HoloCapture: A Framework for Data Collection in Collaborative Mixed Reality Environments

JOHN DALE and DANI KASTI, University of Massachusetts, Amherst, USA

ABSTRACT

MR technology holds the potential to revolutionize collaborative experiences, yet there remains a notable scarcity of applications capable of collecting detailed user and virtual object data. Recognizing the value of such data as a quantitative metric for analyzing and enhancing user performance and the overall collaborative MR experience, we have developed a framework designed specifically for data collection in MR environments. Our application engages multiple HoloLens users in a cooperative task of solving a word puzzle by manipulating fifteen poster hologram objects within a virtual space. Throughout the activity, we collect data on hand poses, eye gaze, head positions, and object interactions. This data provides insights into user performance, behavior, engagement, and interaction patterns. It allows researchers to examine how participants collaborate, communicate, and problem-solve in a shared virtual environment. Moreover, this approach enables the measurement of performance outcomes, task completion times, and error rates. Such metrics are invaluable for developers seeking to tailor MR experiences to the cognitive and physical capacities of users. This fine-tuning aims to streamline interactions, reduce cognitive load, and enhance the efficiency and enjoyment of collaborative tasks in MR settings.

ACM Reference Format:
John Dale and Dani Kasti. 2023. HoloCapture: A Framework for Data Collection in Collaborative Mixed Reality Environments. 1, 1 (December 2023), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Mixed reality (MR) is an evolving field that merges the physical and virtual worlds. Its core features revolve around enabling environments where digital and physical elements interact in real-time. MR sits on a spectrum between augmented reality (AR) and virtual reality (VR). AR allows for the overlay of digital information and interactive content onto scenes and objects. MR interaction is more complex than AR, enabling a deeper integration of the two worlds and leading to a more immersive experience. MR anchors virtual objects to the real world as if they were physically present. These objects can interact with and respond to physical world structures and dynamics. For example, a virtual basketball under MR could bounce off the ground. On the other end, VR deals with entirely virtual scenes: the physical world is disconnected from one's reality. All virtual objects will be present in a virtual environment, unable to interact with the physical environment.

MR experiences are generally delivered through an advanced, spatially-aware headset such as the Microsoft HoloLens 2, which will be used in this paper. This device is equipped with sensors, advanced optics, and processing power to understand the environment and enable complex interactions.

1.1 Applications and Motivation

Mixed reality offers many ways to improve shared experiences. For example, a Skype scenario from the HoloLens launch. In Figure 1, a user works through how to fix a broken light switch with help from a remotely located expert [12]. This experience demonstrates a large improvement from the current experience, as video is employed in a much more immersive and effective manner.

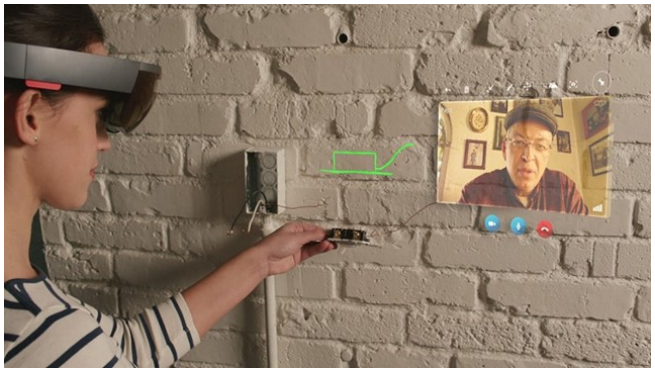

Fig. 1. An expert provides 1:1 guidance from his 2D, desktop computer to a user of a 3D, mixed-reality device. The guidance is synchronous and the physical environments are dissimilar. & (Adapted from [12])

Another example of the opportunities in MR is the OnSight collaboration tool, which was created by NASA's Jet Propulsion Laboratory. Scientists working on data from the Mars rover missions can collaborate with colleagues in real-time with the data from the Mars environment [12]. This unlocks a fresh way for people to work together in virtual settings. It leverages the immersive nature of mixed reality to enhance the way we interact by allowing team members to point out specific aspects directly in the virtual world or to virtually stand alongside a coworker to see what they're seeing and hear what they're saying about their discoveries.

