XaiR: An XR Platform that Integrates Large Language Models with the Physical World

Sruti Srinidhi*
Carnegie Mellon University

Edward Lu*
Carnegie Mellon University

Anthony Rowe*
Carnegie Mellon University
Bosch Research

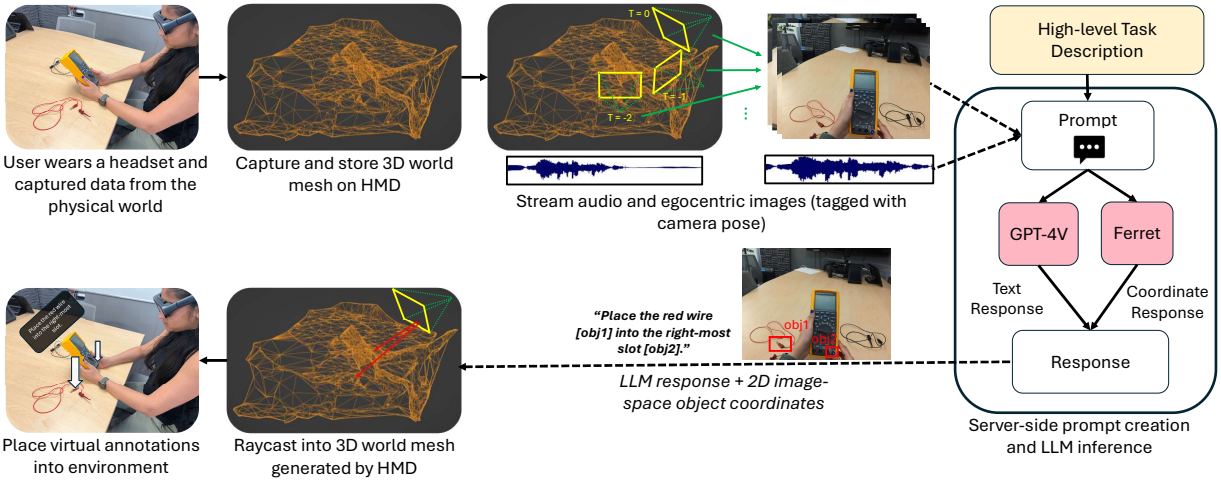

Figure 1: Overview of XaiR data flow: The figure illustrates the usage of Multimodal Large Language Models to automatically place and generate XR content. It shows the process from capturing egocentric inputs on a HMD to placing precise AR content in physical environments, highlighting the cognitive assistant's role in real-time task guidance.

Abstract

This paper discusses the integration of Multimodal Large Language Models (MLLMs) with Extended Reality (XR) headsets, focusing on enhancing machine understanding of physical spaces. By combining the contextual capabilities of MLLMs with the sensory inputs from XR, there is potential for more intuitive spatial interactions. However, the integration faces challenges due to the inherent limitations of MLLMs in processing 3D inputs and their significant resource demands for XR headsets. We introduce XaiR, a platform that facilitates integrating MLLMs with XR applications. XaiR uses a split architecture that offloads complex MLLM operations to a server while handling 3D world processing on the headset. This setup manages multiple input modalities, parallel models, and links them with real-time pose data, improving AR content placement in physical scenes. We tested XaiR's effectiveness with a "cognitive assistant" application that guides users through tasks like making coffee or assembling furniture. Results from a 15-participant study shows over 90% accuracy in task guidance and 85% accuracy in AR content anchoring. Additionally, we evaluate MLLMs against human operators for cognitive assistant tasks which provides insights into the quality of the captured data as well as the current gap in performance for cognitive assistant tasks.

1 Introduction

The integration of eXtended Reality (XR) with Artificial Intelligence (AI) has long been a dream for XR enthusiasts. Recent advancements in Multimodal Large Language Models (MLLMs) and XR headset technologies have made the integration of these two domains increasingly feasible. In this paper, we address several pivotal questions: How can XR headsets transmit egocentric physical information to MLLMs? Can MLLMs apply human-like common sense to interpret complex spatial scenes? And how effective are we at providing functional cognitive assistance through these technologies? If successful, these types of systems can completely transform the level of abstraction used to program XR systems and unlock the potential to dramatically increase human efficiency.

To explore these questions, we introduce *XaiR*[1], a platform that bridges the gap between MLLM inference and real-world environmental contexts. The high-level workflow is shown in Fig. 1 with sensor data as input, going through various LLM processing steps and resulting in AR content anchored to object locations in the scene. Unlike typical LLM applications on head-mounted devices (HMDs) that predominantly handle verbal queries [1, 7], XaiR integrates a live 3D world mesh to enhance the situational context and support AR annotations, allowing a more immersive user interaction that extends beyond simple question-and-answer tasks. 3D world data is important for many tasks where components might be out of the current view within the scene. For example, imagine a system that trains users to do a task that requires objects that are stored away from the user's view.

Currently, LLMs are primarily trained on text and image data from the Internet, making it difficult to integrate the crucial 3D spatial data needed for XR applications. This means that while some MLLMs can process 2D images and videos [16, 17, 25, 32, 34, 36], their ability to handle 3D data remains uncertain. A possible solution might involve training models directly on 3D datasets, although acquiring the necessary large-scale data presents its own challenges. XR headsets, constrained by limited memory and processing power, are ill-suited for running resource-intensive MLLMs natively. Alternative approaches involve smaller, mobile-optimized

*e-mail: {ssrinidh, elu2, agr}@andrew.cmu.edu

[1]Pronounced "x-air", we insert (a little) AI into XR experiences using the cloud (X→ai←R)

2024 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) | 979-8-3315-1647-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISMAR62088.2024.00091

