, 26], the latency in synchronizing the user movements in real world with those in virtual environment was carefully considered among all studies to minimize the risk of cybersickness symptoms in VR [38] and not to deteriorate the sense of immersion [18]. Additionally, the lack of haptic/tactile feedback in passive expression methods created problems as the latency combined with vague feedback often led to user confusion and a low level of user usability as the motions were almost mime-like.

Gaze input is another input method without physical device, often involves eye-tracking [31]. This interaction technique utilizes user's direction of gaze as the input signals. A gaze at a fixed angle towards certain point would position item, and longer fixation performs the select. This simple method was also integrated into modern smartphone-based HMDs, on to systems even without the eye tracking support. A simple workaround for this to work was to set the center of the screen as the initial position cue, then utilize rotational values of the device as a directional modifier. The limitation of gaze input combined with current generation HMD could be mentioned as losing accuracy overtime. As the systems utilizes relative head positions using 3-axis gyroscope sensors, recalibration is required in certain periods of time due to slippage of the sensor values. Maintaining center becomes harder after usage due to noise from outside world. Other methods utilizing eye-tracking methods mainly suffer from delay, lost gaze position, and out of envelop [37]. Out of envelop refers the wandering eye that moves to the target in a curved path, prone to error eye movement that is out of the prediction trajectory model.

Other passive input approach includes the speech command system [12]. An example of the system is a web-based police training tool [20] to use the voice commands as input for controlling a virtual robot in VR. As the accent would vary from user to user, it requires a pre-recording session for system to learn and classify exact commands for voice recognition. Although the large vocabulary continuous speech recognition (LVSCR) decoder used in the research claims to be able to perform in real-time, misinterpretation or noise often made the system unstable for controlling input. In these word-based speech input systems, only simple chunks of pre-selected words could be utilized as inputs and required controlled environment as the noise worked as a critical failure point.

Posture and gesture tracking platforms that require outside cameras for monitoring often suffered from occlusions. This led to a loss of tracking while recognizing the motion of the user and eventually made the platform unreliable. As the problem was coming from a passive, third eye perspective way of monitoring, numbers of studies focused switching the perspective into first person view by utilizing physical devices designed to be attached on the user or to be held by the user, rather than utilizing monitoring devices placed outside of the players.

Device-oriented input methods showed different types and forms in implementation including hand-worn devices [17], hand-held wand style controllers, and non-universal unique structures for specific contexts [7]. Input methods involving physical devices mainly focused on egocentric view of the user, as it enabled the perspective match from outside to inside. Hand-worn and hand-held input devices are usually universal controllers for VR contents, meaning that they provide diverse input methods compared to specific input structures designed only for a single scenario.

The wand style controllers are refereed as the standard physical input controllers for commercial HMD systems including the mobile based simple HMD platforms recently. The degree-of-freedom (DoF) of controllers vary from 3Dof to 6Dof, depending on the type of the platform. As most PC-based HMD now supports motion tracking by indoor positioning systems (IPS), their controllers are also tracked in the installed environment. This enables 6DoF on their controllers, as cartesian coordinate data of the controllers is recognized inside the virtual environment. To ensure the expandability in input, they are equipped with multiple clickable buttons, axis input touchpads and integrated with a myriad number of sensors for acquiring positional/rotational values of the controller. However, these systems are not interoperable among different platforms or contents, and often claimed to be too complex, acquires a high level of learnability and being not user centric.

2.2 Augmented Virtuality

While being immersed in virtual reality with wearing HMD, users challenge the isolation from the outside world. Because the vision is occluded by the HMD, it is nearly impossible for users to interact with an object placed outside, and even they could it creates a dangerous scenario as they cannot avoid collision within tracking space and cannot respond to sudden change of the real-world context. To overcome the isolation of the virtual reality, researchers focused on combining both virtual and real elements, augmenting reality onto virtual environment [29]. Augmented virtuality overlays real elements from the real world on to the synthetic background of virtual environment, mostly implemented by mounting cameras on HMDs or integrating other monitoring solutions to capture visual elements. Numerous applications under this term was developed for games, simulations, engineering and education, with interactivity [45]. The AV did not get much attention compared to augmented reality (AR), as there were more technical obstacles to break. Those obstacles include a low-resolution display, a tethered device platform and the necessity of an active power.

The SIMNET [11] is one of an early work that combined a real element into a virtual scene. A head mounted camera was placed on the HMD to provide a window to the outside from inside virtual environment, giving users an ability to visually perceive objects placed outside their HMD. Steinicke et al. [36] used chromakeying technique to import user's hands into the scene by also using the camera outside the VR interface. This method with the usage of a depth camera was further investigated by Tecchia et al. [39]. They used a motion capture platform to track multiple markers installed on the HMD and user's fingertip rings to create an interactive platform that could lead users to manipulate virtual objects with their actual hands.

Similar approaches to our work, in order to smoothen the isolation problem in virtual reality are the SDSC [16] by Desai et al., and NRAV [2] by Alaee et al. Both implementations import a mobile phone to a VR scene and let users recognize the outside context with the real object. In the research of Desai et al., they used Smartphone Detector based on a Statistical Classifier (SDSC) method with a LeapMotion controller. The approach resulted in high accuracy in

detecting smartphone within the field of view (FoV) of an HMD user but had inability to show the screen in real-time without a minimal delay as they were capturing the stream of screenshots instead of live stream of the visual data as the image representation. In NRAV system, they also represented the real image of mobile phone into the virtual scene to smoothen the isolation in VR and let users utilize their phones. In this research, the phones were used with the same purpose as it had in real life, in the case of adding an attempt to utilize as an input device for VR. In this work, we focus more on importing objects from outside for the purpose of using them as input devices in the virtual environment. Paperstick[2] is an example of utilizing an outside, offline object into the VR scene and use it as a control device, but the texture of the object remains synthetic, lacking in real-image visual cues and limited ability to deliver visual feedback to users.

Previously introduced methods in AV term lacks in three conditions. First, they could not import the original texture of the object into the virtual environment. Most methods substitute synthetic texture to represent the object. This also led to a problem that users' hands are not visible with the HMD, making it difficult to let users recognize the initial position of the object. Second limitation is the lack of abilities for simultaneous data transmission with outside objects. As gathering signals for status change of the outside object was not possible, it was impossible to use the object as interactable device. Objects with LCD screens which were utilized in some methods, but they were only able to work as additional information displays for users. And finally, most methods could not response to the necessity for precise input control. The existence of resolution and latency issues made limitations for methods to be utilized in scenarios which required to control high-density interfaces, such as soft keyboard UI.

In this study, we implemented our augmented virtuality method to use mobile phone for virtual reality, using it as a medium for linking the real world and the virtual reality. We compare our method by evaluating the effectiveness in terms of performance and presence in virtual environment, aiming to establish guidelines for the design of AV environment and explore gaps in the current literature. The remainder of this paper presents the investigations of our methods and findings.

3 Prototype design

We explain our augmented virtuality based input method prototype with details in this section. The purpose of the prototype is to enable mobile phone manipulation without taking off the HMD by importing outside imagery into the virtual environment and establish a data transmission functionality between the mobile device and the 3D engine. First, we describe the system design of the proposed method. In the following subsection, we explain the process we went through for implementing the pattern-matching technique to recognize a target object. Finally, we describe the specific configuration and how it integrates all together.

3.1 System architecture

Figure 2 illustrates the overall architecture and elements of the system. Our system mainly consists of four elements which are the HMD, the 3D rendering engine, the RGB camera and

[2] Paperstick. https://sites.google.com/site/gameplusvr/

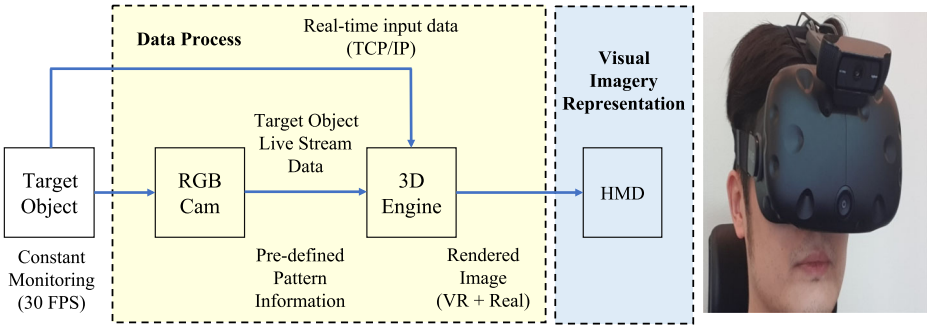

Fig. 2 Overall system architecture

finally the target object, which is a mobile phone in this prototype. The RGB camera is attached on the top of the HMD as shown in the right side of Fig. 2. The camera mounting position is calibrated to be aligned with the center of the HMD, for the horizontal field of view (FoV) of the RGB camera would stay inside of the HMD FoV all the time. This enables to deliver the image of the target object to users even in scenarios which users turn their heads side to side to change facing directions. Also, the vertical angle for facing direction for the camera is set roughly to 130° degrees downwards as we assumed that the target object will be placed on the under area of the HMD where the users' hand would be placed. The RGB camera is a Logitech C920 webcam which has a vertical FOV of 43.3° degrees, and even if the placement of the camera sometimes fluctuates due to different style of HMD wearing positions upon different users, initial test showed that there wasn't any serious issue reported related to this slight angle misplacement. The pre-defined target object texture data is stored in the 3D Engine, which is Unity 3D. In the engine, pattern-matching scripts are set to constantly check on every frame if the pattern exists in the current frame. If detected, the system overlays the detected object into the current playing scene inside the FOV of the HMD, otherwise it does not show. When represented in the VR scene, users are able to see the actual phone and interact with the device. A Samsung Galaxy S9 Android phone acts as the target object in this method. The real-time input data from the users is simultaneously transmitted to the Unity 3D engine by a custom Android app.

