GLOBAL RESEARCH IMMERSION PROGRAM FOR YOUNG SCIENTISTS

AR-Enhanced Robot Control: Developing an Immersive Interface Using MRTK for Intuitive Teleoperation

Author: Wenbo Zhu (ZHUW0019@e.ntu.edu.sg)
College of Computing and Data Science, Nanyang Technological University (Singapore)

1 Background

Traditional robot control interfaces often lack intuitive spatial mapping, leading to reduced efficiency and increased cognitive load for operators. Existing teleoperation systems frequently suffer from limited immersion, hampering precise real-time control. This research addresses these challenges by using Augmented Reality (AR) headsets to create an intuitive, immersive interface for robot control in real-world environments. This approach aims to enhance spatial understanding, improve operation accuracy, and reduce cognitive burden, potentially revolutionizing fields such as search and rescue, remote maintenance, and hazardous environment operations.

2 Objectives

1. Conduct research on Microsoft's Mixed Reality Toolkit (MRTK), the HoloLens AR headset, and Unity.
2. Design and implement an immersive robot control user interface using MRTK and Unity.
3. Investigate communication protocols between the robot and the client application.
4. Leverage the API provided by Lailu to facilitate real-world robot movement and control.

3 Methods

3.1 MRTK & Unity Overview

The Mixed Reality Toolkit (MRTK), developed by Microsoft, is a powerful framework that enables the creation of diverse user interface scenes for the HoloLens AR headset using Unity. By utilizing the prefabs provided by MRTK, developers can easily instantiate elements within a newly created Unity scene, allowing for the rapid design and customization of immersive environments. This flexibility empowers developers to define tailored AR experiences that enhance user interaction and engagement.

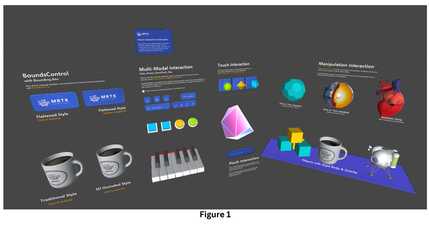

Figure 1

3.2 User Interface Design

The user interface for the robot control system was designed and implemented using the Mixed Reality Toolkit (MRTK) in Unity. The UI consists of a main control panel featuring a prominent joystick for robot navigation. Below the joystick, a blue panel displays the title "Sterilization Robot Control Joystick" and includes several interactive buttons for different control modes: Profiler, Hand Ray, Hand Mesh, Hand Joint, and Rec. These buttons allow the operator to switch between various hand tracking and interaction methods, enhancing the versatility of the robot control interface.

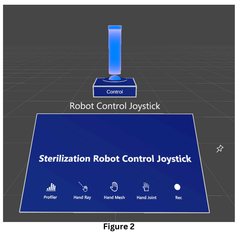

Figure 2

3.3 Communication

The communication system utilizes Netty as the robot's server and Google Protobuf for data serialization, enhancing security by making the data difficult to decompile. In Google Protobuf, a `.proto` file defines the schema for structured data in a language-agnostic manner. By using the proto compiler, we can generate code in various programming languages for both the server and client. To validate this approach, I developed a Netty server in Java on my laptop and created a simple C# script on Unity for testing. As shown in the figure 4, the data was successfully received and recognized.

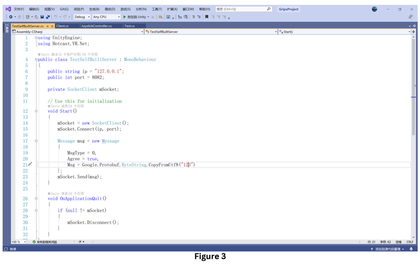

Figure 3

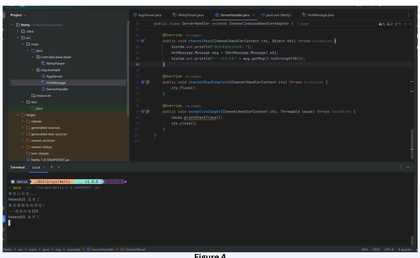

Figure 4

3.4 API Documentation

Our robot communicates using a unified message body called `TransferData`, defined in the `.proto` file provided by Lailu. This message body includes a `type` field that specifies the action for the robot, a `robotID`, and a `data` field that contains information such as location, `UUID`, and `robot state`. The various message types used in the `data` field are also outlined in the provided `.proto` file.