979-8-3315-1647-5/24/$31.00 ©2024 IEEE
DOI 10.1109/ISMAR62088.2024.00091

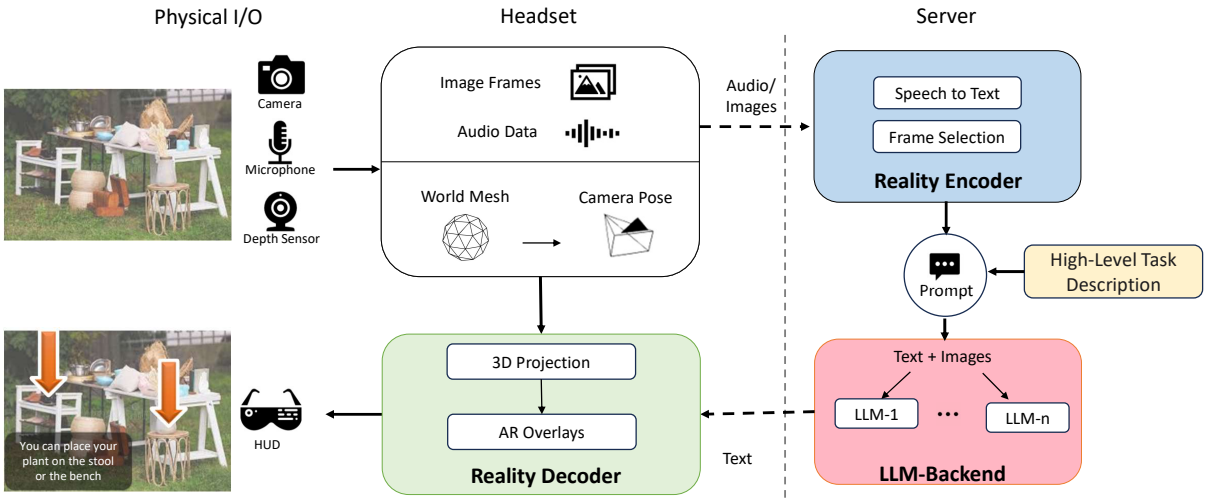

Figure 2: High-level XaiR system architecture and dataflow. The figure shows the different components of the system, which device they run on, and the different types of data captured and transmitted over the network.

LLMs [35], but these often sacrifice accuracy for performance. As a result, many LLM-based applications depend on a cloud-based architecture to offload intensive computations [1, 11].

XaiR employs a similar split processing architecture to address the limited spatial comprehension of MLLMs and the restricted processing capabilities of XR headsets. It offloads complex operations such as MLLM inference to a nearby server, while the headset handles 3D world mesh operations locally. The system efficiently manages the concurrent execution of multiple models that process egocentric images, speech, and text input, all synchronized with time-stamped pose data within the world mesh. This separation of processing tasks allows the system to appropriately project AR content into the scene, aligning the output text with the user's time, location, and orientation. Using these data and the LLM text response, virtual content is integrated into the world through raycasting from recorded camera poses into the 3D map produced by the headset. Our system consist of three main components: the *Reality Encoder*, which gathers data about the physical environment (pose timestamped images and audio); the *LLM-backend*, which synthesizes output from various specialized MLLMs; and the *Reality Decoder*, which refines text outputs and overlays AR annotations. A *High-Level Task Description* is used as a prompt to guide the LLM-backend to customize the system for specific tasks (see Fig. 2).

We evaluate our platform using a "cognitive assistant" application, which aims to guide users through detailed physical tasks using AR annotations. This application addresses tasks that involve sequential steps, such as guiding users through the assembly of industrial machinery or instructing someone on using a new coffee machine. The cognitive assistant application operates in two main phases. First, an expert user walks the system through the task and asks the MLLM to generate instructions from an egocentric videotream. Second, during a live session, the system guides a novice user through the same task using the pregenerated instructions as a High-Level Task Description and the captured egocentric images as references.

Our evaluation involves a study designed to compare the performance of MLLMs with that of a human guide operating in real time on the same data feed at a remote computer, establishing a human baseline. The participants performed various physical tasks, some guided by our MLLM system and others by a human back-end. We evaluated the effectiveness of each guide using metrics such as task completion time, number of user queries, and the NASA Task Load Index (TLX) [12] to measure cognitive load. In addition, we analyze how the type and quantity of input data influence the efficiency and accuracy of the MLLM to provide step-by-step instructions. In general, the human outperformed the MLLMs, but the difference in performance was at times very close.

In summary, this paper contributes the following:

- An open-source system for developing XR headset applications that integrate spatial data with MLLMs[2].
- A comparative study on the effectiveness of MLLMs in supporting physical tasks against human benchmarks.
- Insights into the optimal types and amounts of data needed for MLLMs to effectively support real-world tasks.

2 Related Work

We split our related work into two categories: Large Language Models and their integration into XR, and prior work with instruction following-based cognitive assistants.

2.1 XR Large Language Models

With the development of LLMs, we see an increase in models that are both general-purpose and have human-like understanding and responses [10, 21]. With LLMs now having multitmodal capabilities, they can also perform zero-shot inference on images and audio [28, 30]. This naturally leads to an integration into AR applications, where their advanced capabilities can enhance user experiences and interactions in immersive environments.

However, these powerful LLMs are resource-intensive and cannot run efficiently on mobile AR devices due to their high computational and memory requirements. As advancements continue in the development of general-purpose LLMs, considerable efforts have been directed towards enabling their operation on mobile platforms. These models are undergoing quantization and sparsification, as evidenced by works such as Zhang et al.'s TinyLLAMA [35] and Hsieh et al.'s Distilling Step-by-Step [13], enabling deployment on devices like mobile XR headsets. While this improves memory usage, lowers latency, and reduces network demands, it does so at the expense of decreased descriptiveness, accuracy, and general applicability [14]. On the other hand, robust LLMs such as GPT-4 [21], Llama [5] and LLaVA [18] boast billions and trillions of parameters and are trained on vast datasets, enabling them to offer more detailed and imaginative responses across a broad spectrum of topics. However, their size renders them unable to run on edge devices, requiring them to be run on powerful servers. Recognizing

[2]Github Link: https://github.com/srutisrinidhi/XaiR

these limitations, our system performs LLM processing in the cloud while handling XR-related tasks on the headset, balancing performance and resource usage. We acknowledge that this approach increases latency, but we believe that as technology advances, latency will eventually decrease. The trade-off between latency and response quality is an area for future exploration, potentially leading to a system that can adapt to changing networks or application demands, similar to the approach in [19].

We will now discuses early work on connecting LLMs with physical data. OpenAI's GPT-4 omni [6] is a step towards creating a more natural interaction with an LLM with multimodal capabilities to understand the user's context. Xu's Penetrative AI [31] demonstrates how LLMs can apply common-sense style knowledge to sensor data to infer higher-level concepts. Our system applies similar techniques, but connects them more closely with egocentric data with XR outputs.

2.2 Instruction Following and Cognitive Assistants

We demonstrate potential of XaiR through an instruction-guiding "cognitive assistant" application that can be compared to a number of existing approaches. Despite significant strides in developing such instruction-following assistants in XR, many of these systems need extensive manual effort to craft XR instructions for each specific task [24, 33]. These processes are labor-intensive and lack adaptability across varying tasks. Moreover, these systems often offer fixed responses, lacking the capability to dynamically adjust guidance based on user interactions. Although the work by Chidambaram et al. [9] contributes to instruction generation and work by Stanescu et al. [23] detects task completion through object detection techniques, both still fall short in adapting guidance to the user's interactions. Commercial systems like Step Check [4] combine AI with XR for manufacturing inspections. While this automatically detects errors, it still requires significant effort to create each of the AI models for the specific manufacturing processes.