At the core of any productive collaboration is intuitive communication and teamwork. Grasping how to naturally extend these collaborative instincts into the more complex and layered world of mixed reality is key. The goal is to not just replicate the way we share experiences in the real world, but amplify them, improving

Authors' address: John Dale, jkdale@umass.edu; Dani Kasti, danikasti@umass.edu, University of Massachusetts, Amherst, Amherst, Massachusetts, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2023 Association for Computing Machinery.
XXXX-XXXX/2023/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

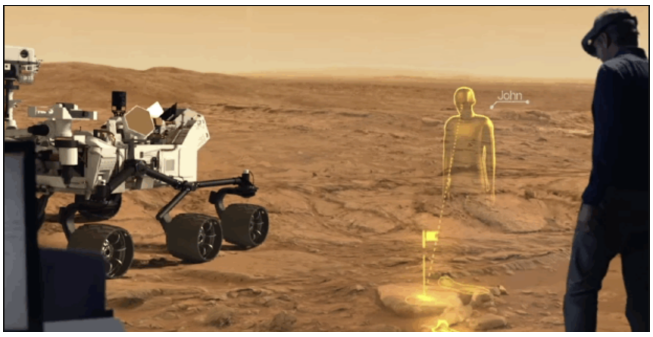

Fig. 2. A scientist explores an environment using a 3D, mixed-reality device with a small group of remote colleagues using 3D and 2D devices. The collaboration is synchronous (but can be revisited asynchronously) and the physical environments are (virtually) similar & (Adapted from [12])

our approach to collaborative work. Using data collection is one such way to improve MR collaborative experiences.

With the integration of MR applications in various fields such as education, healthcare, and engineering, the ability to collect and analyze data is vital for improving these experiences. For example, in educational settings, data on how students interact with virtual objects and each other can shape the development of future educational tools that are engaging and effective. Students can enhance their learning by interacting with virtual content through 3D projections and simulations. MR helps reinforce educational concepts without the risk that real-life learning can bring. For example, a student learning the skills for working in a museum can use MR to practice cleaning an ancient artifact without potentially damaging a valuable item. Using hand pose data and touch positions (as a measure of cleaning coverage) would be vital to assessing how well this trainee is doing.

In [20], researchers created a mixed-reality system that integrates a virtual assembly environment with augmented reality. This approach is mainly based on the development of a hybrid tracking system for the synchronization of the virtual and the real hand of the user. Similarly, technicians wearing an MR headset can view a holographic image of a piece of equipment, such as an engine, and take it apart virtually to examine its inner workings. Using MR is efficient time-wise and avoids needing the tools needed to inspect the physical equipment [2].

Recording the hand poses used in these scenarios can give insight into common gestures and those posing physical strain to the user. For example, the duration of certain poses, the range of motion - which can indicate when one is extending hands or fingers beyond comfortable limits, and frequent repetition of certain hand poses - especially those that are awkward and deviating significantly from a neutral position - are all things that can indicate risk of strains. Tracking which parts of the engine are touched, moved, and zoomed in upon can provide helpful information. The focus time and gaze patterns of the technician can indicate which engine parts are causing the most problems. The position of the technician's head over time can give insight into how he is navigating the engine and if the setup is optimal for the activity. Similarly, the physical layout of the space and the navigation around it can help developers understand if virtual object placement is optimal. Lastly, the time it takes to complete the application and how often he makes mistakes can provide information into the areas where he may need to improve. The ability to do this with collaborative MR, where two technicians are working together on the same engine, is important so that we can obtain quantitative metrics on the activity of both users.

Another example would be retail workers using MR to help them visualize a store layout before carrying it out in the real world. A hologram can display how it will look to add, for example, a display in a certain location of the store. The worker could walk through the store and see a full 3D hologram of the display in the exact spot it would be, possibly even rearranging components interactively.

1.2 Contribution

We propose HoloCapture, a method of data collection on collaborative mixed reality applications. Such applications aim to have people working together to achieve some common goals. We employ an application that involves multiple HoloLens users collaborating to solve a word puzzle by moving fifteen poster hologram objects throughout the virtual environment. For this application, users are colocated; that is, all users will be in the same physical space when solving the puzzle. Our intended purpose is to do this synchronously, that is, sharing the holographic experience at the same time. Photon Unity Networking and Azure Spatial Anchors allow us to enable sharing between object interaction in a synchronous multi-user mixed reality environment. Anchor sharing gives us the ability to align coordinates across multiple devices. Networking allows positions, interactions, and movements of people and holograms to be synchronized in real-time across all participants [12]. The application serves as a base structure for future work in the collaborative mixed reality research area. Furthermore, we create a system that collects data on the user and the environment. This includes hand pose, eye gaze, head position, object positions, and touch positions throughout the application. Our work makes the following contributions:

- Unity design of this collaborative mixed reality application with data collection scheme
- The data collected throughout the collaborative experience

2 RELATED WORK

In the field of mixed reality (MR), research has increasingly focused on leveraging data feedback to enhance collaborative experiences and address multi-user environment challenges. By strategically using this feedback, these studies aim to optimize MR environments for more engaging, efficient, and user-centric collaboration.

