2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW) | 978-1-6654-4057-8/20/$31.00 ©2021 IEEE | DOI: 10.1109/VRW52623.2021.00045

Using a 6 Degrees of Freedom Virtual Reality Input Device With An Augmented Reality Headset In A Collaborative Environment

Adam S. Williams*
Colorado State University

Francisco R. Ortega†
Colorado State University

Abstract

Augmented reality headsets have become increasingly consumer-available. Often gesture and speech are the main input modalities provided by these headsets. For some tasks, users may need a more precise input method. Tracked controllers can be added by using image tracking; however, this is not always the most accurate solution. This work outlines how to use off-the-shelf products to create a collaborative cross-device mixed reality experience. In that experience, the positionally tracked inputs from one headset can be used by another headset that may not natively support them.

Index Terms: Human-centered computing—Human computer interaction (HCI)—Interaction devices—Haptic devices; Human-centered computing—Human computer interaction (HCI)—Interaction devices—Pointing devices; Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Collaborative interaction; Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Mixed / augmented reality; Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Virtual reality;

1 Introduction

Augmented reality (AR) head-mounted displays (HMDs) are becoming increasingly popular, as products such as the Microsoft HoloLens 2 [1] (HL2) becoming more readily available. This trend is exhibited in the United States governments purchase of 100,000 HL2 units for military use [4]. AR-HMDs are being used in an increasingly wide span of research, including work on situated analytics [6], multimodal input elicitation [14], and city planing [12]. Some consumer-available AR-HMDs (i.e., the Magic Leap 1 [2]) ship with a 6 degrees-of-freedom (6DoF) controller; however, others do not (i.e., HL2). The HL2 ships with mid-air gesture, speech, and gaze as its primary interaction methods. It's predecessor the HoloLens 1 (HL1) shipped with those same inputs with the addition of a single button clicker that was not tracked [3].

Access to the mid-air gestures used by these devices is often included in the software development kit (SDK) provided for each device. As an example, the Magic Leap 1 included 8 pre-defined gestures in the Lumin SDK. Even with these gestures being easily accessible to developers when used they require the user's hands to be tracked by the device's cameras. The tracking space provided by these cameras is often limited [13], which can make certain interactions difficult. Furthermore, gestures are not always the most precise interaction method for selection or manipulation tasks [3].

In some use-cases, the addition of a tracked mid-air controller would be beneficial to both user immersion and interactions. In AR adding a tracked input is commonly done by using image tracking targets that are attached to emulated pen [11]. The downside to this solution is that the pen can only be tracked while it is in range of the headset's front-facing cameras and while the image tracking targets are visible to that camera. This type of tracking may cause drift or errors where the image tracked input is not well calibrated to the actual location of the physical input. A second, more problematic issue is that this input can only be used while it is held in front of or within the tracking area of the AR-HMD. This can be limiting to users, especially so when the added input is a pen, where users may wish to write notes on a surface in front of them while looking at another location. Consider taking notes in a lecture while looking at the whiteboard.

This limitation may not be detrimental for all use cases. For web-browsing, streaming videos, and simple games, the combination of gesture and speech may provide plenty of interaction technique options. Other use cases, such as the selection and manipulation of nodes on a scatter plot in an immersive analytics environment, may require a more robust tracked input solution. This necessity is even more prevalent in applications that require a user's gaze to be in one location while they hold, or manipulate an object at a different location. As an example, if an analyst is writing notes while viewing a complex data structure in an AR-HMD they may wish to be able to write on a surface in front of them while glancing back and forth from that surface to the data representation they are analyzing. In this case, a camera tracked input would lose tracking when the user looks away from their hand (e.g., the image tracked input).

The main contributions of this solution are:

- A networked cross-device environment for standalone use, co-located collaboration, and remote collaboration
- The ability to use VR controllers with the HL2 or other MR devices
- Easy integration of less standard VR inputs such as the Logitech VR-Pen [4]

2 System Design

There are various commercially available AR-HMDs on the market, with a reasonable market share held by the Microsoft HL2. The HL2 is developed to run using the Universal Windows Platform (UWP) and the Mixed Reality Toolkit (MRTK) which makes integrating Steam virtual reality devices difficult due to their use of a different MR toolkit (i.e., Steam VR) and different tracking systems (e.g., base-stations compared to on device tracking).