LLMs have demonstrated significant potential to power virtual assistants across various domains such as programming, personal tasks, and medical diagnosis [2, 15, 20, 27]. However, recent advancements in multimodal AI and perception techniques have ushered in a new era of "intelligent" assistants, capable of interpreting and responding to the physical environment with tailored responses. Meta's Ray-ban Smart glasses [1], for instance, leverage AI inference to analyze images captured by the glasses and can provide insights into the user's surroundings. Similarly, Google's Project Astra [7] can understand and respond in the context of the user's physical environment using visual and audio prompts. While these types of systems excel at perceiving the environment, they are limited to displaying a fixed, static output and lack the capability to dynamically project and anchor responses back into the environment.

Amidst discussions surrounding the development of LLM-powered assistants for physical tasks [8], our work represents an initial effort in realizing such a platform using current-day technologies. We aim to bridge the gap by creating a platform that integrates LLM capabilities with real-world interactions, laying groundwork for intelligent assistance that understand physical environments. While not the main focus of our work, the ability to rapidly prototype instruction following with just a few high-level instructional prompts is a dramatic departure from prior, more bespoke methods.

3 System Design

Fig. 2 shows the three main software components (Reality Encoder, LLM-backend, and Reality Decoder) that are partitioned between a client headset and a nearby server. The client is an Android application developed in Unity [26] running on a Magic Leap 2 [3],

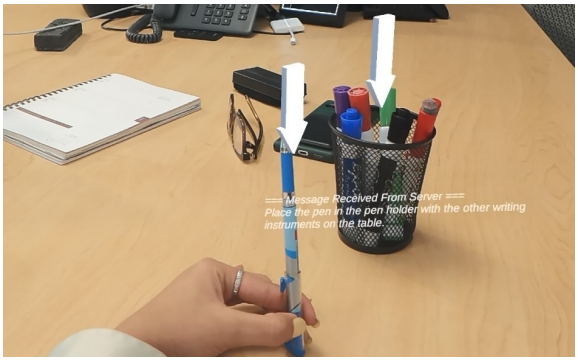

Figure 3: Example of virtual AR annotations overlaid on physical world, generated by XaiR as a response to a user question. Taken from screen capture of ML2. Through the headset, the 3D content is more vivid with segmented dimming turned on.

that can stream audio, color images, and depth data. The client locally constructs a 3D world model of the environment that is used to project objects found in the images into 3D locations. The server has significantly greater resources (we use a Linux workstation with dual RTX 3090 Ti GPUs connected via NVLink) that perform tasks like such as audio-to-text translation, MLLM inference on models up to 13 billion parameters, and text post-processing. This division of labor ensures that all 3D processing remains on the headset, while MLLM-related text and image processing are delegated to the powerful server.

3.1 The Reality Encoder

The Reality Encoder is responsible for converting audio to text and capturing frames of interest that should be passed to the LLM-backend. In our current implementation, data is transferred from the headset to the Reality Encoder running on the server (to aid in debugging), but this component can also be run on the client to reduce network overhead. We use OpenAI's Whisper model [22] to transcribe audio into text that is passed as user input to the MLLM. Audio data is continually received, decoded, and queued on the server for further post-processing. Identifying and filtering conversational components is an extremely hard problem in its own right. As a simple heuristic solution, we apply ambient noise adjustment and use a window of 3 seconds when listening for a query (i.e., we assume a query will not last for more than 3 seconds). In the cognitive assistant application, we augment the transcribed text query to include supplementary information like the current instruction set being executed.

Image frames from the ML2 are marked with a unique identifier, timestamp, and camera pose indicating where they were captured before transmission to the server. The camera pose is determined by ML2's tracking algorithms, calculating its position relative to the local origin of the tracking space. These frames are then added to the user text queries for contextual information and forwarded to an LLM beackend. The ML2 generates a 3D map periodically, storing it locally for later 3D content anchoring, using Magic Leap's `XRMeshSubsystem` in Unity. Since the user's head pose and environment are constantly changing, the 3D map is periodically updated at an approximate rate of 1Hz. This decision saves bandwidth and keeps 3D processing localized, benefiting from ML2's built-in functionalities. Timestamped poses enable variable-time processing, allowing locations labeled in the scene to be anchored back into the 3D world map once processing is complete.

The ML2 streams audio through WebRTC [29] and sends captured 640x480 resolution images using HTTP POST requests. To maintain optimal inference quality, images are compressed losslessly using PNG. As a default, images are sent at a 1Hz interval,

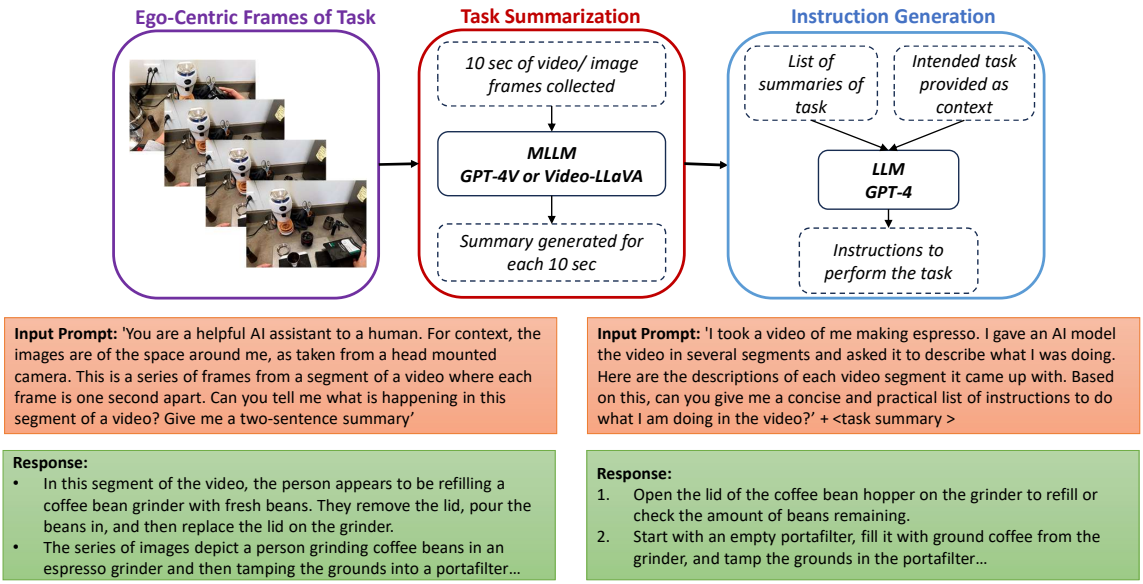

Figure 4: Detailed breakdown of the instruction generation pipeline for the cognitive assistant using the XaiR system. The figure shows example input prompts and an example output response from the MLLMs at each step.

while audio is constantly being streamed. We describe the bandwidth requirements for XaiR in Sec. 5.3.3. By sending data in this manner, the backend can be easily modular and interchangeable.