3.2 Pattern matching and integration

We used OpenCV for processing the simultaneous image stream of the target object. OpenCV is an open-sourced library functions mainly aimed at real-time computer vision. Unity 3D mainly supports development environments based on Microsoft C# or Java Script. Since OpenCV does not natively support those languages mentioned, a modified wrapper version on C#, OpenCVforUnity was used. Among various pattern matching algorithms such as ORB [35], SIFT [27], SURF [4] and AKAZE [3], we decided to utilize the ORB as it is known to be fast and rotation invariant. ORB is a fusion of FAST [34] keypoint detector and BRIEF descriptor with modifications to improve the performance [10]. It uses FAST to find keypoints at first then to find the top N points, a Harris corner measure is applied. As FAST does not detect rotation, it computes the intensity weighted centroid of the patch with located corner at center. The direction of this vector from corner point to centroid gives the orientation. An ability for fast detection with rotation invariance and partial scale invariance was efficient

enough to be applied to our method balancing between the performance and the delay. We modified the maximum number of features to be retained, which is the nFeatures and is set to 800. In the very edge of keypoints, four corners (top left, top right, bottom left and bottom right) is created and put into a new image matrix of the size of the target object then represented as a rectangle with cropped RGB input matrix of the current frame. The rectangular image of the target object is represented in the VR scene. Figure 3 shows target object, with the trace of the target using the recognized patterns. The actual trace visualization is not visible to users in real usage scenarios. Unlike other AR based implementations that mainly focus to track users' hand or arm exclusively, our method did not consider those elements and focused only on the target object.

After the initial implementation, an additional idea was considered. Since there are no visual cues while "holstering" the phone (e.g., pockets, outside the camera FOV) and to help the users to see the object always straight, a perspective wrapping method was applied. This enables to see the straight top-down view of the image regardless of the rotational angle of the target. However, there were some issues with this method discovered on the first pilot test. As the algorithm continuously checks the pattern and wraps perspective every frame in Unity 3D, we found out a problem that in some ambiguous environments with much of visual noise, the image represented kept flickering and twisting. The users reported that this problem caused discomfort and made them hard to concentrate on executing inputs. And also, they claimed that auto-correcting the rotation causes more confusion. This method was later discarded in the main user study and the method was switched back to the previous ORB based pattern-matching detection.

All virtual imagery rendering, real world overlaying and input control were managed on Unity 3D engine. The communication between the Unity engine and the Android device was implemented using TCP/IP protocol. Communication scripts for both platforms have been created in order to transmit input data from the device and show received data on the engine in real-time. With all components mentioned above, users are able to manipulate an external mobile phone as controller within a VR context. The live image feed from the camera was

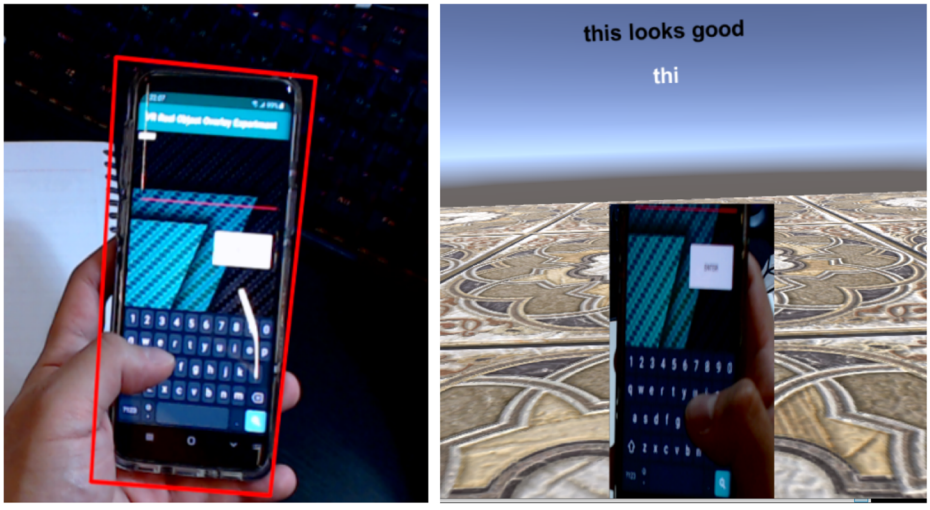

Fig. 3 An example of a target object in sight with trace visualization (left) and actual representation in VR scene (right)

done at 720p (1280 × 720 resolution). We used 720p instead of 1080p (1920 × 1080 resolution) even the camera itself was capable of a higher resolution, due to delay caused by slow data pipeline of the Unity 3D. As we were utilizing default WebcamTexture API to deliver outside image to inside, it was difficult to maintain minimal frames per second (FPS) of 30 when using 1080p. Thus, to maintain fastest performance with maximum readability, 720p was selected. However, the methods to resolve this issue do exist, which will be discussed later in discussion. Nevertheless, the image at resolution 720p was still able to deliver an enough level of readability for users to recognize contents while wearing the HMD.

As mentioned earlier, a mobile phone is considered as an object without a texture in general, because usually it is not represented with continuous or repetitive visual patterns on its screen surface. To utilize the keypoint-based pattern matching technique in this situation, we designed the screen contents to have recognizable textures by the RGB camera. Then, fixed the LCD screen brightness of the target device to minimize visual noise. The pattern for the target object was pre-captured before deployment and saved in Unity 3D engine. In addition, a feature to save and utilize multiple snapshots of different keypoint patterns was implemented to cope with scenarios with multiple objects, each with different texture patterns. During the VR play, all the pre-recorded patterns can be utilized inside the virtual environment and they are able to be switched in-between scenes to recall/detect the corresponding pattern for the context.

4 Experiments

In order to measure the performance of the prototype system in terms of input in virtual environment, multiple tasks were designed considering the types of the input for different scenarios in virtual scenes. Conditions to carry the tasks were separated into three groups including the baseline, while differentiating the real image representation inside the VR with two conditions.

4.1 Task design

The tasks for the experiment were consisted of three different sets. Each task was designed to investigate the usability and the performance of the proposed method upon different scenarios in VR which involves different types of UI elements and input styles. The structure and the placement of the UI elements were designed identical across the baseline and two overlay conditions to minimize a risk of perception gap and confusion across participants caused by the platform difference. In addition, we added slightly different variations and performing orders for the same task, to maintain a balance throughout the repeated measurements. In the following, we describe the components and structures of the tasks in detail. Visual representations of each task are shown in Fig. 4.

- Menu Selection. Menu selection task was designed to utilize one hand. In a static location, participants manipulated the menu UI accordingly to given instructions. As shown in Fig. 4, we utilized commonly used software input components of graphics user interface (GUI) when creating this. And also, the menu had a hierarchy, as the main menu is separated into three submenus including toggle switch, slider and sequence button input. When the virtual scene started, the participants first entered to the main menu where three choices

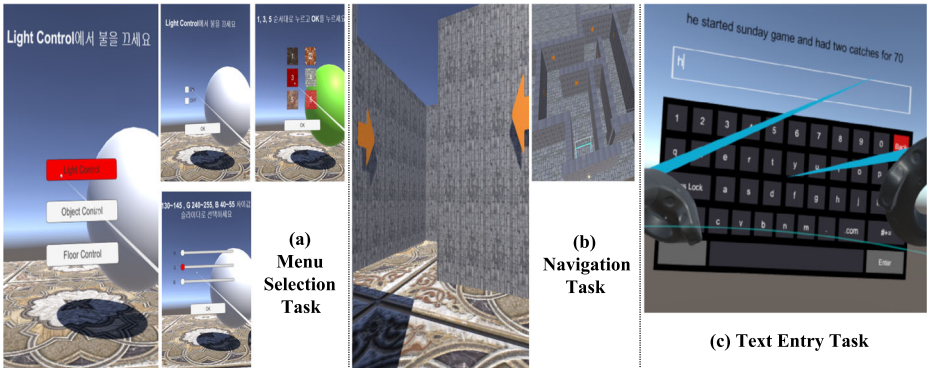

Fig. 4 Visual representations for each task. Starting from the left: (**a**) Menu Selection, (**b**) Navigation, and (**c**) Text Entry. The menu design for Overlay conditions were implemented identical to the baseline using Android UI elements. For the text entry task, the keyboard UI element was excluded in Overlay conditions. The locations of instruction text boxes shown in (**a**) and (**b**) remained the same for both Overlay conditions

could be made to navigate to different submenus. The submenus had three little tasks: 1) Toggle switch manipulation for binary on/off, 2) Three slider UI elements to give different number combinations as answers and 3) Six clickable buttons with numbers for providing sequential press order. The participants completed all three small tasks, and an instruction for a task order was given to the participants one at a time in a random order.