To remedy these issues we present an input solution using the Unity 3-dimensional (3D) development engine. This solution allows the use of any 6DoF VR base-station tracked inputs to be used with the HL2 or other MR-HMDs.

At a high level, this solution uses a multi-user client-server architecture to network various devices into the same virtual experience. The positions of these devices are tracked and centered relative to a fixed physical location. This allows synchronous co-located collaborators to view the same virtual environment in real-time. All

*e-mail: AdamWil@colostate.edu
†e-mail: F.Ortega@colostate.edu

[1]www.microsoft.com/en-us/hololens
[2]www.magicleap.com/en-us
[3]www.docs.microsoft.com/en-us/hololens/hololens1-clicker
[4]https://www.logitech.com/en-us/promo/vr-ink.html

connected HMDs can view and interact with the same virtual content; however, the world synchronization step differs by device. The instance running on the VR-HMD with the desired inputs can be connected and left on, while the controllers for that VR-HMD are given to the AR-HMD user, which in this project was a HL2. This allows the HL2 user to utilize the 6DoF input as tracked by the VR-HMD's base stations (e.g., infrared tracking stations). The VR-HMD used in this project was the HTC Vive-Pro [5].

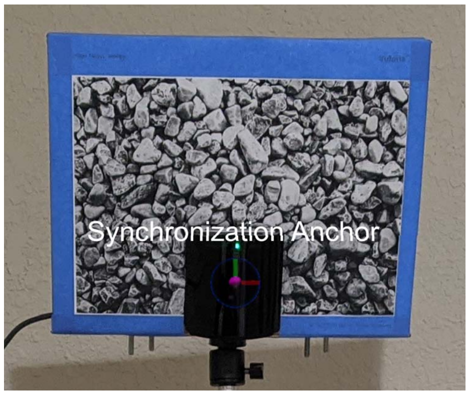

Figure 1: Vuforia image target mounted on top of Vive-Pro Base Station 2.0, synchronization anchor position adjusted down from the center of the image target to the base station center.

Synchronization is achieved on the AR-HMD by using Vuforia image tracking to align a virtual anchor with a real-world location. This anchor is referred to as the synchronization anchor, which is an empty game object. This anchor and its corresponding Vuforia image target are shown in Figure 1. VR-HMDs place their synchronization anchor at this real-world location by using one of the trackers provided by the device (the black object in Figure 1). This project used one of the base-stations provided by an HTC Vive-Pro. GameObjects can be parented in that synchronization anchor and have their locations relative to that anchor synchronized between devices.

Networking the devices adds a need to share the object locations over the network, which can add some latency. The benefit of this approach is that the networked solution provides an expandable synchronized collaborative environment, where more than one user may log in from different MR devices.

3 Related Work

When looking at input devices for object selection and manipulation in MR, work has found that mid-air pen outperforms standard vive controllers in terms of speed and user preference [8]. There is also evidence that pen-like input devices can outperform finger-pointing (i.e., gesture) inputs [3], and that controller based hand-tracking has outperformed the mouse in some 3D positioning tasks [9]. That controller and pen based inputs show promise in MR environments motivates the use of a mid-air pen or other tracked controller with an AR headset.

In 2017 Bai et al. outlined an approach for using a Vive VR-HMD 6DoF controller with an HL1 AR-HMD [2]. That solution used image tracking to synchronize the HL1 with a fixed world position. The Vive HMD was also similarly synchronized by using one of the Vive controllers placed at the same location as the HL1 tracked image in the real-world. Bai et al. then used an off the shelf solution for Bluetooth networking to transmit the Vive controller coordinates and information to the HL1 [2]. This work differs in a few major ways. First, the devices and software used are different simply due to the time that has passed since the publication of that paper. That difference also leads to the necessity of the shared anchor. The older solution was able to move the world origin to synchronize virtual content. The most important difference is that the previous solution is for a single device where this solution can allow several devices to connect to the same environment. The basic implementation of this solution allows 20 clients. Some minor modifications can be made to the server (i.e., local hosting) to allow up to 100 clients to join the same experience.