For our cognitive assistant demonstration application, we provide two image frames with each prompt so that we can give the MLLMs context about what is currently happening and a snapshot of what has previously happened in the user's surroundings. While we experimented with incorporating more than two frames into the MLLMs, we observed that it significantly slows down the system without any significant boost in accuracy of responses. We find that values tend to require application-specific tuning with accuracy and latency trade-offs explored in Sec. 5.2.

3.2 LLM-based Backend

After the reality encoder step, the generated prompt is sent to our LLM-backend. Since the server has ample computational resources compared to a headset, we provide an API that enables the execution of multiple MLLMs in parallel along with potential remote cloud calls. In our baseline system, we concurrently use GPT-4V [36] on OpenAI's servers and Ferret [32]. Ferret is a specialized MLLM, which can identify objects, discern relationships between multiple regions in an image, and provide 2D bounding boxes of object locations. Despite Ferret's superior spatial understanding, we rely on GPT-4V for reasoning, since Ferret lacks the advanced reasoning capabilities of GPT-4V. The two models are queried at the same time and our server combines the responses of both models, using GPT-4V's response for textual feedback and Ferret's response for object locations which are used to anchor AR content. Thus our inference time is limited by the slowest MLLM, which is typically GPT-4V. This overhead can potentially be reduced with faster models like GPT-4o.

3.3 The Reality Decoder

The Reality Decoder is responsible for packaging the output of the LLM-backend into an intuitive visual interface. We use a simple language for AR annotations provided by our prompting system that can draw graphical primitives anchored at 3D locations. Our prototype system has a small dictionary of arrows and text boxes, but this could easily be extended to include more complex models and eventually small scripted interactions.

One of the main challenges is converting 2D coordinates into 3D coordinates. For example, the 2D boxes generated by Ferret need to be projected into 3D in order to anchor AR content into the scene. The ML2 stores all the previous camera poses along with their associated image IDs in a lookup table. Once the ML2 receives a response from the server with an image ID, it looks up the associated camera pose. Using this pose, it raycasts into the stored 3D mesh to get the 3D coordinate of that object when the image was captured. The text output is also displayed on the screen for the user in AR. See Fig. 3 for a visualization. Because the headset internally tracks each virtual 3D object within the local tracking space, we do not have to continually process each image to update the object locations. One limitation is that since the objects are not being actively tracked after frames are captured, the anchors are rendered at fixed locations and will only be updated after processing new frames. This can cause significantly lag if objects are moving. One could imagine adding a post-processing step on the client that does continual tracking of certain objects that are known to be dynamic.

4 XaiR Cognitive Assistant Demonstrator

We evaluate our system as a cognitive assistant [8], specifically one that can walk non-expert users through step-by-step instructions and can leverage XR content to highlight important objects in the physical world. Our cognitive assistant application is divided into two main stages – instruction generation for a new task and live feedback mode for guiding users through the task.

4.1 Instruction Generation from Egocentric Videos

Fig. 4 shows how our system can perform automatic instruction generation from a single egocentric video. Wearing a headset, an expert user can record themselves performing a task and use XaiR to extract step-by-step text instructions. We leverage the ability of MLLMs to interpret text and visual data to generate a log of the expert's actions from an egocentric video stream and speech. The collected images provide additional visual context of how specific objects and tools are used for a task, data which is not always known by an off-the-shelf MLLM. We take frames from 10 second batches of the image stream and use GPT-4V to summarize the actions being performed in those sections as text. We then collect all the text and input it back into GPT-4V to generate holistic instructions

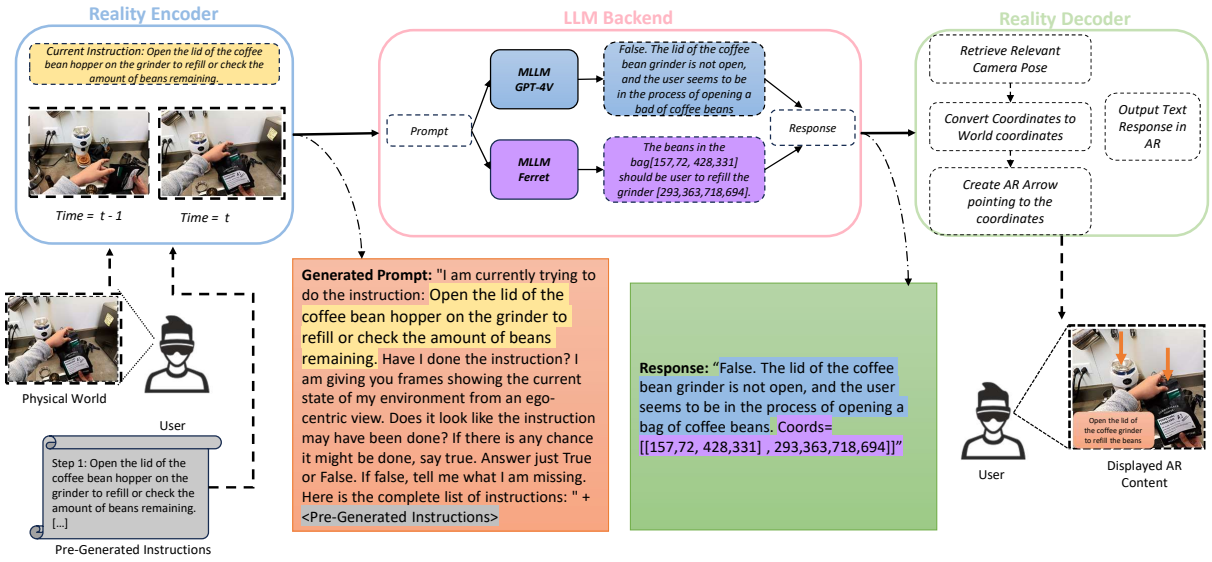

Figure 5: Walkthrough of the instruction following pipeline for the cognitive assistant. The process uses instructions that are generated in the process details in Fig. 4. The figure shows in the intermediate prompts and MLLM responses that are generated

to perform the task for a non-expert user. This automates the instruction generation and user action logging process. Fig. 4 shows examples of our actual input prompts with the various responses generated by the system.