In a paper by S.K. Ong and Y. Shen, a prototype MR space was designed where users interact with virtual products through intuitive interfaces, blending 3D physical and virtual spaces effectively. They accomplish this through a client/server architecture where multiple users can create and modify objects in 3D physical space through an intuitive drag-and-drop method based on shape control points [18].

Another study bridged parametric architectural modeling software with virtual reality (VR) headsets, delving into concurrency

issues in multi-user scenarios. They proposed a basic locking strategy to manage editing conflicts, considering the complexities of concurrent immersive editing. This concept mirrors web-based collaborative tools, where a similar approach is applied to 3D design systems, allowing multiple users to independently modify designs without conflict [17].

Research on collaborative control strategies in MR and AR environments has also been explored. Authors Jonathan Wieland, Johannes Zagermann, Jens Müller, and Harald Reiterer examine three manipulation techniques in handheld AR: Separation, Composition, and Hybrid. The Separation technique divides rotation and positioning tasks among users for clarity in collaboration, Composition merges inputs for shared tasks to offer varied perspectives, and Hybrid flexibly combines these approaches for both clear role distribution and collaborative freedom. They found that while accuracy was consistent across techniques, the Separation and Hybrid methods enabled faster task completion compared to Composition. This was attributed to the ability to perform multiple manipulations simultaneously in Separation and Hybrid, as opposed to the sequential approach in Composition. Ultimately, the study suggests a preference for the Separation technique due to its efficiency and reduced workload for participants [19].

Research combining interaction with data feedback has been examined as well. Authors Tan Jiang, Xiao'er Gan, Zheng Liang, and Guang Luo, implemented a novel approach to data collection in a virtual digital museum environment. This method integrates mixed reality and software-driven techniques to gather visitor behavior, exhibit information, and environmental data. Visitor behavior data, like movement and interaction patterns, aids in personalizing visitor guidance and optimizing exhibit layout. Exhibit information data enriches visitor experience with cultural and historical insights. Environmental data, collected to maintain exhibit preservation, includes factors like temperature and humidity. The collected data is analyzed to enhance visitor experience and improve museum operations [1].

Our paper aims to develop a comprehensive framework for collaborative mixed reality (MR) environments, enhancing data collection and interaction. Utilizing the Mixed Reality Toolkit and HoloLens 2, we strive to create a user-friendly platform that simplifies the implementation of advanced MR features, fostering innovation and streamlining the development of collaborative MR applications. Mixed reality applications require constant interaction between users and their virtual environment to provide an engaging experience. By designing a platform that provides collaboration and rich user interaction and movement data, researchers can implement these advanced features in an efficient approach to further advance Mixed Reality applications.

3 SYSTEM DESIGN

This section will overview the hardware and software components utilized for integrating sensor capture and collaborative MR environments into our HoloLens application.

3.1 HoloLens Setup

Developer Mode: Developer mode on the HoloLens 2 allows external management of the device's file structure and additional performance evaluation metrics. In order to deploy applications to the HoloLens from a Windows machine, both devices must have Developer mode enabled and run through a pairing process. Functionality including live performance traces, diagnostic information, and interaction with the filesystem can be managed through the Windows Device Portal, accessed through the device's web interface. The HoloLens 2 Device Portal utilizes HTTPS protocol and can be reached by entering the Device IP address into a browser [15].

Research Mode: HoloLens 2 research mode is a feature designed for academic and research purposes which provides access to raw sensor streams and additional functionalities not available during standard operating mode. The following sensors and cameras provide raw data streams through Research Mode [5].

Depth Camera: Utilized for spatial understanding and 3D mapping; provides data access through Azure Kinetic Sensor technology

Visible Light Camera (PV Camera): 8 megapixel RGB camera; 1080p videos at 30FPS; autofocus and auto exposure

Long Throw Depth Camera: Extended range depth sensing; beneficial for outdoor or spacious indoor environments

Visible Light Environment Understanding Cameras (EUC): Four gray-scale cameras capturing 30 frames per second; enable advanced vision applications like Simultaneous Localization and Mapping (SLAM) and motion tracking.

Infrared Cameras: IR illuminator; two infrared (IR) cameras; capable of advanced sensing including eye tracking, depth sensing, and hand tracking

Accelerometer, Gyroscope, and Magnetometer Data: capturing motion and orientation information

3.2 Collaboration Setup

Photon Unity Networking (PUN) and Azure Spatial Anchors facilitate the sharing of objects and virtual interactions among individuals in a multi-user mixed reality environment. PUN, a networking framework tailored for Unity, synchronizes game states across multiple devices, ensuring a consistent experience for all participants. Azure Spatial Anchors, a cloud-based service from Microsoft Azure, enables developers to create shared anchors that are crucial for positioning objects accurately in a shared virtual space.