The toggle switch was used for turning on and off the light of the virtual environment. The instruction was either to turn on or to turn off the lights in the virtual scene. The three sliders combination was represented as an "object color changer" to control the color of a sphere in the scene. Each slider had a range from zero to 255, and all together they worked as an RGB color modifier. Specific values of red(R), green(G), and blue(B) were shown in the instruction and the participants had to match the given RGB value combination by moving each slider with their input. Finally, in the six buttons task, the participants were told to sequentially select buttons to a given number order (length of three). The time for each task completion, with the total time of duration was recorded. For the baseline, ray-casting was used for positioning the desired UI element, and a physical button click for selection.

- Navigation. Navigation task was also designed to utilize one hand. In this task, the participants stood still in a static location and moved the first-person perspective 3D player by using controller to solve a small maze in VR scene. There were a starting point and an ending point in the maze, which involved six direction changes to complete. The navigation hints for solving the maze were shown in the VR scene, represented by arrows pointing the right direction. The total time of travel from the start to the end was measured. There was only a forward button for both the baseline and the overlay conditions, and the heading direction was determined by the facing direction of the participants. This is a scenario which the participants do not require a constant visual cue their control devices once they first get used to. This simulates a traditional direction input device such as a joystick or a keyboard with a simple mapping set such as WASD, which are easy-to-reach, and always in user's hand. In this one-dimensional input task, this could be executed by a simple movement of finger by pushing the same button (the baseline: physical trigger button and Overlay: Android touch UI button) repeatedly.

- Text Entry. Text entry task was a two-hand operation, and the participants were asked to type to five stimulus sentences one at a time. A different set of five sentences were randomly drawn for each condition. As the visual feedback to the participants, the response text was shown in the VR scene with the stimulus sentence. The current text view was placed below the stimulus element. For the baseline condition using Vive controllers, a keyboard 3D UI element with QWERTY layout was constructed. And to operate the keyboard, we chose the ray-casting technique. The rays came from both controllers to point the keyboard element and the participants used trigger button to select the highlighted key. For Overlay conditions, we used Google keyboard for Android combing with a custom app delivering text input in real-time via TCP/IP. Automatic auto-correction feature was disabled for this app. The measurements were word per minute (WPM) values. Five consecutive characters were counted as a word, including spaces. Error rate was measured as character error rate (CER), which is the minimum number of character-level deletion, insertion and substitution operators required to match the response text to the stimulus text, divided by the total numbers of characters in the stimulus. Stimulus sentences were acquired from the mobile phrase set [40].

4.2 Conditions

To test and examine the feasibility of our proposed method in terms of usability and performance as an input system against existing VR dedicated controllers, we conducted various experiments in different conditions followed by a variety of input tasks. First, three conditions for the experiment were created as follows:

- The Baseline: The Baseline condition was created by using HTC Vive VR controllers. Participants used either a single or dual controller to complete the experiment, according to the task they were given.
- Overlay Normal: The real image of the detected object (mobile phone) was delivered to the VR environment when detected. Participants then manipulated their mobile device to complete the task using one or two hands. The size of the object shown through the HMD, was set to be perceived as a real-world size.
- Overlay Large: There was no difference in keypoints nor registered patterns from the pattern-matching technique between the Normal and Large conditions. The only difference was the represented scale of the overlaid image to participants.

In the Overlay Normal condition, the size of target object was calibrated to be perceived as a real-world size whereas the Overlay Large condition was set to have 150% scale compared to the previous condition. The reason for supplementing an additional condition in different scale of represented image was to further investigate the influence of the visual image size represented to users in terms of task solving performance, readability and usability. For visual representation for participants, HTC Vive HMD was used for all conditions.

4.3 Participants and procedure

We recruited 15 participants in this study and performed repeated measures for three conditions. The participants were 12 male 3 female, aged between 24 to 35 years. None of them

claimed to be a serious VR HMD user, but they had experienced the virtual vision through the HMD at least once. All participants had zero experience of using VR dedicated hand-held controllers except one. Also, all of them were familiar with modern smartphone usage, and accustomed to using QWERTY desktop keyboard as typing interface. Before the experiment, all participants were briefly told about the operating methods for both controller and overlay object as the VR controller. Then they were asked to wear HMD and stare at a demo VR scene which was irrelevant to this experiment conditions to adjust inter-pupil distance (IPD). Regular HTC Vive controllers and overlay object were shown at the same time in the same scene, but participants were not allowed to interact, just to grasp the visual representations of the two.

The experiments were within-subjects design. Thus, the executing order of the tasks as well as the specific instructions were different in menu selection task, orientation was modified for navigation task, and different sets for stimulus sentences at similar length were selected for the text entry to maintain the balance across all conditions. For all input process including menu selection and typing in across different conditions, the participants were asked to make inputs as quickly and as accurately as possible. For the navigation, the participants were asked to avoid any collision with the wall. After completion of each condition, the participants took five minutes of break while filling out a simulator sickness questionnaire (SSQ) [24] and system usability scale (SUS) [8] as feedback for each condition. At the beginning of each condition, all participants were asked to hold the controllers / mobile phone in their hands.

5 Results

For statistical analysis, we performed Friedman tests with an initial significance level at $\alpha = 0.05$ to find statistically significant differences among three conditions (the baseline, Overlay Normal, and Overlay Large). The post-hoc analysis was then carried out using Wilcoxon Signed Rank Sum test with Bonferroni correction to investigate a statistically significant difference between each condition.

5.1 Measurement data

Figure 5 shows the average task completion time values of the participants for each task as well as the total time of duration from each condition. In the first task (menu selection) with the total time, A Friedman test showed that there was a statistically significant difference among those three conditions, χ^2 (2) = 20.8, $p < 0.001$. The post-hoc Wilcoxon tests showed that the baseline showed significantly lower seconds (median = 59.79, SD = 11.7) for completion of the tasks compared to the Overlay Normal (median = 40.39, SD = 8.79, $z = 3.57$, $p < 0.001$) and Overlay Large (median = 39.71, SD = 10.46, $z = 3.26$, $p < 0.001$). No significant difference was found between the two Overlay conditions.

Coming down to the small tasks in the menu selection task, first noticeable result was shown in the RGB color slider combination task. A significant difference had been found in the task, χ^2 (2) = 20.13, $p < 0.001$. Comparing the baseline (median = 36.84 SD = 8.15) to each Overlay Normal (median = 21.47, SD = 6.94, $z = 3.37$, $p < 0.001$) and Overlay Large (median = 20.46, SD = 5.78, $z = 3.56$, $p < 0.001$), our method showed faster completion time.

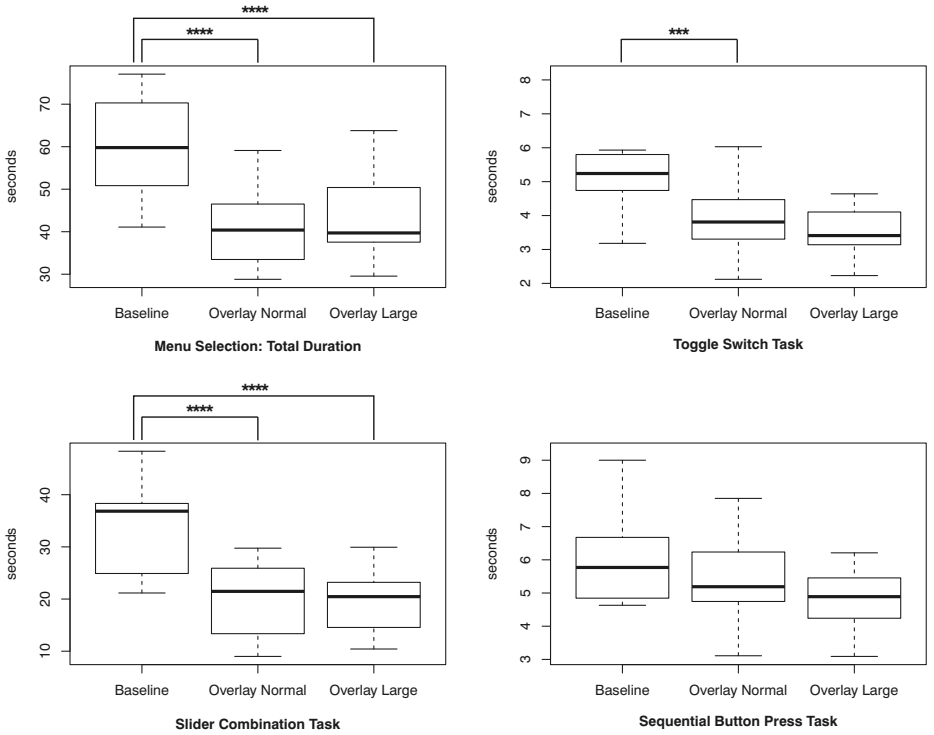

Fig. 5 Menu selection task results among three conditions (* $p < 0.05$ ** $p < 0.01$ ***$p < 0.005$ **** $p < 0.001$)

The results in the first task suggests that our system provides a better task completion performance as the task requires more visual attention and gets more complex. In addition, another difference was also found on the toggle switch task (χ^2 (2) = 12.4, $p < 0.005$). The difference between the baseline and Overlay Normal was reported as significant as $z = 2.69$, $p < 0.005$, whereas other comparisons between the conditions showed no significance. We assume this is due to a high IQR (Inter-quartile Range) found on toggle task of Overlay Normal condition. The median values for the baseline were 5.24 seconds (SD = 1.06), for Overlay Normal was 3.81 seconds (SD = 1.06) and 3.41 seconds (SD = 1.52) for Overlay Large condition.