While MLLMs are quite capable of creating step-by-step instructions to do a task purely from textual input, using this video summarization and action logging method ensures the instructions generated are specific to the way the expert did the task and to the tools they used. For example, while GPT-4V can easily generate instructions to make a cup of coffee, the results are often very generalized and non-specific to the particular tools a user may have. Using our technique, the model will generate instructions that walk through and utilize the correct tools. This is especially useful when working with specialized devices (like in factories), or when a particular method of doing the task is important. It also simplifies instruction generation, since an expert does not have to write down the steps to do a task manually and risk missing details which the MLLM can pick up on, and removes the requirement for the LLM to be retrained in a particular task making our system very generalizable.

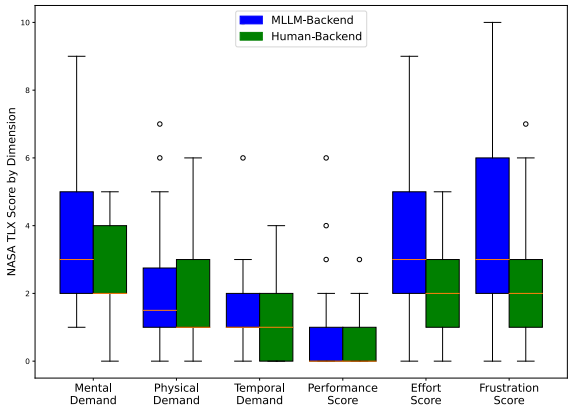

Figure 6: NASA TLX values are shown broken down by each of the individual dimensions. The figure compares the scores given by users during user studies for the MLLM-backend and Human-backend cognitive assistant.

4.2 Live Feedback when Performing a Task

When a non-expert user wants to perform the same task, XaiR can walk the user through the task using the instructions generated in the previous stage. In order to create the prompts for the MLLMs, we use the latest two image frames as well as the current instruction. The prompt asks the MLLM to provide the status of the current instruction by using the image frames as context for what the user has done. We also provide the entire instruction list as context in each prompt so the MLLM has information about the completed and upcoming tasks, improving understanding of the user state.

The ML2 overlays virtual arrows in the environment to highlight essential objects required for the task. Our system also generates text to display feedback needed to complete the task and what the user might be doing incorrectly. A detailed example of this can be seen in Fig. 5 with example prompts that are generated from the combination of the Reality Encoder and pre-generated instructions.

Throughout the process, the user has the ability to ask follow up questions which are also injected into the queries of GPT-4V and Ferret. This allows the user to ask clarifying questions or get additional information from the assistant.

5 Evaluation

In the following sections, we evaluate XaiR's effectiveness as a instruction-guiding cognitive assistant.

5.1 Reasoning Capabilities Compared to Humans

We conducted a user study to evaluate the performance of GPT-4V and Ferret against a human's performance in cognitive assistant tasks. We considered a human as the "ideal" model, setting them as a benchmark. The comparison assessed the ability of MLLMs to comprehend physical world data, interpret user actions from images, and offer task guidance relative to human capabilities.

To ensure equal information access between the human assistant and the MLLMs during our study, we developed a web-based dashboard, as shown in Figure Fig. 7. This dashboard displays five images per second, captured from the user's headset camera. The human assistant types responses and draws bounding boxes on the most recent image, similar to the functionality of the Ferret subsystem. The human's text responses includes feedback on current instructions and decisions about proceeding to the next task segment,

	Basic Tasks		Advanced Tasks	
	MLLM	Human	MLLM	Human
Accuracy (%)	86.7	100	93.3	100
Queries per instruction	8.05 ±2.98	2.52 ±0.43	10.72 ±3.80	3.88 ±0.49
Time per instruction (sec)	38.37 ±3.16	27.88 ±1.51	37.68 ±3.10	27.85 ±2.34
Time to complete task (sec)	166.26 ±15.81	139.4 ±7.55	228.38 ±13.63	162.93 ±12.73
NASA TLX Raw Scores (out of 100)	28.11 ±3.08	16.67 ±2.06	33.00 ±4.41	17.78 ±2.62

Table 1: User Study results comparing the performance of the MLLM-backend with the human-backend for the cognitive assistant with "basic" and "advanced" tasks. The values show an average over the 15 participants' results. For the TLX scores, a lower average is a better.

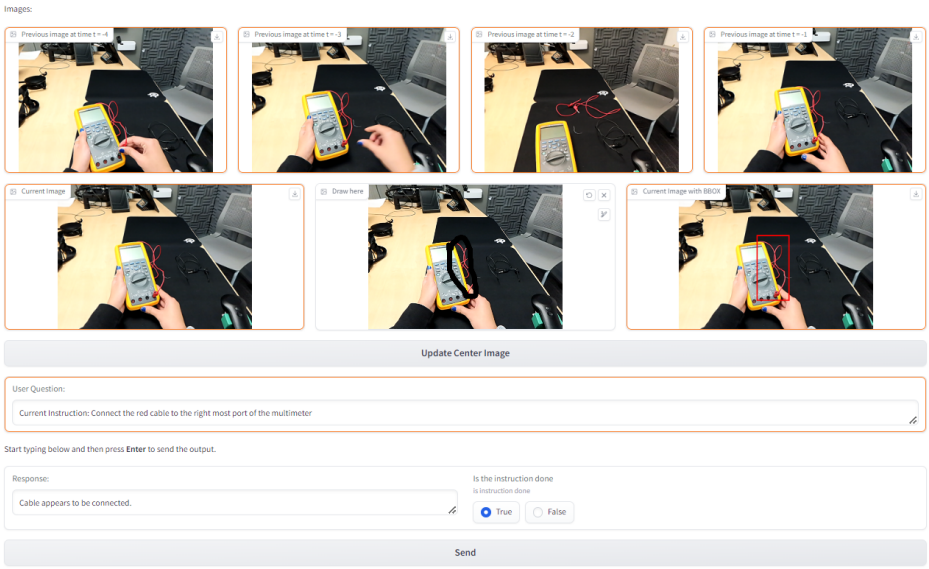

Figure 7: Interface for the human-backend dashboard. It shows five previous frames in a sequence and includes a sketchpad to annotate the location(s) for AR overlays. It also includes the user's input prompt, a textbox to respond, and a checkbox to decide whether the current step as been completed.

similar to the functionality of the GPT-4V subsystem. Both the human and the MLLMs were provided with the same data – identical image sets and text prompts – to maintain a fair comparison. The human reasoning agent was selected from among the authors and remained the same across all the trials conducted to ensure consistent typing and annotation speeds, in an attempt to eliminate the specific human reasoning agent as a bottleneck across the studies.

