Vision Language Model-Based Solution for Obstruction Attack in AR: A Meta Quest 3 Implementation

Yanming Xiu[1] Maria Gorlatova[2]

Department of Electrical and Computer Engineering, Duke University

Abstract

Obstruction attacks in Augmented Reality (AR) pose significant challenges by obscuring critical real-world objects. This work demonstrates the first implementation of obstruction detection on a video see-through head-mounted display (HMD), the Meta Quest 3. Leveraging a vision language models (VLM) and a multi-modal object detection model, our system detects obstructions by analyzing both raw and augmented images. Due to limited access to raw camera feeds, the system employs an image-capturing approach using Oculus casting, capturing a sequence of images and finding the raw image from them. Our implementation showcases the feasibility of effective obstruction detection in AR environments and highlights future opportunities for improving real-time detection through enhanced camera access.

Index Terms: Mixed / Augmented Reality—Vision Language Models—Object Detection—Task-Detrimental Content—Scene Understanding—Head-Mounted Display;

1 Introduction

Augmented Reality (AR) seamlessly integrates virtual content with the real world, offering transformative experiences across domains such as education and entertainment. However, improper placement of virtual content can obstruct critical real-world objects, leading to confusion or even hazardous situations [1]. For example, when a user is working in a factory, a virtual object overlaid on a "flammable" sign can obscure vital information, jeopardizing user safety.

To address this issue, our recent research introduced ViDDAR [6], a system using advanced vision language models (VLMs) and multi-modal object detection techniques to detect obstruction attacks. While the system has shown promising results on mobile devices, extending its capabilities to head-mounted displays (HMDs) presents unique challenges, such as restricted access to raw camera views and more complex communication procedures. In this demonstration, we showcase a novel implementation of obstruction detection on an HMD, specifically the Meta Quest 3, as shown in Fig. 1. By leveraging a combination of Oculus casting and user-controlled detection mechanisms, the system overcomes hardware and access limitations to accurately identify obstructions. The demonstration highlights the system's capability to efficiently analyze raw and augmented views with low latency on HMDs. The demo video can be accessed via this link.

2 System Architecture

Fig. 2(a) shows the diagram of our proposed system. It consists of three components: an HMD, a local device, and a cloud server.

2.1 HMD

The system uses the Meta Quest 3 headset as the HMD, accompanied by two controllers. The visual experience for users is managed by a Unity application running on the local device. The headset connects to the local device via a Meta Quest Link cable, allowing the Unity game scene to be rendered and displayed on the headset. Interaction with the system is facilitated through two controllers. The headset processes inputs from the controllers and transmits relevant data to the processing unit. This setup allows users to interact with the system efficiently and perform various tasks.

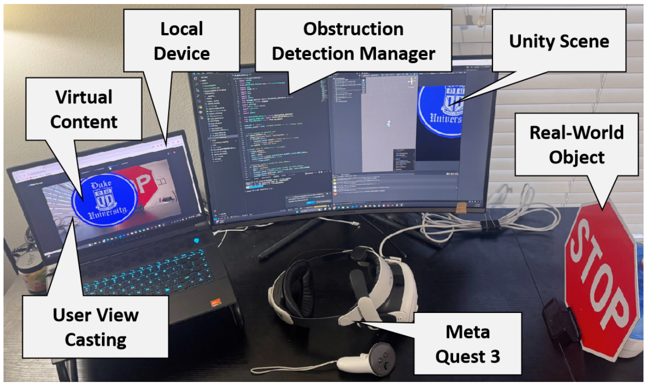

Figure 1: The setup of our HMD-based obstruction detection system. The cloud server is accessed remotely via OpenAI API.

2.2 Local Device

The local device performs three roles: hosting Oculus casting, managing the Unity game scene, and controlling the detection process via the obstruction detection manager. The system uses an Alienware M16 R1 equipped with an RTX 4060 GPU as the local device, enabling efficient multitasking, including rendering the Unity environment and hosting multiple deep learning models.

Oculus Casting: The local device launches a web browser to access the Oculus casting service [4] provided by Meta. This service enables the Meta Quest 3 to cast its display—representing the AR view visible to users—to the browser in real-time via a Wi-Fi connection.

Unity Scene: We use Unity 2022.3.28f1 to design an interactive AR game using the Meta XR Interaction SDK Essentials. The user takes on the role of a medieval warrior and can interact with virtual objects in the scene, such as a sword and a shield with a Duke University icon, by grabbing and moving them. To enable this, the Quest 3 and the local device are connected via a Meta Quest Link cable. This setup allows the Unity-rendered game scene to be displayed on the Quest 3 and stabilizes its communication with the local device.

Obstruction Detection Manager: The obstruction detection manager consists of a communication module and an obstruction detection module, described in Sect. 3. When a user sends a detection command via the controller, the command is first received by Unity, which forwards it to the communication module of the obstruction detection manager. The detection process begins by capturing images displayed on the Oculus casting webpage, including both raw and AR views. These images are processed by the obstruction detection module, which identifies obstructions of important real-world objects, especially those related to users' safety such as a stop sign shown in Fig. 1, and outputs detection results. Based on the results, the communication module sends scene management commands back to Unity, ensuring the virtual content is properly adjusted.