For the final small task in the first menu selection task which is the sequential button press, there wasn't statistically significant difference reported. The measurements were median at 5.77 seconds (SD = 1.28) for the baseline, 5.19 seconds (SD = 1.58) for Overlay Normal and 4.89 seconds (SD = 1.25) for Overlay Large. The task itself showed that it was too short to discover any difference nor participants reported awareness of difference among three conditions.

In the results of the second task (the maze task), we could not find any statistically significant difference among three conditions. The median completion time of the baseline was 36.01 seconds (SD = 9.12), whereas Overlay Normal condition showed 41.66 seconds (SD = 8.01), and 42.47 seconds (SD = 7.68) for the Overlay Large condition. Once the controller is held in participants' hand, a constant visual information was not necessary to move around the maze. We believe that is the reason for no difference. This is the kind of task

that could be executed without any problems with the VR isolation. The result suggests that our system did not show any advantages over the baseline on the such task that does not require the intervention or interaction with the real world.

The results from the third task, the text entry is shown in Fig. 6. The medians of Word Per Minute (WPM) and Character Error Rate (CER) values recorded from text entry task is shown in each plot. A Friedman test found a statistically significant difference among three conditions of WPM results (χ^2 (2) = 12.93, $p < 0.005$). The median entry rate WPM for the baseline was 9.98 (SD = 1.7), 16.31 (SD = 4.06) for Overlay Normal, and 16.21 (SD = 2.56) for Overlay Large. Post-hoc test found a significant difference between the baseline and Overlay Normal ($z = 2.7$, $p < 0.005$). There was also a significant difference comparing the baseline with Overlay Large ($z = 3.37$, $p < 0.001$) condition. No significant improvement in WPM score was found in Overlay Large condition over Overlay Normal. Even though no significant difference was found on those two conditions, the result imply that the Overlay method in general enables a better text entry performance over the condition with hand-held controllers.

In the results of error rates in the text entry task, another Friedman test was conducted with the data of the error rate measured with CER values. The results also showed that a statistically significant difference exists within three conditions (χ^2 (2) = 6.93, $p < 0.05$). The post-hoc Wilcoxon tests found that there is a significant difference between the baseline and Overlay Normal ($z = 2.69$, $p < 0.005$). The median CER for the baseline was 0.031 (SD = 0.04), 0.1 (SD = 0.04) for Overlay Normal, and 0.08 (SD = 0.05) for Overlay Large. Results suggest that our methods enabled participants to archive a higher WPM score, whereas also had a higher risk to make an error during entry according to the CER values around 8 to 10%. A possible reason to explain this result would be the lack of the depth information of the real image inside the VR scene, which will be discussed in the later section.

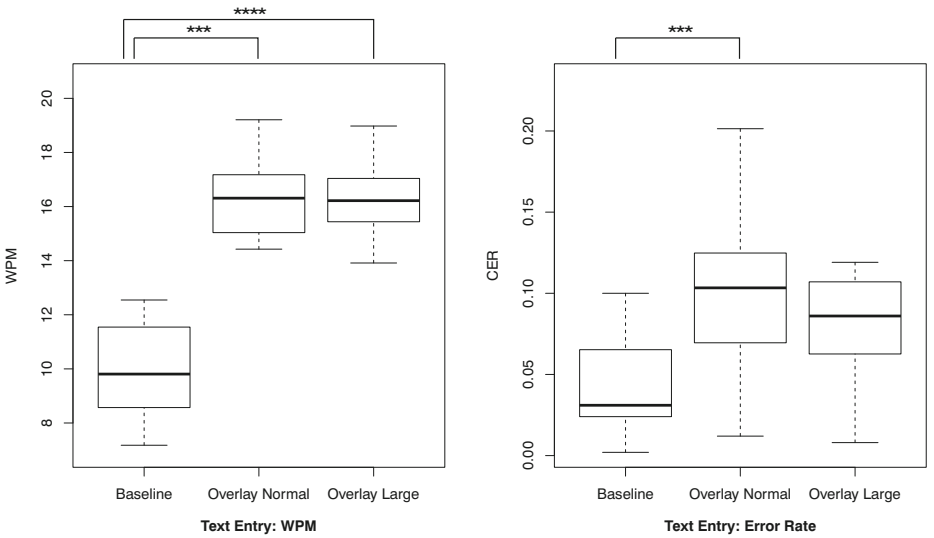

Fig. 6 WPM (left) and CER (right) scores across conditions (* $p < 0.05$ ** $p < 0.01$ ***$p < 0.005$ **** $p < 0.001$)

5.2 User feedback and survey

SSQ and SUS survey data was gathered from the participants from the survey they took on each five minutes break in between experiment conditions. Developed by Kennedy et al., the SSQ consists of 16 symptoms in three distinct clusters including nausea, oculomotor, and disorientation. Participants rated each symptom on a 4-level Likert scale ("None = 0", "Slight = 1", "Moderate = 2" and "Severe = 3). Overall SSQ score was calculated with combing all three clusters with corresponding weights. Only post immersion data was gathered this time. The results are shown fin Fig. 7. It shows that the average total score is significantly lower for our both methods (Overlay Normal: mean = 24.68, SD = 21.52, Overlay Large: mean = 18.94, SD = 24.625) than the baseline with controller (mean = 49.61, SD = 43.87). A Friedman test result shows that a statistically significant difference among SSQ values from three conditions, χ^2 (2) = 18.926, $p < 0.001$. The post-hoc analysis presented that the baseline had significantly higher SSQ total scores compared to Overlay Normal ($z = 2.27$, $p < 0.05$) and to Overlay Large ($z = 2.66$, $p < 0.005$). The same pattern in statistically significant differences existed throughout the subscales of the SSQ.

To compare the conditions in terms of usability, SUS survey was conducted. SUS was created by Brooke and provides a "quick and dirty" reliable tool for measuring the usability. It is consisted with 10 questions with on 5-level Likert Scale from strongly agree to strongly disagree. A Friedman test on the SUS showed that there was a significant different among

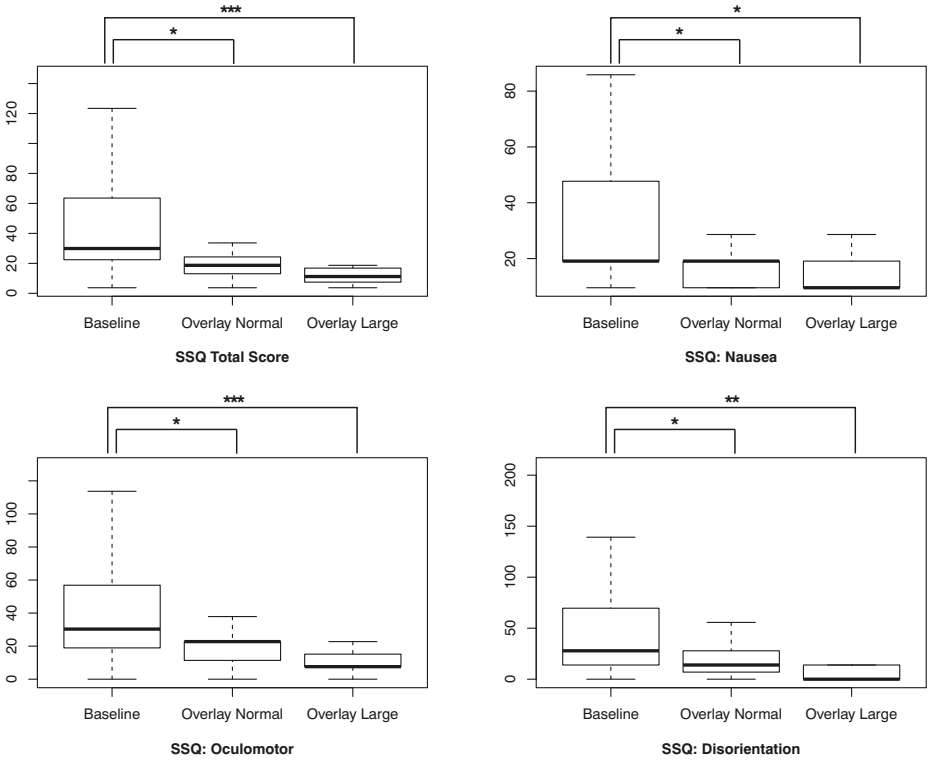

Fig. 7 SSQ Scores among three conditions (* $p < 0.05$ ** $p < 0.01$ ***$p < 0.005$ **** $p < 0.001$)

conditions, χ^2 (2) = 18.429, $p < 0.001$. A Post-hoc test found that comparing the baseline (mean = 44.33, SD = 15.76) with each Overlay Normal (mean = 78.33, SD = 9.33, z = 2.8, $p < 0.005$) and with Overlay Large (mean = 80.17, SD = 11.11, z = 2.75, $p < 0.005$) was significantly different. Based on the research by Brooke, a SUS score of a 68 would be considered as an average. Therefore, the result suggest that our system provides a better usability above average level, with considering all input types we have involved in this experiment.