Our study involved 15 participants between the ages 22 and 45 with a 46% female/male gender ratio. Each participant was tasked to complete a total of four tasks: two "basic" and two "advanced" tasks. The basic tasks required participants to assemble simple children's blocks into various configurations based on provided instructions. The advanced tasks involved setting up a humidifier and using a multimeter to measure the resistance of a through-hole resistor. We randomly assigned one task from each category to be guided by GPT-4V and Ferret MLLMs, while the other task was guided by a human through a web dashboard. Participants were not informed that there were different guiding agents until after the study was completed. For each task, we recorded the completion time, the number of queries made, and user experience using the NASA Task Load Index (TLX). The results are in Tab. 1.

We see that 3 out of the 30 tasks performed with the MLLMs were incomplete, meaning the MLLMs did not advance from a step for more than 4 minutes even though the step had been completed, or the MLLMs falsely advanced from a step even though the step had not been completed. In these cases, we marked the task as incomplete. This gives the MLLM-based cognitive assistant a 90% overall accuracy whereas the human-based assistant had a 100% accuracy. We saw that participants needed to make more queries when interacting with the MLLMs when compared to the human before the system considered the task completed. We see an average of 8.04 queries per instruction for basic tasks with the MLLM-backend while the human-backend only needed 2.52 queries. The MLLMs requires about 6-7 more queries for each instruction when compared to the human, which shows that the MLLM responses are not always accurate (p-value from T-test $= 1.56\text{x}10^{-5} < 0.05$). We observe that if the system does not detect that the instruction has been completed, the participants re-position the objects in front of them, or try to view the objects from a different angle; the participants naturally assumed that the assistant could not "see" the completed step and thus moved objects around to account for that.

While the human-based cognitive assistant is faster at helping the user complete the tasks, we see that the MLLMs take 19% more time for the basic tasks and around 40% more time for the advanced tasks, showing that the reasoning speeds of MLLMs today are not that far off from those of humans. We observed that the MLLM-based trials spent the majority of the time making multiple calls to the MLLM, whereas the human took longer to look at the images, identify and locate the relevant objects, and type a text response. While we initially thought that the human time would be signif-

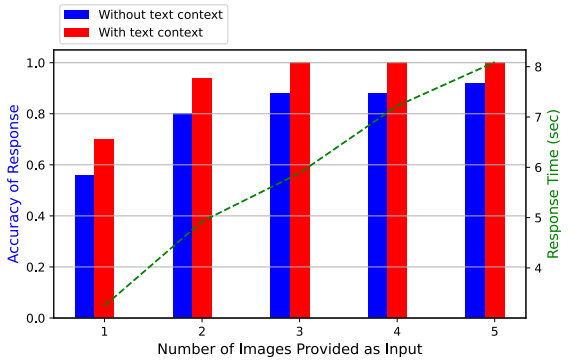

Figure 8: Accuracy and speed of GPT-4V when different amounts of input data are provided as context. Shows the trade off between response accuracy and response time.

icantly faster, we were surprised to only see an average of 0.5-1 second difference per query (p-value from T-test = 0.0065 $<$ 0.05).

To assess user experience, we used the NASA Task Load Index (TLX) to measure user workloads with both types of cognitive assistants. Tab. 1 shows a consistently higher average TLX scores for the MLLM-based assistants compared to the human-based ones, with lower scores indicating lower cognitive load. Further analysis of individual components of the NASA TLX in Fig. 6 shows that the MLLM-backed system exhibits a higher task load index than the human-backed system. This suggests that the MLLM-based cognitive assistant demands a higher mental workload from users. Nonetheless, the performance metrics are nearly identical, indicating that both systems are capable of achieving the task goals, albeit with different mental workloads required from the users.

5.2 Comparing the Value of the Data Input to the System

The purpose of the Reality Encoder is to gather essential data to allow MLLMs to better understand the user's environment. To determine how much context we needed to provide GPT-4V to achieve the most optimal performance (in terms of accuracy and speed), we experiment with providing GPT-4V with varying levels of context, including different numbers of images and textual information. In this experiment, the number of image frames provided ranged from one to five, with each frame captured one second apart during the task. Additionally, we vary the amount of textual context provided. For instance, we provide textual context indicating that the user is trying to making coffee for half of the trials, while in the other half, no textual context is given. The effects of these variations on the accuracy of task prediction is shown in Fig. 8. We observed that as the number of image frames provided to the MLLM increases, the response time increases. To balance maintaining a low system latency while achieving a satisfactory level of accuracy, we concluded that using two image frames accompanied by textual context for the Reality Encoder as detailed in Sec. 3.1 was the most optimal amount of context for our use case.

5.3 System Performance

5.3.1 Accuracy of AR overlays

During the Reality Decoder step Sec. 3.3, Ferret returns the 2D image space coordinates of where we should place an virtual overlay raycasted into 3D. In order to measure the accuracy of Ferret, we ask a verbal question to the cognitive assistant about the user's surroundings about the objects in the space, like "where is the pen?," and then visually evaluate the resulting overlays. For our evaluation, we ask three different questions of varying levels of difficulty and repeat each questions 30 times and record the visually determined success rate. Tab. 2 shows the resulting accuracy of AR overlays for each of the questions. We see that as the complexity of the question increases (in other words, as the question goes from simply finding a known object in the scene to understanding what the purpose of the object is and knowing where to put it), the accuracy of responses reduces.

Question	Answering Question Requires:			AR Overlay Accuracy
	Identify Object from Image?	Identify Object from Description?	Find Object?	
Q1: Where is the pen?	No	No	Yes	93.3%
Q2: What can I use to write on the white board?	No	Yes	Yes	83.3%
Q3: Where should I put this [pen]?	Yes	Yes	Yes	76.7%

Table 2: Accuracy of AR overlays for questions that involve different types of comprehension – identifying what an object is, finding an object in the scene from the description, and identifying the object given a description.

5.3.2 Accuracy of Instruction Generation

As described in Sec. 4.1, our cognitive assistant can summarize and understand the user's actions from the egocentric video of an expert user performing a particular task. To test the effectiveness and accuracy of the MLLM in generating correct and clear instructions, we have the user perform five every day tasks and generate instructions from them. The tasks are of varying levels of complexity – setting up a humidifier, making coffee an espresso machine, making a sandwich, setting up a dinner table, and soldering a wire.

We use GPT-as-a-judge [37] to compare the generated instructions to ground truth instructions generated by the authors. Our procedure is as follows: we take egocentric videos, extract images, and generate instructions using GPT-4V as described in Sec. 4.1. Then we give GPT-4 both the generated instructions and ground truth instructions and ask it how many of the generated instructions match the ground truth, how many instructions are wrong, and how many additional instructions that are unspecified yet not wrong. Due to the nondeterministic nature of instruction generating, we repeat this procedure 10 times and average the returned values to get the number of correct, wrong, and additional instructions.