Outside of using tracked controllers, other work has used image tracking along with a tangible object, such as a 3D printed pen [11]. While those image tracked solutions are viable, they can lack precision and the ability to be used when the images are not in view. Additionally, most tracked controllers can provide haptic feedback which may be desirable for some projects. Another solution for mapping a 3D model in a VR environment to a real-world environment is to find three or more real-world locations and to use the VR-HMD's controllers to locate and record those points, after which the system can adjust the alignment of those same three points in the virtual model to those points in the real-world [7] [6]. A similar approach could be used here where more than one image target is used to synchronize the devices which would help to reduce the calibration step used here.

4 System Components and Integration

This section outlines how to integrate the various components needed for this solution first. Later more detail is provided on the setup steps for each component. This solution uses the following off the shelf components: Unity (version 2019.3.1f LST), Photon Networking version 2 (Photon PUN 2), Vuforia image tracking, the MRTK (version 2.5), a Microsoft HoloLens 2, an HTC Vive-Pro with base stations and controllers, and a Logitech VR-Pen.

4.1 Unity

The first step to setting up this solution is to install the Unity game engine. Unity is a game development engine that supports development for most MR-HMDs. This work uses the Unity version 2019.3[7]). Unity may be downloaded at `https://unity3d.com/get-unity/download`.

4.2 Mixed Reality Toolkit

This project uses the MRTK version 2.5 [8]. The MRTK is a multi-platform mixed reality SDK that is compatible with the HL2, Windows mixed reality, open VR, and most consumer-available MR devices. The recommended way to integrate the MRTK into the Unity project is to add the required packages to the project manifest file. This process is further detailed in the MRTK documentation[9].

4.3 Photon Engine

Next, the networking software needs to be added. This project uses Photon engine 2 [10]. Photon engine is a Unity compatible multiplayer networking system that offers both free and paid usage options. The free solution offers access for up to 20 networked devices, or 100 networked devices if hosted on your own server. To add Photon to

[5]`https://www.vive.com/eu/product/vive-pro/`

[6]`https://github.com/felixkosmalla/unity-vive-reality-mapper`

[7]https://unity.com/releases/2019-3

[8]`https://microsoft.github.io/MixedRealityToolkit-Unity/`

[9]`https://microsoft.github.io/MixedRealityToolkit-Unity/version/releases/2.5.0/Documentation/usingupm.html`

[10]`https://www.photonengine.com/`

the Unity project go to the asset store and then search for and import "PUN 2 - aFree" [11].

4.4 Image Tracking

This step is not necessary unless the project uses devices without provided trackers (i.e., HL2). A simple to use image tracking solution is provided by Vuforia engine [12]. Vuforia is a free to use image tracking system that offers several types of image tracking options. This project will only use the basic image tracking capabilities. In Unity 2019 Vuforia can be added to a project in the "Project Settings →Player" section. To add it check the "Vuforia Augmented Reality Supported" check box. Note that Vuforia is not supported in "windows standalone builds", which are often used when deploying to VR-HMDs. If a VR-HMD is used, the location of a provided device tracker can be used in-place of Vufoira. This project uses the location of one of the provided Vive base-stations.

5 System Setup

This section provides steps for how to use the above-outlined components in unison towards the goal of creating a synchronous collaborative cross-device experience.

5.1 Networking Setup

First, set up the project to start a networked game instance. Once an instance is initiated add the connected clients as players by using Photon engine's "instantiate" function. This will require setting up a networked lobby, room, and player. An overview of how these can be set up can be found in the Photon Unity tutorial [13]. Other details more specific to using Photon with the MRTK can be found in the multi-user MRTK tutorial[14]. When tested on a residential network the latency encountered when synchronizing objects was around 78.6 milliseconds (ms) when averaged over 1000 updates. Prior work using 2D selection tasks found that this level of latency can cause minor (e.g., 15% performance decrease at 40 ms, 50% performance decrease at 225 ms) performance losses; however, those results were not directly extended into a 3D environment [10].