No participant mentioned about the object detection accuracy. While monitoring data in real-time, the object monitoring session can lose its track caused by a high level of occlusion created in between the target object and the participant's hand, or with a random visual noise at the specific frame. This can cause a sudden disappearance of the target imagery in the VR scene if the track is not immediately resumed within a second, which are consisted of 30 frames. However, we assume this positive feedback was due to a simple filter to maintain the sight (keeping the window open) at the last seen point for three seconds in case when the tracking was lost. This made users to naturally move their holding positions in seldom "lost" scenarios, then the system was able to resume the tracking. Apparently flickering caused by the tracking loss often occurred during the experiment, but survey showed that it had a small to none effect on the overall usability of the system.

6 Adding stereoscopy in real images

After the experiment, we discovered that the current implementation had a limitation when delivering real image to users. The camera we used was not a stereo camera, rather a regular single-lens, RGB webcam. Thus, it was impossible to represent the image with stereoscopy. The result would be overlaying a flat 2D image without depth information onto a stereoscopic virtual rendered image. There wasn't any serious issue reported while just observing the outside world or when manipulating the phone menu UI which had noticeably huge margins between interactable elements. However, when the participants started the text entry task and focused on the software keyboard which had near-to-no margin between the keypads, the issue had been risen. The participants reported that readability was fine, but the notion that there was a difference between the planned finger landing point and the actual landing point, often caused a confusion and a slight difficulty to focus. We assume this is caused by the lack of the depth information of the image, as a mismatch is likely to occur between the visual and motor cortex. The actual landing point of users' fingertips on the screen have high risk of not being aligned as they expected, as there is no depth representation. As some researchers mentioned similar findings in AR study [19], alignment of imagery may be a crucial factor in HMD-based environment when delivering visual information to users, as the precondition of the HMD is that it always provides stereoscopy in virtual imagery by using its two separate screens. Aligning the real visual representation with the virtual information is required to minimize confusion and improve presence while being immersed in virtual reality.

In order to further investigate this issue, we first redesigned the platform with added stereoscopy in real images then conducted a preliminary test in two phases: 1) with or without stereoscopy in real image representation in VR, 2) comparing conditions between stereoscopy and without the HMD ("bare eye" condition).

6.1 Updated image representation with stereoscopy, touch point test

To conduct the comparison test of touch input accuracy and user perception upon different visual imagery representation, we prepared two conditions with the monoscopic and the stereoscopic image representation. In this test, the only variable factor was set to image stereoscopy. The monoscopic condition utilized system setup from the previously developed system with regular single-lens webcam device, whereas the newly updated stereoscopic condition utilized ZED mini stereo camera to deliver offset in real imagery, capturing two streams of images at different angles and each frame simultaneously. The final resolution of represented outside imagery inside the HMD was equally set to 720p (1280 × 720 resolution) which was the same resolution compared to the previous method. The pattern matching method we have utilized in previous sections was disabled to ensure the result was not affected by any delay or flickering from the tracking algorithm. Nor the stereoscopic method utilized any tracking algorithms in any form. As a result, a permanently opened "window" of outside context was shown on top of the virtual elements through the HMD. The adjustments were calibrated prior to the experiment as we found out that the mismatch of IPD variables in two devices may result a distortion of the imported image.

For the task design, we created a UI interface on the Android phone with multiple touch points in both coarse and dense conditions regarding the margins among the touch points. The total size of the canvas on the LCD was 2960 by 1440. Within the whole canvas, the UI on Android device was setup to have 16 touch contact points marked with visual crosshairs in a 4 × 4 grid spaced evenly across 2250 × 1200 pixels in coarse interface and 1250 × 800 pixel in dense interface. The margins among target touch points were 750 pixels for horizontal 400 pixels vertical in the coarse interface, 416 pixels horizontal and 266 pixels vertically on the dense interface. Participants were required to place their fingertip on the surface of the Android device corresponding to the displayed points. The most upper left point was considered as the first point, and the numbers increased in transversal order, as the most lower right point was numbered 16. All participants were requested to touch all 16 points in the UI in a numerical order as accurately as possible. The system recorded all detected actual points of touch contacts with their pixel coordinates including x and y axis, allowing us to see study the accuracy and variations of the result.

We recruited 15 participants for this particular experiment and performed repeated measures. The participants were 11 male 4 female, aged between 25 to 35 years. The experiment was conducted in repeated measures, letting all participants experience both conditions including monoscopic and stereoscopic real image inside the virtual environment. Prior to the experiment, participants were informed with the objectives of the task, and given at least 30 seconds to get used to the environment with the HMD before the initial touch began. Any auditory or haptic feedback upon a keypress from the mobile phone or the HMD device has been disabled to gather raw finger landing locations without any assistance.

As we recorded the x and y pixel coordinates of all points at the moment that that participant pressed the touch screen, a scatterplot is shown in Fig. 8, visualizing all 16 points of two different tasks with the data from 15 participants. 240 points per conditions were gathered from the user test, consisting a total of 960 points.

For the coarse task with larger margins between the touch points, monoscopic touch condition showed an average pixel error (RMSE) of 63.78 pixels on horizontal axis, 33.56 pixels on vertical axis and 50.96 (SD = 4.77) pixels for combined axes. On the other hand,

stereoscopic condition resulted in 55.87 on x axis, 34.53 on y axis and 46.44 (SD = 5.24) for combined. Significant difference was not found between the conditions.

In the results of dense task, monoscopic condition showed an average error of 113.06 on horizontal, 51.56 on vertical axis resulting 87.87 (SD = 13.16) pixels for combined axes. Stereoscopic condition resulted in 62.93 pixels on the x axis, 28.47 pixels for the y axis and 48.84 (SD = 15.01) pixels for both axes. The difference between the conditions showed a statistically significant difference at $p = 0.001$. While the stereoscopic condition maintained consistent accuracy in finger landing points without any statistically significant difference across the coarse and the dense tasks, the monoscopic condition showed an increasing amount of RMS error from coarse to dense interface. Similar behavior was observed in the text entry task from previous section with a high rate of CER. The result proves that our suspicion that the culprit might be from the lack of stereoscopy. The correlation of RMS error values with the different axes including x and y was considered not significant as the actual horizontal length of the canvas was more than twice long compared to the vertical length.

As shown in Fig. 8, the distributions of touch points on the monoscopic condition scatters more on dense task than on coarse task, especially on the far left and far right column. Even though the x and y gap between the crosshairs in dense conditions was far wider than those in soft keyboards, we could observe uneven distributions on those areas. We suspect that this came from the combination of system and psychological factors, meaning that participants' intentions for not to touch the wrong point was added on top of the limitation of the monoscopic visual representation itself.

Overall, the points from the stereoscopic condition relatively maintained consistency in distribution throughout the tasks, whereas monoscopic system could not.

6.2 Phase one: Text entry experiment

From the results of previous experiments, we have observed that the negative effect in user input performance was directly proportional to the level of complexity in user input tasks under the platform without stereoscopic imagery. As the complexity in UI elements regarding the margins between the buttons and pixel density increases, the confusion and errors from users also escalate. Among different tasks from the previous experiments, the results from text entry task particularly showed a noticeable error rate and reported numbers of confusion from the participants as it was considered as the most difficult task. To further investigate and overcome the suspected culprit, we conducted an additional experiment over the touch point test, as we rearranged the text input task from Section 4 in order to reassess the effects of stereoscopy and latency.