In our application, we leverage both spatial anchors and PUN to enable the creation, updating, and sharing of objects within a common environment, thus enhancing collaborative experiences.

Key components of PUN and Azure Spatial Anchors used in our application are detailed below:

The *Photon.Pun* and *Photon.Realtime* namespaces provide the necessary functionality for real-time multiplayer networking in our Unity application [10, 11].

The *SpatialAnchorManager*, a central element of Azure Spatial Anchors, handles communication with the Azure cloud service. This manager facilitates the creation, update, and maintenance of spatial anchors, allowing multiple users to interact with and share the same spatial references.

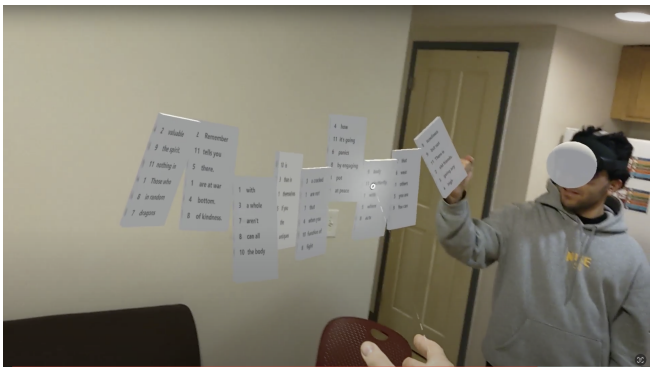

Fig. 3. First person perspective of two HoloLens users interacting to solve a Word Puzzle in our collaborative application

3.3 Data Collection Setup

The Mixed Reality Toolkit 2 (MRTK2) provides abstraction from raw sensor streams, creating easily accessible high level information. MRTK2 acts as a bridge between raw sensor data captured at the hardware layer and the application layer, where developers build mixed reality experiences. We used MRTK2's classes, methods, and enumerations for logging hand tracking, eye tracking, positional, and touch event data.

Key elements of the MRTK2 and Unity physics engine utilized in our implementation are detailed below.

The *Microsoft.MixedReality.Toolkit.Utilities* namespace offers a suite of utility classes and tools essential for various operations in mixed reality development [7].

The *TrackedHandJoint* enumeration is used to identify specific joints in the user's hand. This allows for precise tracking and interaction capabilities by identifying individual fingers and hand parts [14].

The *MixedRealityPose* struct provides a framework for handling the positional and rotational data of hand joints [9].

The *IMixedRealityEyeGazeProvider* interface yields data on user gaze, including direction, origin, and the intercepted GameObjects [6].

The *Collider* component is a Unity physics element used for defining the physical shape of GameObjects for collision detection. In MRTK2, it's utilized for triggering events when virtual objects interact with each other, or with a user [3].

The *StreamWriter* class, part of the .NET framework's System.IO namespace, is used for logging and writing output data to files during application runtime [13].

4 IMPLEMENTATION

This section introduces the hardware and software tools employed to facilitate collaborative mixed reality features and data collection on the HoloLens 2 platform in our application. We utilized a range of software tools including Unity, OpenXR, MRTK2, Visual Studio, and Windows 10. Additionally, this section will discuss the challenges encountered during this process and present the solutions we implemented.

4.1 Application

In our MR application, multiple users wearing HoloLens 2 headsets collaborate to solve a word puzzle. The task involves moving fifteen poster holograms within the virtual environment and verbally announcing sentences that are correctly formed when the puzzle is solved. Throughout this experience, each user collects data about themselves and their surroundings, including eye, head, and hand tracking, as well as the positions of posters and touchpoints on the posters. Data is recorded at a frequency of every 100 milliseconds, yielding ten measurements per second. The data collected during this collaborative experience is the primary focus and contribution of this paper, providing a foundational structure for future research in the field of collaborative mixed reality.