5.2 Coordinate Synchronization

Often the world origin (e.g., world-space coordinate 0,0,0) for AR-HMDs the origin is set based on where the HMD is turned on or where the HMD was when the app was started. VR-HMDs may have a world origin set in the same way or set based on a location set up in the "set up playspace" step of configuring the headset. Changing this location is not an optimal solution for synchronization and is discouraged by the MRTK [1, Section 4]. A better solution is to use a Unity GameObject with its location set to the desired world anchor (i.e., the synchronization anchor). Any networked objects that need to be synchronized can be parented in that anchor GameObject. While this type of object synchronization is not provided by Photon by default, it can be added with relatively low effort. When synchronized objects are the local instance they will need to send their transform information to their networked corollaries. When these networked instances receive that information they can be set to update their position accordingly.

The steps for placing a shared anchor in the appropriate real-world location differs based on the type of devices being used. These differences fall into two major categories: devices with image tracking and devices with position trackers. Some part of the decision to use one method over the other may be influenced by the build target used. When using the HL2 or other Windows MR devices the build should be set to UWP. When using Steam VR based or other VR based devices the build should be set to "PC, Mac, & Linux Standalone".

The two different build targets have access to a different set of functionalities in the Unity engine application programming interfaces (API). These differences can be seen in the Unity documentation under "unityEngine.XR". For the scope of this paper, the most important difference in functionality is that the UWP does not have access to base-station locations. We recommend using Unity's "Platform Dependent Compilation symbols" [15] if the project will be run from both build targets. These allow the specification of which parts of code get compiled for which build targets.

5.2.1 Synchronization of Devices With Trackers

Unity XR nodes and device trackers can be used to set the location of the real-world aligned synchronization anchor. This is most easily done by accessing the information of the tracked device through its unity XR node and centering the synchronization anchor on it. For this solution, one of the two base-stations used by the HTC Vive-Pro is used; however, the use of a controller or other tracker can be similarly effective [2]. Within the scope of this project, no interference between the HL2 tracking and Vive Pro 2.0 base stations was noticed. More information on how to locate XR tracked objects in Unity is listed under "XRNodeStates" in the Unity documentation[16].

5.2.2 Image Tracking Based Synchronization

Vuforia should be used with devices that have image tracking capabilities but no positional trackers. Vuforia requires that an image target is set up before tracking is started. This process is outlined in their tutorial'[17]. Once the image is registered for tracking with Vuforia it can be printed and affixed to the desired position of the real-world location for the synchronization anchor. In this project, that location was on-top of the Vive base station. This setup is shown in Figure 1.

This project only uses Vuforia once, to locate the base station image target. Once the target has been found, Vuforia is no longer needed. To conserve computational overhead we recommend manually enabling and disabling Vuforia so that it is only active once. Allowing manual enabling on Vuforia can be done by setting Vuforia to delayed initialization. The option for that is option is located in the Vuforia settings. For this project, we used platform dependent compilation symbols to enable Vuforia when an AR-HMD was connected and the appropriate scene (i.e., level) was loaded. Vuforia was then disabled upon manual acceptance of the identified image target. If a lot of headset movement is expected, it may be best to leave Vuforia tracking on or to set up an MRTK spatial-anchor to continually adjust the location of the anchor in case of tracking drift.

The accuracy of the tracked controller inputs depends on the accuracy of the image target and its tracking. We have found that a full page size image provides a reasonably easy to find target. The placed anchor can drift when the headset is used too far from the placed anchor or when the task involves looking around. This is caused by minor changes in the location of the spatial mapping that the HoloLens uses to track the location of the target. When a lot of headset movement is expected we recommend keeping Vuforia on to allow re-centering the synchronization anchor. An alternative approach would be to use the azure spatial anchors that are provided by the MRTK to maintain the synchronization anchor's position.