We ran the same experiment with Video-LLaVA [16] as the instruction generation agent instead of GPT-4V, as Video-LLaVA can interpret videos directly rather than a collection of images. We use the same egocentric videos we used for generating the instructions with GPT-4V and repeat the procedure described above. The results for instruction generation with the models are shown in Fig. 9. We notice that the accuracy values are in the 50-60% range for the GPT-4V, given that the wrong instructions are below 20% for the majority of times, the overall generated instructions are quite realistic. We noticed that LLaVA performs worse than GPT-4V in generating correct instructions, mostly because it has been trained on much less data. GPT-4V is rumored to have around 1 trillion parameters while LLaVA has only 3 billion, making GPT-4V the obvious choice for better accuracy and response quality. Thus, even though Video-LLaVA is open-source and faster, we decided to use GPT-4V as our MLLM for inference on image frames for our cognitive assistant system.

5.3.3 System Benchmarks

XaiR streams audio and image frames over the network from the client to the server every second. The server streams text back to

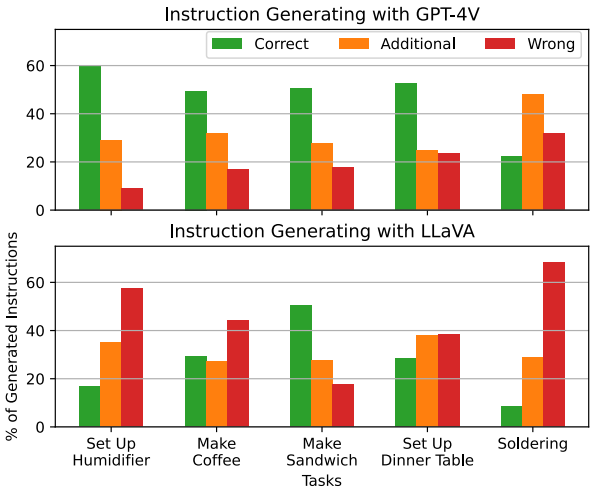

Figure 9: Analysis of instructions generated from egocentric videos of different tasks performed by a user using **GPT-4V** and **LLaVA**.

the client. Given that the text and audio has a negligible data size compared to images, we measure the approximate bandwidth consumed primarily based on the image frame streaming needs. Each frame is a 640x480 PNG compressed image. The average bandwidth consumed during our study is approximately **2.32 Mbps**.

System Components	*Time Taken (s)*
End to End System	4.241
MLLM Backend	4.173
GPT-4V	4.169
Ferret	2.883
Reality Encoder and Decoder	0.067
Image Streaming	0.005
RayCasting	0.001

Table 3: Average latency of various key system components.

To measure the end-to-end latency of the system, we measure the average time from when the images are captured on the headset to when the AR objects are generated. This takes roughly **4.241 sec** on average. See Tab. 3 for a detailed breakdown of the timing.

6 Discussion and Limitations

One limitation of the XaiR platform is its ability to create accurate and complex augmented reality (AR) overlays. We employ Ferret to generate 2D bounding boxes for objects. However, Ferret, with its 13 billion parameters and training on only 1.1 million spatial grounding data points, lacks the robustness of larger models like GPT-4V, which is rumored to have around 1 trillion parameters. Consequently, Ferret's accuracy and capability to detect complex objects are less reliable. Moreover, Ferret is limited to detecting objects that are visibly present in the scene; it cannot perform more intricate spatial detections necessary for following instructions in tasks. For instance, while it can identify and provide coordinates for a pen and a book based on the instruction to "move the pen to the right of the book," it cannot determine the coordinates for "the right of the book." Hence, our AR overlays are currently restricted to simply highlighting relevant objects but not demonstrating how to perform tasks with more complex animations, as we often lack precise spatial locations for such animations. In the future, MLLMs improve in spatial understanding, we anticipate significant advancements in AR content generation for systems similar to XaiR.

Additionally, the Magic Leap 2 headset, while capable of generating a 3D mesh of the environment, faces limitations that affect the projection of our 2D coordinates into the 3D world. The process of reconstructing this mesh from the cameras and depth sensors is notably slow (the fastest update rate is approximately 1Hz). As a result, AR overlays produced within the first 30-40 seconds of program operation often exhibit poor depth values due to the incomplete mesh reconstruction. Furthermore, while the mesh generation is more effective for large objects, the limitations of depth cameras mean that smaller, more detailed objects tend to result in a noisier and incorrect mesh. Consequently, AR overlays for smaller objects can sometimes be inaccurate, due to issues with the mesh and/or errors from Ferret's object detection.

A known issue with large language models (LLMs), due to their extensive number of parameters, is their slow response time, often taking several seconds for inference. This delay introduces significant latency to systems, particularly noticeable in conversational assistants where users might wait 4-6 seconds for a full completed response. This lag is especially problematic in tasks that involve following instructions, as the user must wait several seconds after completing an action before the LLM even processes images of the completed task. Consequently, the system's speed is inherently limited by the speed of the LLMs it employs which we imagine will decrease over time.

There has been a recent surge in industry efforts like those from Google [7] and OpenAI [6] to integrate physical world data into MLLMs to help users better understand the spaces around them. Their work mainly focuses on improving how a MLLM can answer questions about the user's surroundings using only audio/text and color images. Our reality encoder/decoder architecture for scene understanding generalizes this workflow to include mesh data/world information (and possibly other sensor modalities in the future). We provide an open-source architecture with scaffolding that allows developers to experiment with MLLMs in the context of XR applications. Additionally, as researchers and engineers improve MLLMs and machine learning models, these new models can easily be integrated into our system in parallel to enhance performance and response quality. With our reality encoder and decoder system, we provide both the ability to understand as well as ground responses in the physical world, without having to modify the MLLMs, making it easy to test new models for their ability respond to physical world data. We also believe that our MLLMs vs human user-study methodology could serve as an important tool for evaluating future systems and creating offline benchmarks.

7 Conclusion

In conclusion, this paper introduces XaiR, a research platform that integrates MLLMs with XR, showing promise for enhancing machine understanding of physical environments. XaiR uniquely facilitates the concurrent use of multiple MLLMs within physical contexts by employing a computational split, where resource-intensive MLLM operations are processed on a server, and processes requiring the world model are managed directly on the XR headset. We demonstrated the system through a cognitive assistant application and conducted a user study to evaluate the differences in MLLM and human operators in terms of task completion rate and time. The results indicated that while MLLMs may not match human levels of understanding and responding to the physical world, they often process tasks faster than human operators. We believe that this framework could be used to generate datasets that could inevitably be used to train and evaluate future model progress.