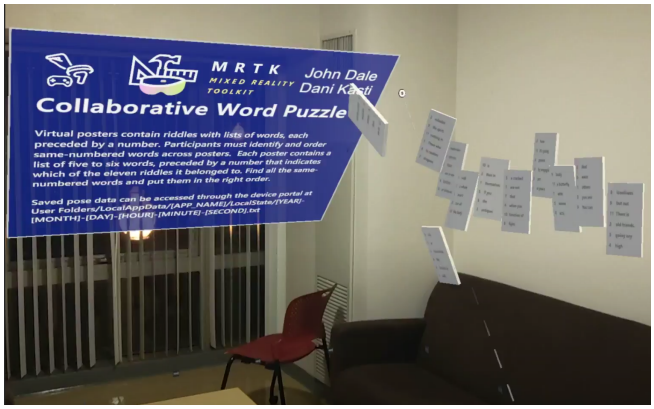

Fig. 4. User view of the WordPuzzle collaborative HoloLens 2 application

Prerequisites For application development, a development machine must be capable of running Unity and support the Universal Windows Build Platform for HoloLens applications. Consequently, this machine should be equipped with either Windows 10 or Windows 11, as there are no separate SDKs available for Windows Mixed Reality development [4]. Additionally, the machine must have Visual Studio installed, complete with the .NET framework and C++ desktop development tools, to facilitate compilation and deployment. We recommend using the Mixed Reality Feature Tool, provided by Microsoft, for integrating Mixed Reality feature packages. The following packages were imported into Unity to provide key features, including Azure Spatial Anchors, PUN2 networking, OpenXR, and MRTK2 assets and tools.

Imported through Mixed Reality Feature Tool:

- Azure Spatial Anchors SDK Core [2.12.0]
- Azure Spatial Anchors SDK for Windows [2.12.0]
- Mixed Reality OpenXR Plugin [1.8.1]
- Mixed Reality Toolkit Examples [2.7.0]
- Mixed Reality Toolkit Extensions [2.7.0]
- Mixed Reality Toolkit Foundation [2.7.0]
- Mixed Reality Toolkit Standard Assets [2.7.0]
- Mixed Reality Toolkit Tools [2.7.0]

External Packages from MRTK HoloLens2 Unity Tutorials Assets:

- MultiUserCapabilities [2.7.2]
- GettingStarted [3.0.0]
- AzureSpatialAnchors.XRpluginManagement [2.5.3]
- AzureSpatialAnchors [3.0.0]

Other Packages:

- Photon Unity Networking (PUN2 - Free) [2.43]

Software Versions:

- Unity Editor [2020.3.48f1]
- Visual Studio 2019 [16.11.31]
- AzureSpatialAnchors XRpluginManagement [2.5.3]
- Microsoft Windows 10 [OS build 19045.3693]

4.2 Problems Encountered

In our initial application development, we attempted to implement Mixed Reality features independently, following the tutorial series provided by Microsoft. However, with the introduction of MRTK3 on September 6, 2023, many of these Microsoft tutorials have shifted to using the updated toolkit for demonstrating Mixed Reality features [8]. Consequently, we encountered dependency issues with newer versions of packages designed for MRTK3. This necessitated a complete backward compatibility reset of our development environment. To simplify our environment and debugging processes, we implemented features such as spatial anchors, hand tracking, eye tracking, user interaction, networking, and data collection individually.

Integrating shared interactable GameObjects required precise attachment of scripts inside of the Unity hierarchy. These scripts add functionality and properties to GameObjects, allowing for advanced features like networking, data collection, and intractability. During our implementation, we ran into many issues originating from scripts being attached improperly to GameObjects or parent objects. Documentation outlining the proper usage of these scripts, commonly included in the assets folder for the package, was critical to solving these issues.

While building the application, the Unity project editor's play mode was critical for making and testing local changes. This mode provided instant feedback when implementing features like Unity networking, object instantiation, object interaction, and the visual appearance of the application. However, Unity play mode has its limitations, especially in testing scenarios involving collaborative interaction, data collection, runtime errors, and performance. Consequently, these features proved the most challenging to implement and debug, as they necessitated deployment to the HoloLens 2. For addressing these implementation issues, app crash dumps, performance traces, and console error logs accessed through the Windows device portal, along with the output from Visual Studio debugging, were the most beneficial debugging methods.

It's important to note that the integration of these features into a unified application necessitates changes in the project's hierarchy. Scripts responsible for networking objects, managing and sharing spatial anchors, and collecting data, must be reorganized around parent objects. This restructuring is essential due to the instantiation of numerous objects within the scene.

While building the application, incorrect setup of the MRTK and OpenXR profiles, as well as project settings, were the main cause of non-functional features. Developers should be aware of the following changes in these settings:

To access these settings in the Unity editor, navigate to File → Build Settings → Player Settings.

MRTK Profiles: Our application utilized an imported MixedRealityToolkit profile, with additional configuration as required. The profile, sourced from the Mixed Reality Toolkit Foundation, is named *DefaultMixedRealityToolkitConfigurationProfile*. The following modifications were made to adapt this configuration profile to our specific use case:

- Input → Pointers → Is Eye Tracking Enabled (This will enable eye gaze input).
- Spatial Awareness → OpenXR Spatial Mesh Observer → Display Settings → Display Option → Set to Occlusion (This prevents the spatial mesh from being constantly displayed).