[11]https://assetstore.unity.com/packages/tools/network/pun-2-free-119922

[12]https://developer.vuforia.com/

[13]https://doc.photonengine.com/en-us/pun/v2/demos-and-tutorials/pun-basics-tutorial

[14]https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/tutorials

[15]https://docs.unity3d.com/Manual/PlatformDependentCompilation.html

[16]https://docs.unity3d.com/ScriptReference/XR.InputTracking.html

[17]https://library.vuforia.com/articles/Training/getting-started-with-vuforia-in-unity.html

5.2.3 Cross-Device World Synchronization

When both image targets and the device's positional trackers are used for world alignment, some minor positional adjustments may need to be made. For this project, the position of the image target centered synchronization anchor has been adjusted down a fixed amount to accommodate for the difference in height between the base-station tracked location, and the image target affixed to the top of the base station. Visualizations of the controllers and the connected HMDs are shown in the experience to make the adjustment process easier between devices. These models are shown as faint blue outlines with transparent gray bodies for the controllers and as a 3-axis GameObject with a user or device name displayed above it for connected headsets. These models are shown in Figure 2.

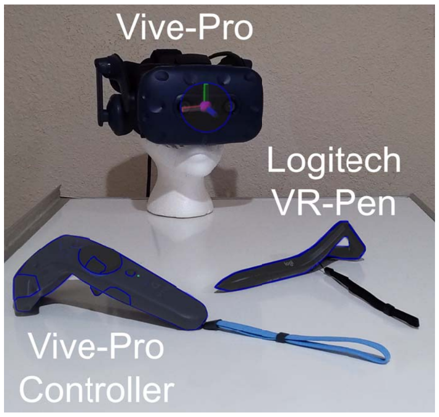

Figure 2: Vive-Pro client and controllers as viewed through the HoloLens 2.
Legend: The faint blue / gray controller outlines are the rendered controller representations to visualize cross-device synchronization, the controller labels are not part of the system, the Vive-Pro label is enlarged for legibility.

5.2.4 Tracking Adjustments

Photon engine provides object synchronization natively. However, due to the implementation of the world aligned anchor, the Photon provided synchronization will not work. Custom photon information streams can be set up to achieve cross-device synchronization. This process is briefly described in the MRTK multi-user tutorial [18]. This tracking will either send the object's transform information relative to the synchronization anchor (e.g., the objects local transform), or it will receive that information and adjust its own transform.

Note that as of the publication of this guide (2021), some users may have issues where the devices will connect to Photon but are unable to join the same instance of the networked environment. The solution is currently to uninstall the Windows SDK version 10.0.19041.0 and to use version 10.0.18362.0.

5.3 Controller Usage

At this point, the project can synchronize content across devices. To use the 6DoF tracked controller with an MR-HMD that does not natively support it, both headsets need to be connected to the same instance of the running Unity application. When the headset that owns the desired controller is connected it will need to register its controllers with the application and then parent their virtual representations in the synchronization anchor. The controllers can then be used by the person wearing the other HMD. The inputs and positions from the controllers are still tracked by their HMD; however, as everything is synchronized they can now be accurately used by any party present in the same physical space.

[18]`https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/tutorials/`

6 Mid-Air Pen Use-Case

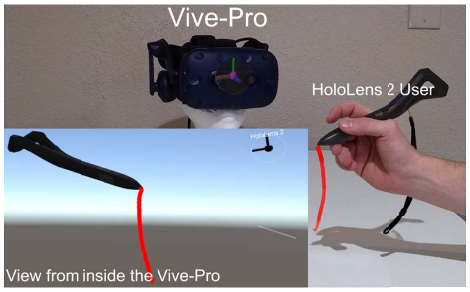

Figure 3: A HoloLens 2 user drawing a line using the VR-Pen as seen by both the HoloLens 2 user and the Vive-Pro user
Legend: Bottom left: view from inside the Vive-Pro with the HoloLens user shown in the upper right, outside of the Vive-Pro view: the HoloLens 2 user's view

To demonstrate the cross-device functionality of this solution we have implemented the Logitech VR-Pen[19] for basic line drawing on the HL2. An example of this functionality is shown from both the Vive-Pro user's and the HL2 user's viewpoint in Figure 3. This was done by setting up the project according to the above-mentioned specifications.