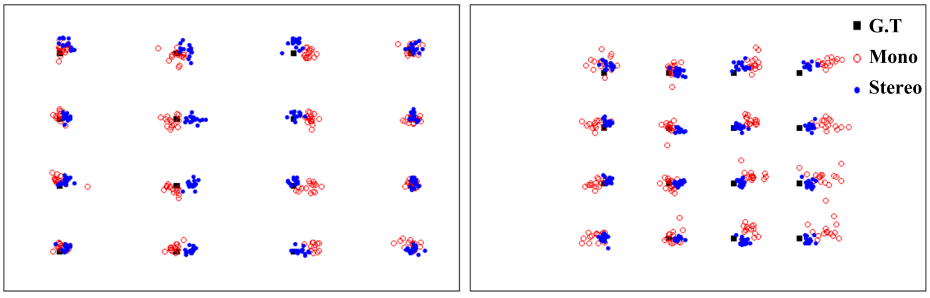

Fig. 8 Visualization of actual touch points from participants in coarse (left) and dense (right) condition

The comparison between the 2D camera condition (Mono Overlay) and stereo camera condition (Stereo Overlay) was set up. The Mono Overlay condition and the Stereo Overlay conditions were created to investigate the effects of stereoscopic real imagery and latency while participants were controlling high-density UI interface. The first condition was set to the same specification from the previous experiment with the single camera and pattern matching algorithm, while the latter condition utilized stereo camera and distance-based object detection. By utilizing the real-time depth map feature of the stereo camera, we used (ZED camera). We roughly set the setting to only represent objects within the range of 45 cm. Figure 9 shows an example of image representation in the Stereo Overlay condition. With this new implementation, additional reduced latency became possible as the measured image display latency was less than 50% than the pattern-matching technique.

In terms of the given task, the text input task was a two-hand operation, as participants were holding the mobile phone in their hands, they were asked to type five different stimulus sentences one at a time. Five sentences were randomly drawn at each trial out of 30 sentences phrase set. The stimuli sentences were shown in the VR environment and the visual feedback for the real-time typing content was shown both in the VR and mobile device. Participants were 8 male and 7 female aged between 21 to 33 years. A total of 15 participants were familiar with using modern smartphones and had at least one experience with the HMD device. For the software keyboard, we chose Google keyboard for Android with a specific keyboard UI skin that has no boundary margin among the keys as shown in Fig. 9. This was set to measure the performance and accuracy of touch points of the system under an extreme circumstance, which requires a high level of precision in control for a proper text input. Additional supported features of software keyboard such as auto-correction and swipe-to-input were all disabled to minimize any interruptions during the experiment. A custom Android application with above-mentioned software keyboard layout was configured to transmit entered inputs to the 3D engine using TCP/IP protocol. Same as before, the measurements were word per minute (WPM). Five consecutive characters were counted as a word, including spaces. Error rate was measured using character error rate (CER), Stimulus sentences were drawn from the mobile phrase set [40]. Participants performed the task for each condition, and they had a five-minute break between the conditions.

Fig. 9 An example of image representation in Stereo Overlay condition

6.3 Phase one: Text entry results

For statistical analysis, we performed Wilcoxon Signed Rank Sum tests with Bonferroni correction to investigate a statically significant difference between each section for two tasks. Figure 10 shows the WPM and CER results from the text entry task. The results from the Stereo Overlay method archived significantly ($p < 0.05$) higher WPM values compared to entry results from the Mono Overlay condition. The Mono condition (median = 16.22, SD = 3.58) showed slower input results to the Stereo condition (median = 25.34, SD = 9.37) in terms of WPM.

In terms of CER, there was also a significantly difference ($p < 0.01$) between the conditions as the Mono condition (median = 0.117, SD = 0.054) showed a higher error rate compared to the Stereo condition (median = 0, SD = 0.034). Comparing the CER value of the Stereo condition with previous Baseline condition in Section 4, it did not show any statistically significant difference.

The most frequently mentioned additional feedback from users during the text input task were about the image representation method and improved latency. With a new image representation method utilizing the depth map, the latter condition could not only provide a better input performance with stereo images but also a faster image delivery compared to the method utilizing the pattern-matching technique. Less confusion was also reported on Stereo Overlay condition, which explains the lower CER values.

The results imply that our new method with an aligned stereoscopy and less latency in image representation shows a more robust, and a safer way of importing an interactable object from the real-world compared to the first implementation.

6.4 Phase two: Comparison with bare eye conditions

As we have compared our newly designed proposed (with stereoscopy vision) vs. previous method (without stereoscopy), for the last investigation we wanted compare how users could manipulate the mobile phone with our new method versus with bare eye condition, meaning that the users are not wearing the HMD. If our proposed method provides enough accuracy/precision and minimizes confusion while controlling the touch interface, then we would not see any significant difference between the two conditions. Two conditions are described below:

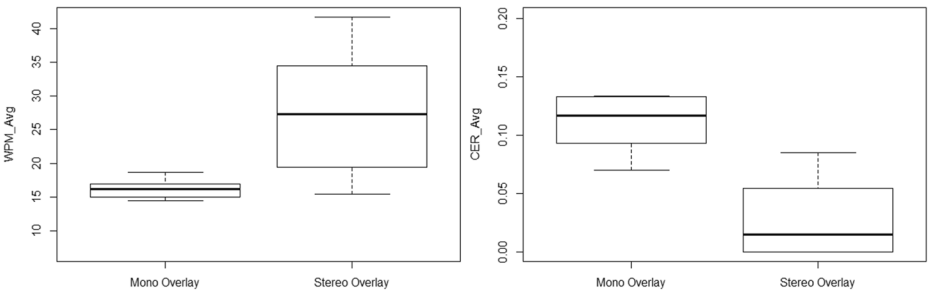

Fig. 10 WPM (left) and CER (right) scores between Mono and Stereo Overlay conditions ($p < 0.05$ and $p < 0.01$)

- Bare-Eye: This condition was set as the baseline of the experiment in this task. In this condition, users were asked to control the mobile device without wearing the HMD and the visual information was shown through a 2D flat screen monitor. The monitor we used in this condition was a 32-in. LCD screen, with a resolution of 1920 × 1080. The monitor was placed in front of the participants, showing the VR content in a full screen mode.
- Stereo Overlay: Stereo Overlay condition was set as the same condition as the Stereo Overlay condition of the previous text input task. The mobile phone was visible inside the VR with its original real-world texture.

The core of gestures for touch commands are divided by the numbers of fingers in usage [41]. According to the Touch Gesture Reference Guide developed by Villamor, there are basic, object-related, navigating and drawing actions in the categories of the touch gesture. In this task design, we chose most used one gesture from the object-related actions, and another from the navigating actions. A total of two selected tasks included the pinch-spread action with two fingers and scroll-select action with a single finger. Thus, two touchscreen specific tasks were set to measure the fluency on operating touchscreen gestures. First involved the pinch-spread interface for zooming in or out and the second utilized scroll actions to select certain section of the interface. The control UI design for the Android device is shown in Fig. 11. The same UI interface was used for both conditions with the HMD and with the "natural" bare-eye condition.

Both gesture tasks were set to have random quiz-answer form. The pinch-spread task was set to modify the scale of a sphere in the VR scene. Random target scale values of the scale were given as quizzes and the participants tried to match the absolute value by either zooming in or out by pinching and spreading two fingers. The range of target scale value was set to 10 to 200, and on the interface part, the initial number was set to 100 for both enabling zooming in and out scenarios staring from the center point. For the UI part on the Android mobile, there wasn't any UI visible element on the screen as the purpose of this interface was only to gather two-finger gesture actions to control elements in the virtual environment. The measurements

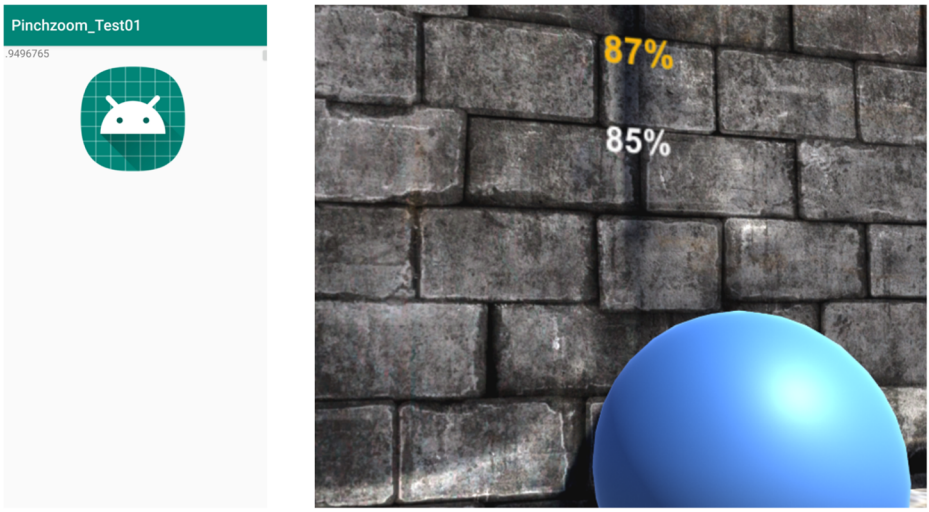

Fig. 11 UI interface (left) and target object in VR (right) of the pinch-spread task

for this task were completion time and error. The error was calculated by comparing the difference between the given target value and the entered value.

For the scroll-select task, random value of year / date / day values in ISO 8601 format (e.g., 1999-03-11) were given as instructions for participants to provide the corresponding answer. Starting from the year select screen, month and day select screen appeared consecutively upon users' final finger tap input as shown in Fig. 12. Participants controlled the touch interface to scroll up and down using their fingertips to select then transmit the corresponding value to the 3D engine.