OpenXR Settings and Profiles: The following Feature Groups should be enabled in XR Plug-in Management → OpenXR:

- Hand Tracking: enables the tracking of hand movements and gestures, allowing for natural interaction within virtual environments.
- Hand Interaction Poses: provides predefined hand poses for common interactions, simplifying the development of hand-based controls.
- Motion Controller Model: integrates physical controller models into the virtual environment, enhancing the realism and interactivity of user inputs.

Ensure that the *Runtime Debugger* is not selected in the features page. This debugger encountered memory access exceptions during runtime and resulted in the crashing of our application.

Interaction Profiles: The following interaction profiles should be added:

- Eye Gaze Interaction Profile: facilitates eye tracking for interaction, allowing applications to respond to where the user is looking.
- Hand Interaction Profile: defines standard interactions for hand tracking, ensuring consistent and intuitive user experiences.
- Microsoft Hand Interaction Profile: optimizes hand tracking and interactions in the context of Microsoft's ecosystem and hardware.

Project Settings: The following Capabilities should be enabled in Player → Publishing Settings:

- InternetClient: allows the app to access the internet.
- InternetClientServer: enables network communication capabilities.
- PrivateNetworkClientServer: permits communication on private networks.
- WebCam: needed for accessing the device's webcam, crucial for AR experiences.
- Microphone: enables voice input and audio recording functionalities.
- Spatial Perception: allows the app to understand and interact with the physical space around the user.

- GazeInput: enables eye tracking, allowing users to interact with the app using their gaze.

These settings and features will enable and include the necessary systems in the application build. Developers must ensure the correct build and deployment steps of their application to ensure its compatibility with the HoloLens 2 platform.

Build Steps:

- File → Build Settings
- Target Device: HoloLens
- Architecture: ARM64
- Build Type: D3D Project
- Target SDK Version: 10.0.19041.0
- Minimum Platform Version : 10.0.10240.0
- Visual Studio Version: Latest installed
- Build and Run on: Local machine
- Build Configuration: Release
- Check Development Build box
- Compression Method: Default

Deployment Steps: Deployment to the HoloLens through Visual Studio will load the built Unity application onto the HoloLens device [16].

- Open your project .SLN file in Visual Studio 2019
- Select Release and ARM64
- Project properties → Debugging → Configuration [Active(Release)] and Platform [ARM64]
- Debugging → Machine Name → Enter IP address of HoloLens
- Debugging → Authentication Type → Universal (Unencrypted Protocol)

Testing the Application: After deploying our application to the HoloLens 2, we faced numerous challenges, including performance issues, object skew, and problems with object instantiation. These issues were primarily rooted in incorrect project hierarchy and project settings. By detailing the settings and steps above, we aim to help prevent these issues in future applications.

5 EVALUATION

This section will detail the qualitative performance of the implemented application and results of testing.

5.1 Overview

Our implementation was primarily focused on enhancing usability and stimulating collaborative actions in a virtual environment. With this objective, our evaluation metrics are qualitative, assessing how well the application facilitates collaboration and interaction.

5.2 Collaboration

The application successfully enabled real-time sharing and tracking of objects. Users interacted with objects through a 'tradeoff' mechanism: while preventing simultaneous movement of an object by both users, it allowed each user to take control of an object upon initiating an interaction. Although this setup restricted simultaneous multi-user input—a limitation in certain scenarios—it still effectively supported collaborative puzzle solving. Users found it more efficient to work on different puzzle pieces separately. Despite the limitations, this approach did not largely hinder user interaction and resulted in substantial data collection during tests.