After that setup Logitech VR-Pen can be accurately tracked with its position being nearly identical in the Vive and the HL2. To ensure that this tracking is functional while using an AR-HMD, a virtual model of the pen is rendered over the physical pen on each client (Figure 2). When the tracking is not aligned properly this model will appear out of place compared to the physical pen (Figure 4).

To enable the pen to draw, and to have that drawing seen in real-time on both devices, Unity's "Line Renderer" class is used along with a separate class for handling synchronization. The line renderer class creates a line that connects a provided set of coordinate locations. To render the line on each display as points are recorded those points must also be sent and added to each networked instance of the line render component. This can be achieved by sending the points as part of the data stream that includes the local position of the pen. In this project, all of the point and location transmissions are handled by a separate synchronization class attached to the line render component. The line drawn can be seen from both the Vive and HL2 user's viewpoint in Figure 3 and from the HL2 user's when not properly synchronized in Figure 4.

For this project, the Vive-Pro headset remains usable with a single 6DoF controller. The HL2 user can use the VR-Pen, which is the Vive HMD's second controller.

7 Discussion

This solution outlines how to integrate and modify several free resources to create a collaborative MR experience and to allow the use of controllers across devices. The steps used to align the devices' world-locations is only relevant when setting up co-located collaborative environments. When creating remote collaborative environments the synchronization provided by Photon will be enough. That said, the use of the custom information streams may still be beneficial as seen in the case of adding line renderer points when drawing a line in real-time. This solution is robust enough to handle a wide set of needs by accommodating both image tracker and

[19]`https://www.logitech.com/en-us/promo/vr-ink.html`

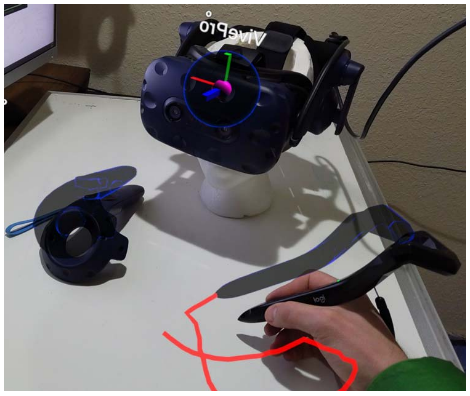

Figure 4: Poorly synchronized Vive-Pro client and controllers as viewed through the HoloLens 2.
Legend: The faint blue / gray controller outlines are the rendered controller representations to visualize cross-device synchronization, the red is the line drawn using the VR-Pen.

positional tracker synchronizations as well as a selection of MR devices.

It is reasonable to assume that as these devices mature, the tracked area provided by them will improve. While the exact specifications of most AR-HMDs tracking and viewing areas are difficult to find, Microsoft has mentioned that the HL1 had a field of view (FoV) of 34-degrees which increased to 52-degrees in the HL2 [5]. Note that the FoV is different from the area tracked by the device. Even so, a similar trend of improving the tracked area in each device iteration can be expected. Yet, as of now, these devices offer limited tracking, and other AR-HMD product releases in the near future may include even less tracking as necessitated by a smaller form factor (i.e., Apple Glasses). This tracking limitation is also present on current mobile AR solutions.

8 Conclusion and Future Work

This paper presents a networked synchronous cross-device MR solution that allows users of MR-HMDs to use the controllers provided by other MR-HMDs located in the same physical space. This opens up opportunities for researchers to use 6DoF inputs with devices that typically do not support their use. A separate benefit of using this solution is that the devices connected are instanced into a shared synchronous MR experience, allowing this system to be used for research on collaborative environments.

This solution is a work in progress and can be improved in several ways. All of the networking was done over the cloud through Photon Engine. That choice enabled remote collaboration and helped to enable cross-device support; however, it was at the cost of increased latency. Setting up a private server or if the intended use-case allows setting up a local area network could both decrease this latency. This project also used Vuforia for image tracking. It is possible to incorporate a custom image tracking implementation if more control over it is necessary. Another path forward would be to set up the image tracked synchronization anchor alignment to work on a VR-HMD's built-in cameras. This would remove the need to adjust the image target to the positional tracker's location.