With the same group of participants from previous text entry test, five trials were given to each participant for each gesture task, entering a total of ten entries for the answers for this section of the experiment. To reduce the risk of familiarity bias, the UI elements on the Android controller were designed not to accept input modifications since the first touch. As the initial touch was started, the answer input session automatically started, and the release motion of finger immediately triggered the closure of the current session and called for the next question. The measurements were operating time and the accuracy. Any values different than the given values were considered as errors.

6.5 Phase two: Comparison with bare eye conditions results

For the results of pinch-spread task we could not find statistically significant difference between the Bare Eye (median = 5.3, SD = 0.57) and Overlay (median = 5.4, SD = 0.28) conditions. The average input response time for the Bare Eye was 5.32 seconds per quiz and 5.48 seconds for Overlay. The percentage error rates for those conditions were 0.4% and 0.38%, respectively. For the scroll select test, the Bare Eye condition showed little better input performance results (median = 4.54, SD = 0.84) than the Overlay condition (median = 4.70, SD = 0.78), but we could not find any significant difference between the results. Since error rate on this task was measured none, we could not compare the error rates from the scroll input.

Although these touchscreen tasks were combinations of simple gesture inputs, users claimed that they did not suffer from confusion or difficulty in terms of providing inputs using hand control. Some participants claimed about the difference in field of view (FoV)

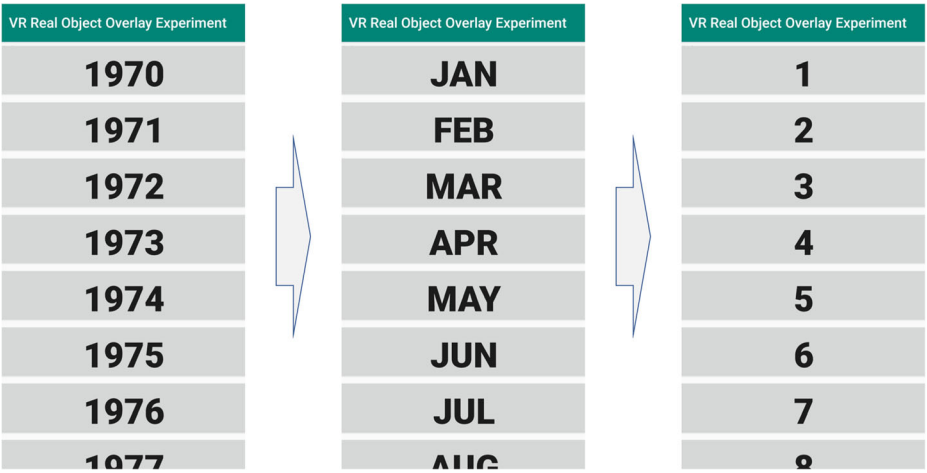

Fig. 12 The sequence of UI interface in scroll-select task

between the bare eye and camera-based vision, but the issue did not affect the result to have statistically significant difference between conditions. Additional feedback from the users included that they could not feel any noticeable inconvenience during the text input, they still could feel a little delay in input feedback but was manageable, which did not affect them to make errors.

7 Discussion and future work

Throughout the experiment, the results demonstrate that our proposed method is capable of delivering a virtual reality experience with integrated mobile phone manipulation support which is located outside of the VR. Not only the method shows the visual representation of the object, but also provides a functionality to use the mobile device as an input interface with a decent level of usability. Even though our approach only incorporated a mobile phone instead of generic items, we still could see the clue that our proposed idea could provide more intuitive and more efficient input experience in virtual environment compared to existing hand-held VR dedicated controllers in certain 2D UI-based tasks. This matches our initial purpose of the study, which is to expand the possibility of input-based interactions in VR. The performance showed better results not only on controlling simple graphic user interface, but also on more sophisticated UIs for typing. Users also reported a lower average level of cybersickness symptoms that could affect the negative aspects of the virtual reality experience. The advantage of the system became more observable in the scenario with a higher level of precision level for control was required. However, the issues found from the first experiment showed that the system began to lack in accuracy when the task became more complex. The good results from the participants varied hugely on a high-density UI in the interactable object which required more precision in input entry. From the results of the first experiment, we observed that the participants had confusion and showed higher error rate when micro-managing the interface with low margin among input boundaries in the UI. We examined the culprit for this issue to discover the underlying problem factor and came to suspect two major factors: stereoscopy and latency. Both factors are a huge factor to define the level of virtual reality experience [21] and combination of slight lack of those together acted as culprit.

The limitation of lack of stereoscopy and latency was further investigated with additional implementation and modification followed by the preliminary accuracy test including the measurement of touch points. The additional text input and touch gesture test were also presented in order to ensure the minimized visual representation of the platform. We first managed to implement stereoscopic image overlay into the HMD by using an external binocular camera and calibrated the image with integration with the external image stream with the 3D engine. In the results of the series of experiments we conducted, we have seen that that modifying the real imagery with stereoscopy to match the type of the virtual imagery represented in the HMD dramatically improves the accuracy in participants' fingertip landing points. And also, we could not find any significant difference between the bare eye and our method while users were giving inputs to the virtual environment.

In terms of the latency problem, due to limitations of the framework we used, we could not maximize the resolution with fastest framerate of the imagery provided to participants. During the implementation we found out that even though ORB is a fast, efficient algorithm for pattern-matching, it did not shine when combined with previously mentioned data pipeline

limitation of Unity 3D and unexpected low performance of the library. The OpenCV wrapper we used for Unity and C# did not support multi-core CPU processing nor GPU computation. As we deployed a PC system with a 16-core CPU and a high-end Nvidia GPU, the computational calculation only relied on utilizing 2 or 3 threads of the CPU. This ineffective computation inevitably led to some delay, and the pipeline problem forced us to stick with low resolution instead of utilizing a full potential.

Thus, our initial attempt for reducing latency was implemented in Section 6 with the stereoscopy issue, as we have switched object detection algorithm from the pattern-matching technique to the distance-based detection using image depth map. Close-by objects within the range of 45 cm were set to appear inside the HMD, and anything other than that was segmented from the real imagery. With additional distance-based object detection shown in the latter text input task comparison, we could see the further potential for the algorithm to be utilized in future for a more realistic target object representation with a reduced level of latency. The utilized method showed a better blended image representation between the real and virtual, and also showed a more robustness to error caused by visual disturbances and obstructed line of sight between the target object and the camera lens.

With the latter implementation including stereoscopy and distance-based object detection, it became possible to create a more robust method for an interaction in VR than the monoscopic system with the pattern-matching technique. The results from the experiments and the reports from the participants were clearly stating that the system showed improvements. Current initial method was set to represent objects in range of 45 cm as the target object was a mobile phone in user's hand in this study, but finding optimal adjustments for the detection range and also background segmentation method development would be further required in order to establish a more solid augmented virtuality platform. Also, according to Abrash's research [1], a latency level of less than 20 ms at motion-to-photon (MTP) is considered as an acceptable level for proper virtual reality experience with a enough level of presence. The term motion-to-photon latency is the time need for a user movement to be fully reflected on the display screen. To meet the level of utilizable MTP latency, we believe that it can be achieved by developing current method of delivering the real image stream from camera, by processing the high-load calculations outside the 3D engine then directly delivering into the virtual environment for efficiency. The experiment results from the latter prototype including the comparison with the bare eye condition, we found out that users barely could feel difference while giving input with our proposed method. This may indicate that the level of the mismatch between the visual representation and the hand manipulation is low, and the AV-based input method could be another available input option for VR scenarios.

Overall, our idea of using mobile device as an input device for VR showed promising possibilities as an alternative input method compared to existing input controllers, with an acceptable usability and minimized visual confusion. We admit that the dedicated 6DOF controllers work flawlessly on certain scenarios even with their complexity and VR isolation, and the tasks we designed mainly focused on 2D-based UIs, which is a natural for a mobile phone but not for VR controllers. However, we believe that our method still could substitute them on some areas where virtual reality needs to consider and recognize real-world context. VR based simulation, education, training and medical applications could be examples of potential candidates. HMD-based virtual reality technology is already widely applied to such scenarios, to take the advantage of synthetic background of virtual environment as building real environments for such applications can be expensive and often dangerous [6]. However, as direct physical object manipulation is not supported on recent HMD platforms, indirect

manipulation methods such as using proxy menu interfaces and involving proxy objects are used in current applications. To provide a precise and intuitive control for learning purposes, those proxy controllers work as barriers for natural interaction with objects, affecting negatively on presence and learnability.

Although our implementations only focused on a mobile phone for VR input interface in this phase of the study, same object tracking algorithms could be applied to other generic objects. Expanding the object range as well as discovering new measures and tasks for platform evaluation would be dedicated to future work of this study. We hope to explore further possibilities of the proposed method for utilization in the area of alternate reality world, where seamless integration between the reality and the virtual environment becomes possible with a full immersion of five senses.