5.3 Data collection

Leveraging the wide array of sensors and cameras on the HoloLens 2, our application gathered extensive data to analyze user behavior. All metrics were referenced to a common coordinate system centered within our environment, ensuring uniformity regardless of the users' initial positions in the application, as seen in Fig. 5. This design allowed for straightforward data comparison without the need to match coordinate systems between users. Data captured included relative head positions, hand joint positions, object positions, and eye gaze directions with hit objects, and touch locations. Each metric was recorded at a frequency of 10 times per second (100ms intervals). This rate of data collection did not adversely affect application performance, and further interpolation is possible to synchronize data streams between the slightly offset data collection timings of different HoloLens units. We did not explore increasing the data collection frequency as it was beyond the scope of this application.

```
    Start Time: 19:31:58.465
Time  Update Time: 2.983
1.  Head Position: (0.0, -0.1, 0.0)
2.  Coffee Cup Position: (-0.1, -0.3, 0.9)
3.  Eye Gaze Origin: (0.7, 0.6, 0.4)
          Direction: (-0.6, -0.7, 0.4)
          Hit Object: CoffeeCup(Clone)
4.  Left Wrist Position: (0.4, 0.4, -0.5)
    Left Wrist Rotation: (308.7, 349.3, 174.6)
    Left Palm Position: (0.4, 0.5, -0.4)
    Left Palm Rotation: (308.7, 349.3, 174.6)
    Left ThumbMetacarpalJoint Position: (0.4, 0.4, -0.5)
    Left ThumbMetacarpalJoint Rotation: (308.4, 276.0, 156.3)
                    •
                    •
                    •
5.  Right PinkyMiddleJoint Rotation: (64.4, 250.2, 309.0)
    Right PinkyDistalJoint Position: (0.5, 0.3, -0.4)
    Right PinkyDistalJoint Rotation: (38.7, 163.1, 208.6)
    Right PinkyTip Position: (0.5, 0.3, -0.5)
    Right PinkyTip Rotation: (38.7, 163.1, 208.6)
6.  Touch Point at 20:41:55.607: (-0.1, -0.2, 0.9)
```

Fig. 5. Example text file displaying all types of data we are collecting from each HoloLens 2 user

This application serves as a functional prototype for collaborative applications. Our evaluation aimed to achieve intuitive interaction and collaboration, along with high-speed data capture. Our tests met these specifications, although quantitative performance metrics were not extensively measured.

6 GENERALIZED SETUP

This section outlines generalized protocols for building and deploying an application to the HoloLens.

First, ensure that both Research Mode and Development Mode are activated on the device. Development Mode can be enabled through

the device's settings. This action allows the developer to connect to the Windows Device Portal and enable Research Mode.

Next, build your Unity application as detailed in the implementation section.

Finally, ensure that both the HoloLens and the development machine are connected to the same WiFi network. You can then perform a remote deployment to the HoloLens. The application will automatically start running once the deployment is complete.

For project hierarchy, custom scripts related to data collection and networking, and more Unity project details, refer to our GitHub repository https://github.com/JohnDale02/HoloLens-Collaborative-Mixed-Reality.git

7 DISCUSSION

In this paper, we have provided a framework for enabling data collection in MR applications. We provide a method of collecting data on hand poses, eye gaze, head position, and object manipulation. Furthermore, the measurement of performance outcomes, task completion times, and error rates are all important metrics that are also useful to users, researchers, and MR developers.

Researchers can use this data to analyze how participants collaborate, communicate, and solve problems together in a shared virtual space. Furthermore, it provides insights into user behavior, engagement, and interaction patterns. This data reveals the gestures and non-verbal cues that users naturally employ in a collaborative setting and provides feedback to each user on how active they were in completing the collaborative task. Data collection is crucial for helping users improve their skills. The user can use the data to understand his strengths and weaknesses, allowing him to improve upon his abilities.

Data collection is important for analyzing the effectiveness of MR as a collaborative tool. Developers can utilize these metrics to adjust MR experiences to align closely with the cognitive and physical abilities of users. This helps in making interaction more natural and reduces the cognitive load when completing collaborative tasks. By reducing unnecessary complexity, and reducing the learning curve of the MR experience, users can focus on the task at hand without being overwhelmed by the technology itself; this leads to better performance and less frustration, enhancing usability and efficiency. Especially in environments where safety is a concern, such as industrial environments or medical simulations, ensuring that the MR interface doesn't overload users is crucial; furthermore, users can learn and retain information more effectively, which is particularly important in educational and training situations. As with any technology, user-friendly designs that consider cognitive and physical abilities are more likely to be adopted and used regularly as they fit more seamlessly into users' lives and workflows. MR and shared experiences are still in their infancy; data collection is one avenue to help improve its usability and applicability.

Data collection in MR is not just about optimization but also about creating new forms of collaborative experiences. As we collect more data and use this data to gain more insights, we can continue to design MR systems that support new types of collaborations, thus paving the way for improvements in how we work and learn together.

8 TRADEOFFS, LIMITATIONS AND FUTURE WORK

8.1 Tradeoffs

With data collection enabled, it is important to note the privacy issues associated with it. Notably, identifiability; such data could be used to identify individuals, especially when combined with other data sources. This is especially true in sensitive environments, such as educational institutions. This means that some organizations may not enable the data collection scheme in their MR application.

8.2 Limitations and Future Work

While our application enables users to enlarge or reduce the size of objects, we observed during testing that these changes were not visible to other users within the same networked room. Resolving this issue is crucial to ensure that all users are aware of each other's interactions. Additionally, collecting data on the degree of enlargement or reduction of objects could offer valuable insights into which elements are being inspected, the depth of this inspection, and the amount of time spent on it. This information is particularly relevant in scenarios like the technician example previously mentioned.