While there is room for improvement, this solution uses an easy to assemble collection of off the shelf hardware's and software's to facilitate synchronous cross-device collaboration and controller use. This project can be easily extended for use in immersive analytics, collaborative AR, and other multi-device MR experiences. This ease of use is also found for extending the project to work with a variety of controllers as demonstrated by the use of the Logitech VR-Pen by an HL2 user.

Acknowledgments

We would like to thank Logitech for their support and the Logitech VR-Pen. This work was supported by the National Science Foundation (NSF) awards NSF CNS 2016714, NSF IIS-1948254, and NSF BCS-1928502 awards.

References

[1] Installation guide — mixed reality toolkit documentation, Jan 2020. Available at `https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/Installation.html#4-add-and-configure-mrtk-with-a-new-scene`.

[2] H. Bai, L. Gao, and M. Billinghurst. 6dof input for hololens using vive controller. In *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications*, SA '17. Association for Computing Machinery, New York, NY, USA, 2017. doi: 10.1145/3132787.3132814

[3] M. A. Brown and W. Stuerzlinger. Exploring the throughput potential of in-air pointing. In M. Kurosu, ed., *Human-Computer Interaction. Interaction Platforms and Techniques*, pp. 13–24. Springer International Publishing, Cham, 2016.

[4] J. Brustein. Microsoft wins $480 million army battlefield contract, Nov 2018. Available at `https://www.bloomberg.com/news/articles/2018-11-28/microsoft-wins-480-million-army-battlefield-contract`.

[5] L. Goode. The hololens 2 puts a full-fledged computer on your face, Feb 2019. Available at `https://www.wired.com/story/microsoft-hololens-2-headset/`.

[6] D. Kalkofen, M. Tatzgern, and D. Schmalstieg. Explosion diagrams in augmented reality. In *2009 IEEE Virtual Reality Conference*, pp. 71–78. IEEE, 2009.

[7] F. Kosmalla, A. Zenner, M. Speicher, F. Daiber, N. Herbig, and A. Krüger. *Exploring Rock Climbing in Mixed Reality Environments*, p. 1787–1793. Association for Computing Machinery, New York, NY, USA, 2017.

[8] D.-M. Pham and W. Stuerzlinger. Is the pen mightier than the controller? a comparison of input devices for selection in virtual and augmented reality. In *25th ACM Symposium on Virtual Reality Software and Technology*, VRST '19. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3359996.3364264

[9] J. Sun, W. Stuerzlinger, and B. E. Riecke. Comparing input methods and cursors for 3d positioning with head-mounted displays. In *Proceedings of the 15th ACM Symposium on Applied Perception*, SAP '18. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3225153.3225167

[10] R. J. Teather, A. Pavlovych, W. Stuerzlinger, and I. S. MacKenzie. Effects of tracking technology, latency, and spatial jitter on object movement. In *2009 IEEE Symposium on 3D User Interfaces*, pp. 43–50, 2009. doi: 10.1109/3DUI.2009.4811204

[11] P. Wacker, O. Nowak, S. Voelker, and J. Borchers. Arpen: Mid-air object manipulation techniques for a bimanual ar system with pen & smartphone. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, p. 1–12. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3290605.3300849

[12] A. S. Williams, C. Angelini, M. Kress, E. Ramos Vieira, N. D'Souza, N. D. Rishe, J. Medina, E. Özer, and F. Ortega. Augmented reality for city planning. In J. Y. C. Chen and G. Fragomeni, eds., *Virtual, Augmented and Mixed Reality. Design and Interaction*, pp. 256–271. Springer International Publishing, Cham, 2020.

[13] A. S. Williams and F. Ortega. Insights on visual aid and study design for gesture interaction in limited sensor range augmented reality devices. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 19–22, 2020. doi: 10.1109/VRW50115.2020.00286

[14] A. S. Williams and F. R. Ortega. Understanding gesture and speech multimodal interactions for manipulation tasks in augmented reality using unconstrained elicitation. *Proc. ACM Hum.-Comput. Interact.*, 4(ISS), Nov. 2020. doi: 10.1145/3427330