Message TransferData	
Value Type	**Name**
int32	type
int32	robotID
bytes	data
string	uuid
bool	state

4 Results

An Augmented Reality (AR) interface was successfully developed using the Mixed Reality Toolkit (MRTK) for robot control (Figure 2). Users can access the control system by wearing an AR headset and launching the application. The interface features a virtual joystick that allows for intuitive robot movement control in the real world. Initial tests demonstrate that the robot responds to user inputs through the AR interface, executing movement commands as directed (Figure 5). This AR-based control method has proven to be a user-friendly solution for remote robot operation, potentially reducing the learning curve compared to traditional control interfaces. The implementation lays the groundwork for further refinement and expanded functionality in AR-based robot control.

Figure 5

5 Discussion

In the current implementation, a significant challenge of latency was encountered, which impacts the responsiveness and overall user experience of the AR-based robot control system. This latency issue is primarily attributed to the extensive use of coroutines in the current code structure, which, while providing asynchronous operation capabilities, may be introducing unexpected delays in the AR-to-robot command pipeline. To address this issue, future improvements could include optimizing the code structure by reevaluating the use of coroutines and potentially replacing them with more efficient asynchronous programming patterns where appropriate.

This may involve implementing a more streamlined event-driven architecture, utilizing Unity's Job System for parallel processing, or exploring alternative methods for handling asynchronous operations that introduce less overhead. Additionally, profiling the existing coroutines to identify bottlenecks and optimizing critical paths could significantly improve system responsiveness and enhance the overall user experience in AR-based robot control.

6 Conclusion

In conclusion, this research successfully developed an immersive Augmented Reality (AR) interface for robot control using Microsoft's Mixed Reality Toolkit (MRTK) and Unity. The implementation showcases the potential for AR to enhance spatial understanding and streamline the operation of robots in real-world environments, ultimately reducing the cognitive load for operators. While initial tests reveal a user-friendly interaction model, the identified latency issues present a significant challenge that must be addressed to improve responsiveness and overall user experience. Future work will focus on optimizing the underlying code structure, particularly concerning the use of coroutines, to enhance the efficiency of the AR-to-robot command pipeline. This advancement aims to further solidify the role of AR technology in revolutionizing remote robot operations across various applications, including search and rescue, remote maintenance, and hazardous environment operations.

7 References

1. Sean-Kerawala. (2022, November 2). Set up a new OpenXR project with MRTK - Mixed Reality. Microsoft.com. https://learn.microsoft.com/en-ca/windows/mixed-reality/develop/unity/new-openxr-project-with-mrtk
2. DavidSheh. (2017). GitHub - DavidSheh/UnityProtobufDemo: use protobuf in Unity3D. Client (Unity3D + Protobuf) + Server (Java + Netty + Protobuf). GitHub. https://github.com/DavidSheh/UnityProtobufDemo
3. MixedRealityToolkit. (2024, June 3). GitHub - MixedRealityToolkit/MixedRealityToolkit-Unity: This repository holds the third generation of the Mixed Reality Toolkit for Unity. The latest version of the MRTK can be found here. GitHub. https://github.com/MixedRealityToolkit/MixedRealityToolkit-Unity
4. microsoft. (2022). GitHub - microsoft/MRTK3-iet-tutorials: MRTK3 IET tutorials. GitHub. https://github.com/microsoft/MRTK3-iet-tutorials
5. microsoft. (2022b). GitHub - microsoft/ZappysPlayground. GitHub. https://github.com/microsoft/ZappysPlayground
6. (2024). Zhihu.com. https://zhuanlan.zhihu.com/p/708342906
7. (2024b). Zhihu.com. https://zhuanlan.zhihu.com/p/367186910
8. (2024c). Zhihu.com. https://zhuanlan.zhihu.com/p/697358008