8 Conclusion

In this study, we investigated and evaluated our proposed method for an alternate input interface in virtual environment, by importing a mobile phone from the outside world using Augmented Virtuality. The device was used as a medium to interconnect between the reality and virtual environment, and was utilized as a more familiar, and a simpler input device for the HMD-based VR platform. Throughout experiments we discovered that it shows decent results in terms of intuitiveness and usability, leading to a potential tool for establishing the interactive virtual reality platform with the form of augmented virtuality. Modifications to the prototype implementation have been applied in order to improve the usability of the system. The experimental results indicate the potentials of the system as a promising AV-based alternative input interaction method for VR with minimized sense of difference or user confusion.

Future work of this study will be dedicated to establishing a more stable and accurate platform that seamlessly integrates the reality and virtual environment with generic object support, allowing users to break out from the virtual isolation and experience more stimulating world in the mixed reality. By redesigning and reimplementing considering remaining issues that we addressed in the study and further investigate the aspects of human perception regarding important factors of virtual worlds such as presence, we believe that the method could deliver a great level of contribution to the growing research area of wide spectrum of mixed reality, especially on augmented virtuality.

Declarations

Competing interests The authors have declared that no competing interests exist.

References

1. Abrash M (2014) What VR could, should, and almost certainly will be within two years. Steam Dev Days, Seattle, 4, 2014
2. Alaee G, Deasi AP, Peña-Castillo L, et al (2018) A user study on augmented Virtuality using depth sensing cameras for near-range awareness in immersive VR. IEEE VR's 4th workshop on everyday virtual reality (WEVR 2018) 10
3. Alcantarilla PF, Solutions T (2011) Fast explicit diffusion for accelerated features in nonlinear scale spaces. IEEE Trans Patt Anal Mach Intell 34(7):1281–1298

4. Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-up robust features (SURF). Comput Vis Image Underst 110:346–359
5. Bhowmick S, Sorathia K (2018) Report: state of the art seminar explorations on body-gesture based object selection on HMD based VR interfaces for dense and occluded dense virtual environments
6. Blümel E, Haase T (2009) Virtual reality platforms for education and training in industry. In: East European conference on advances in databases and information systems. Springer, Berlin, pp 1–7
7. Bolton J, Lambert M, Lirette D, Unsworth B (2014) PaperDude: a virtual reality cycling exergame. CHI '14 extended abstracts on human factors in computing systems 475–478
8. Brook J (1996) A quick and dirty usability scale. Usability evaluation in industry
9. Burdea GC, Coiffet P (2003) Virtual reality technology. John Wiley & Sons
10. Calonder M, Lepetit V, Strecha C, Fua P (2010) Brief: binary robust independent elementary features. In: European conference on computer vision. pp. 778–792
11. Calvin J, Dickens A, Gaines B, et al (1993) The SIMNET virtual world architecture. In: Proceedings of IEEE Virtual Reality Annual International Symposium. pp. 450–455
12. Carruth DW, Hudson CR, Bethel CL, et al (2019) Using HMD for immersive training of voice-based operation of small unmanned ground vehicles
13. Caserman P, Garcia-Agundez A, Konrad R et al (2018) Real-time body tracking in virtual reality using a vive tracker. Virtual Reality 23:155–168
14. Casterson S (2016) Htc vive: a guide for beginners. Conceptual Kings
15. Desai K, Raghuraman S, Jin R, Prabhakaran B (2017) QoE studies on interactive 3D tele-immersion. In: Proceedings - 2017 IEEE international symposium on multimedia, ISM 2017
16. Desai AP, Pena-Castillo L, Meruvia-Pastor O (2018) A window to your smartphone: exploring interaction and communication in immersive VR with augmented Virtuality. Proceedings - 2017 14th conference on computer and robot vision, CRV 2017 2018-Janua:217–224. https://doi.org/10.1109/CRV.2017.16
17. Fisher SS, McGreevy M, Humphries J, Robinett W (2004) Virtual environment display system. 77–87. https://doi.org/10.1145/319120.319127
18. Friston S, Steed A (2014) Measuring latency in virtual environments. IEEE Trans Vis Comput Graph 20: 616–625. https://doi.org/10.1109/TVCG.2014.30
19. Hou M (2001) User experience with alignment of real and virtual objects in a stereoscopic augmented reality interface. In: Proceedings of the 2001 conference of the Centre for Advanced Studies on collaborative research. P 6
20. Hudson CR, Bethel CL, Carruth DW, et al (2018) Implementation of a speech enabled virtual reality training tool for human-robot interaction. In: DISA 2018 - IEEE world symposium on digital intelligence for systems and machines, proceedings
21. Jerald J (2015) The VR book: human-centered design for virtual reality. Morgan & Claypool
22. Jiang F, Yang X, Feng L (2016) Real-time full-body motion reconstruction and recognition for off-the-shelf VR devices. In: Proceedings of the 15th ACM SIGGRAPH conference on virtual-reality continuum and its applications in industry, vol 1. pp 309–318
23. Kasahara S, Konno K, Owaki R, et al (2017) Malleable embodiment: changing sense of embodiment by spatial-temporal deformation of virtual human body. In: Conference on Human Factors in Computing Systems - Proceedings
24. Kennedy RS, Lane NE, Berbaum KS, Lilienthal MG (1993) Simulator sickness questionnaire: an enhanced method for quantifying simulator sickness. Int J Aviat Psychol 3:203–220
25. Koike H, Kobayashi Y, Sato Y (2001) Integrating paper and digital information on EnhancedDesk: a method for Realtime finger tracking on an augmented desk system. ACM Transactions on Computer-Human Interaction. https://doi.org/10.1145/504704.504706
26. Latoschik ME, Lugrin JL, Habel M, et al (2016) Breaking bad behavior: immersive training of class room management. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST
27. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60:91–110
28. Maggioni C, Ring OH (1993) Gestural input device for virtual reality. Image Processing:118–124
29. Mcgill M, Boland D, Murray-Smith R, Brewster S (2015) A dose of reality: overcoming usability challenges in VR head-mounted displays. Proceedings of the 33rd annual ACM conference on human factors in computing systems - CHI '15 2143–2152. https://doi.org/10.1145/2702123.2702382
30. Milgram P, Kishino F (1994) A taxonomy of mixed reality visual displays. IEICE Trans Inf Syst 77:1321–1329
31. Piumsomboon T, Lee G, Lindeman RW, Billinghurst M (2017) Exploring natural eye-gaze-based interaction for immersive virtual reality. 2017 IEEE symposium on 3D user interfaces, 3DUI 2017 - proceedings 36–39. https://doi.org/10.1109/3DUI.2017.7893315

32. Poupyrev I, Billinghurst M, Weghorst S, Ichikawa T (1996) The go-go interaction technique: non-linear mapping for direct manipulation in VR. In: ACM Symposium on User Interface Software and Technology. pp. 79–80
33. Ren Z, Meng J, Yuan J (2011) Depth camera based hand gesture recognition and its applications in human-computer-interaction. ICICS 2011 - 8th international conference on information, communications and signal processing 1–5. https://doi.org/10.1109/ICICS.2011.6173545
34. Rosten E, Drummond T (2006) Machine learning for high-speed corner detection. In: European conference on computer vision. pp. 430–443
35. Rublee E, Rabaud V, Konolige K, Bradski GR (2011) ORB: an efficient alternative to SIFT or SURF. In: ICCV. p 2
36. Satyavolu S, Bruder G, Willemsen P, Steinicke F (2012) Analysis of IR-based virtual reality tracking using multiple Kinects. In: Proceedings - IEEE Virtual Reality. pp. 149–150
37. Stanney KM (2002) Handbook of virtual environments: design, implementation, and applications
38. Steed A (2008) A simple method for estimating the latency of interactive, real-time graphics simulations. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST
39. Tecchia F, Avveduto G, Brondi R, et al (2014) I'm in VR! 73–76. https://doi.org/10.1145/2671015.2671123
40. Vertanen K, Kristensson PO (2011) A versatile dataset for text entry evaluations based on genuine mobile emails 295. https://doi.org/10.1145/2037373.2037418
41. Villamor C, Willis D, Wroblewski L (2010) Touch gesture reference guide. Touch Gesture Reference Guide
42. Wachs JP, Kölsch M, Stern H, Edan Y (2011) Vision-based hand-gesture applications. Commun ACM 54(2):60–71
43. Wei D, Zhou SZ, Xie D (2010) MTMR: a conceptual interior design framework integrating mixed reality with the multi-touch tabletop interface. In: 2010 IEEE international symposium on mixed and augmented reality. IEEE, pp 279–280
44. Xiao R, Schwarz J, Throm N, Wilson AD, Benko H (2018) MRTouch: adding touch input to head-mounted mixed reality. IEEE Trans Vis Comput Graph 24(4):1653–1660
45. Zhou F, Duh HB-L, Billinghurst M (2008) Trends in augmented reality tracking, interaction and display: a review of ten years of ISMAR. In: Proceedings of the 7th IEEE/ACM international symposium on mixed and augmented reality. Pp 193–202