We also encountered orientation issues when two different HoloLens users started from distinct positions. Specifically, when one user looked at another, there was a misalignment between the physical position of the user and their virtual counterpart. This discrepancy caused objects to move in unexpected ways relative to the user's movements. To avoid this limitation, it requires both HoloLens users to start their applications in the exact same orientation for effective co-located interaction. Nonetheless, further research is needed to understand and address this limitation effectively.

REFERENCES

[1] 2021. AIDM: artificial intelligent for digital museum autonomous system with mixed reality and software-driven data collection and analysis. https://doi.org/10.1007/s10515-021-00315-9

[2] 2023. Augmented reality vs. virtual reality vs. mixed reality | TechTarget. https://www.techtarget.com/searcherp/feature/AR-vs-VR-vs-MR-Differences-similarities-and-manufacturing-uses

[3] 2023. Collider PhysicsCollider | Package Manager UI website. https://docs.unity3d.com/Packages/com.unity.physics@0.0/manual/core_components.html

[4] 2023. HoloLens 2 fundamentals: develop mixed reality applications - Training. https://learn.microsoft.com/en-us/training/paths/beginner-hololens-2-tutorials/

[5] 2023. HoloLens Research Mode - Mixed Reality. https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/research-mode

[6] 2023. IMixedRealityEyeGazeProvider Interface (Microsoft.MixedReality.Toolkit.Input). https://learn.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.input.imixedrealityeyegazeprovider?view=mixed-reality-toolkit-unity-2020-dotnet-2.8.0

[7] 2023. Microsoft.MixedReality.Toolkit.Utilities Namespace. https://learn.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.utilities?view=mixed-reality-toolkit-unity-2020-dotnet-2.8.0

[8] 2023. Mixed Reality Toolkit 3 Developer Documentation - MRTK3. https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-overview/

[9] 2023. MixedRealityPose Struct (Microsoft.MixedReality.Toolkit.Utilities). https://learn.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.utilities.mixedrealitypose?view=mixed-reality-toolkit-unity-2020-dotnet-2.8.0

[10] 2023. Photon Unity Networking 2: Photon.Pun Namespace Reference. https://doc-api.photonengine.com/en/pun/current/namespace_photon_1_1_pun.html

[11] 2023. Photon Unity Networking 2: Photon.Realtime Namespace Reference. https://doc-api.photonengine.com/en/pun/current/namespace_photon_1_1_realtime.html

[12] 2023. Shared experiences in mixed reality - Mixed Reality. https://learn.microsoft.com/en-us/windows/mixed-reality/design/shared-experiences-in-mixed-reality

[13] 2023. StreamWriter Class (System.IO). https://learn.microsoft.com/en-us/dotnet/api/system.io.streamwriter?view=net-8.0
[14] 2023. TrackedHandJoint Enum (Microsoft.MixedReality.Toolkit.Utilities). https://learn.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.utilities.trackedhandjoint?view=mixed-reality-toolkit-unity-2020-dotnet-2.8.0
[15] 2023. Using the Windows Device Portal - Mixed Reality. https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/using-the-windows-device-portal
[16] 2023. Using Visual Studio to deploy and debug - Mixed Reality. https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/using-visual-studio
[17] Adrien Coppens and Tom Mens. 2018. Towards Collaborative Immersive Environments for Parametric Modelling. In *Cooperative Design, Visualization, and Engineering - 15th International Conference, CDVE 2018, Hangzhou, China, October 21-24, 2018, Proceedings (Lecture Notes in Computer Science, Vol. 11151)*, Yuhua Luo (Ed.). Springer, 304–307. https://www.wikidata.org/entity/Q58209378
[18] S.K. Ong and Y. Shen. 2009. A mixed reality environment for collaborative product design and development. *CIRP Annals* 58, 1 (2009), 139–142. https://doi.org/10.1016/j.cirp.2009.03.020
[19] Jonathan Wieland, Johannes Zagermann, Jens Müller, and Harald Reiterer. 2021. Separation, Composition, or Hybrid? – Comparing Collaborative 3D Object Manipulation Techniques for Handheld Augmented Reality. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 403–412. https://doi.org/10.1109/ISMAR52148.2021.00057
[20] Ulises Zaldívar-Colado, Samir Garbaya, Paúl Tamayo-Serrano, Xiomara Zaldívar-Colado, and Pierre Blazevic. 2017. A mixed reality for virtual assembly. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. 739–744. https://doi.org/10.1109/ROMAN.2017.8172385