{yanming.xiu[1], maria.gorlatova[2]}@duke.edu

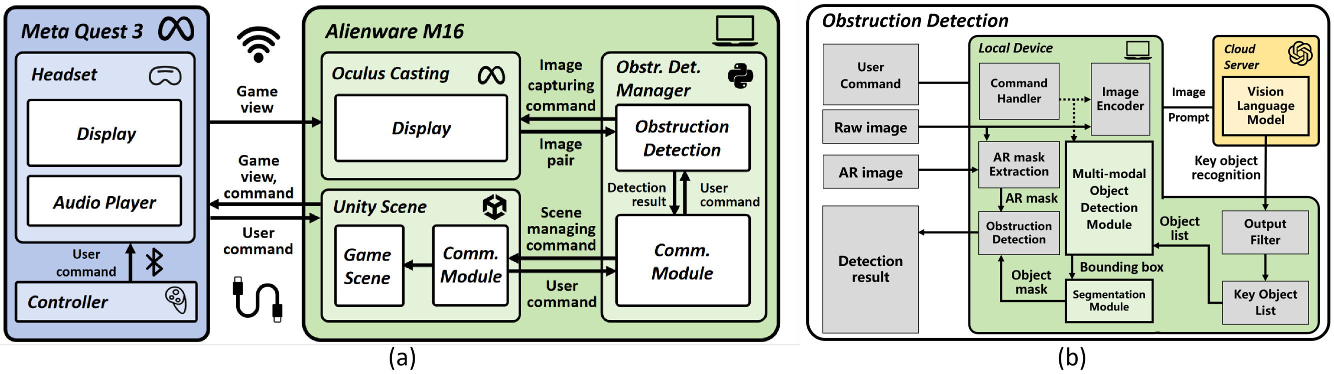

Figure 2: System architecture of our proposed system. (a) Overall system architecture; (b) detailed structure of obstruction detection module.

2.3 Cloud Server

The cloud server is an integral part of the obstruction detection module. It hosts a VLM to enhance the system's capability for scene understanding. To be more specific, the VLM processes an encoded image and outputs the recognized key objects within it. In this implementation, we use the GPT-4v by OpenAI [5] as the VLM.

3 System Implementation Details

In this section, we provide some details in implementing our system:
Obstruction Detection Module: The detailed design of the proposed obstruction detection module, illustrated in Fig. 2(b), builds on the ViDDAR framework [6] we recently built. The module continuously listens to the command from the user. To detect obstruction, the module takes in both the raw image which does not contain any virtual content, and the AR image. Pixel-wise comparisons generate a virtual content mask, and upon user request, the raw image is sent to the cloud server for analysis. The server identifies key objects and updates the key object list. Simultaneously, each raw image and the current key object list are further processed by the multi-modal object detection module, GroundingDINO [3], to produce bounding boxes, which are further refined into binary masks using a segmentation module [2]. These masks are then compared with virtual content masks to detect obstructions. If a certain percentage of a key object, defined by a configurable threshold, is overlaid by virtual content, the system classifies the result as "obstructed."
Raw Image Capturing: As discussed above, the obstruction detection module requires both raw and augmented images to detect obstructions. However, by casting the user's view to a webpage, only augmented images are available. To obtain raw images, when an obstruction detection command is received, all virtual content in the scene is temporarily inactivated for 0.05 seconds. During this window, 15 consecutive frames of the casting screen area are captured, ensuring both raw and augmented images are included within these frames. To distinguish between the raw and augmented images, we calculate the Mean Squared Error (MSE) for each pair of consecutive frames. A significantly large MSE indicates a transition between raw and augmented images. If only one large MSE value is observed, it implies the virtual content disappeared but did not reappear during the capture, with the first image being augmented and the second raw. Conversely, if two large MSE values are detected, it indicates the virtual content disappeared and reappeared, making the first image in the first pair augmented and the second raw.
User Interactive Elements: The system provides intuitive user interactions to enhance usability and feedback. Users can grab virtual content with the grip button on the controllers, trigger obstruction detection with button A, and send commands to identify key objects within the scene and update the key object list with button B. Once an obstruction is detected, the virtual content in the scene is made transparent, and red warning text is displayed to warn the user. Additionally, audio cues are employed to inform users of key events: starting obstruction detection, initiating key object identification, detecting an obstruction, confirming no obstructions, or updating the key object list. This ensures users remain aware of system updates without relying solely on visual feedback.

4 Future Work

The primary challenge in our work is the limited access to the camera feed, a common restriction among video see-through HMDs. This prevents direct access to raw images, requiring reliance on an image-capturing method to identify obstructions. Consequently, obstruction detection is performed based on user-initiated commands rather than continuously across every frame. While recent announcements from the industry suggest that expanded access to camera feeds may soon be available, we will continue to advance our research on existing platforms, such as mobile AR devices, to refine our proposed detection method. Once camera access becomes available on AR HMDs, our system will be prepared to incorporate real-time obstruction detection, enhancing operational efficiency and user experience.

Acknowledgments

This work was supported in part by NSF grants CSR-2312760, CNS-2112562 and IIS-2231975, NSF CAREER Award IIS-2046072, NSF NAIAD Award 2332744, a CISCO Research Award, a Meta Research Award, Defense Advanced Research Projects Agency Young Faculty Award HR0011-24-1-0001, and the Army Research Laboratory under Cooperative Agreement Number W911NF-23-2-0224. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the Army Research Laboratory, or the U.S. Government. This paper has been approved for public release; distribution is unlimited. No official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] K. Cheng, J. F. Tian, T. Kohno, and F. Roesner. Exploring user reactions and mental models towards perceptual manipulation attacks in mixed reality. In *Proceedings of USENIX Security*, 2023.

[2] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[3] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. *arXiv:2303.05499*, 2023.

[4] Meta. Oculus casting. [Online]. Available: `https://www.oculus.com/casting/`, n.d. Accessed: 2025-01-12.

[5] OpenAI. GPT-4 technical report. *arXiv:2303.08774*, 2023.

[6] Y. Xiu, T. Scargill, and M. Gorlatova. ViDDAR: Vision language model-based task-detrimental content detection for augmented reality. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2025.