Acknowledgments

This work was supported in part by the National Science Foundation under Grant No. CNS-1956095, CNS-2148367, the NSF Graduate Research Fellowship under DGE-2140739, and Bosch Research. Any opinion, findings, and conclusions or recommendations expressed in this material are the author's and not the NSF.

REFERENCES

[1] Meta ray-ban glasses multimodal ai. https://about.fb.com/news/2023/09/new-ray-ban-meta-smart-glasses/. Accessed: Jan 2, 2024. 1, 2, 3

[2] Github copilot. https://github.com/features/copilot/, 2023. Online. Accessed: May 2024. 3

[3] Magic leap. https://www.magicleap.com/en-us/, 2023. Online. Accessed: May 2024. 3

[4] Jul 2024. 3

[5] Llama 3. https://ai.meta.com/blog/meta-llama-3/, 2024. Online. Accessed: July 2024. 2

[6] Openai gpt-4o. https://openai.com/index/hello-gpt-4o, 2024. Online. Accessed: July 2024. 3, 8

[7] Project astra google deepmind. https://deepmind.google/technologies/gemini/project-astra/, 2024. Online. Accessed: July 2024. 1, 3, 8

[8] L. Cheng. Introducing copilot in microsoft dynamics 365 guides, Nov 2023. 3, 4

[9] S. Chidambaram, H. Huang, F. He, X. Qian, A. M. Villanueva, T. S. Redick, W. Stuerzlinger, and K. Ramani. Processar: An augmented reality-based tool to create in-situ procedural 2d/3d ar instructions. In *Proceedings of the 2021 ACM Designing Interactive Systems Conference*, DIS '21, p. 234–249. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3461778.3462126 3

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 2

[11] J. A. V. Fernandez, J. J. Lee, S. A. S. Vacca, A. Magana, B. Benes, and V. Popescu. Hands-free vr, 2024. 2

[12] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Human mental workload*, 1(3):139–183, 1988. 2

[13] C.-Y. Hsieh, C.-L. Li, C.-K. Yeh, H. Nakhost, Y. Fujii, A. Ratner, R. Krishna, C.-Y. Lee, and T. Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes, 2023. 2

[14] V. Jain, L. Mei, and M. Verhelst. Analyzing the energy-latency-area-accuracy trade-off across contemporary neural networks. In *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 1–4, 2021. doi: 10.1109/AICAS51828.2021.9458553 2

[15] C. Li, C. Wong, S. Zhang, N. Usuyama, H. Liu, J. Yang, T. Naumann, H. Poon, and J. Gao. Llava-med: Training a large language-and-vision assistant for biomedicine in one day. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds., *Advances in Neural Information Processing Systems*, vol. 36, pp. 28541–28564. Curran Associates, Inc., 2023. 3

[16] B. Lin, Y. Ye, B. Zhu, J. Cui, M. Ning, P. Jin, and L. Yuan. Video-llava: Learning united visual representation by alignment before projection, 2023. 1, 7

[17] H. Liu, C. Li, Y. Li, B. Li, Y. Zhang, S. Shen, and Y. J. Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. 1

[18] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning, 2023. 2

[19] V. V. R. M. K. Muvva, K. Samal, J. M. Bradley, and M. Wolf. A closed loop perception subsystem for small unmanned aerial systems. In *AIAA SCITECH 2023 Forum*, p. 2673, 2023. 3

[20] D. Nam, A. Macvean, V. Hellendoorn, B. Vasilescu, and B. Myers. Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24. Association for Computing Machinery, New York, NY, USA, 2024. doi: 10.1145/3597503.3639187 3

[21] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, and et al. Gpt-4 technical report, 2024. 2

[22] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever. Robust speech recognition via large-scale weak supervision, 2022. 3

[23] A. Stanescu, P. Mohr, M. Kozinski, S. Mori, D. Schmalstieg, and D. Kalkofen. State-aware configuration detection for augmented reality step-by-step tutorials. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 157–166, 2023. doi: 10.1109/ISMAR59233.2023.00030 3

[24] K. Tainaka, Y. Fujimoto, M. Kanbara, H. Kato, A. Moteki, K. Kuraki, K. Osamura, T. Yoshitake, and T. Fukuoka. Guideline and tool for designing an assembly task support system using augmented reality. In *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 486–497, 2020. doi: 10.1109/ISMAR50242.2020.00077 3

[25] G. Team, R. Anil, and S. B. et al. Gemini: A family of highly capable multimodal models, 2024. 1

[26] Unity Technologies. Unity, 2005. Online. Accessed: May 2024. 3

[27] M. D. Vu, H. Wang, Z. Li, J. Chen, S. Zhao, Z. Xing, and C. Chen. Gptvoicetasker: Llm-powered virtual assistant for smartphone, 2024. 3

[28] W. Wang, Z. Chen, X. Chen, J. Wu, X. Zhu, G. Zeng, P. Luo, T. Lu, J. Zhou, Y. Qiao, and J. Dai. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds., *Advances in Neural Information Processing Systems*, vol. 36, pp. 61501–61513. Curran Associates, Inc., 2023. 2

[29] WebRTC Working Group. Webrtc, 2011. Online. Accessed: April 2023. 3

[30] J. Wu, W. Gan, Z. Chen, S. Wan, and P. S. Yu. Multimodal large language models: A survey. In *2023 IEEE International Conference on Big Data (BigData)*, pp. 2247–2256, 2023. doi: 10.1109/BigData59044.2023.10386743 2

[31] H. Xu, L. Han, Q. Yang, M. Li, and M. Srivastava. Penetrative ai: Making llms comprehend the physical world, 2024. 3

[32] H. You, H. Zhang, Z. Gan, X. Du, B. Zhang, Z. Wang, L. Cao, S.-F. Chang, and Y. Yang. Ferret: Refer and ground anything anywhere at any granularity, 2023. 1, 4

[33] J. Zauner, M. Haller, A. Brandl, and W. Hartman. Authoring of a mixed reality assembly instructor for hierarchical structures. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pp. 237–246, 2003. doi: 10.1109/ISMAR.2003.1240707 3

[34] H. Zhang, X. Li, and L. Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding, 2023. 1

[35] P. Zhang, G. Zeng, T. Wang, and W. Lu. Tinyllama: An open-source small language model, 2024. 2

[36] X. Zhang, Y. Lu, W. Wang, A. Yan, J. Yan, L. Qin, H. Wang, X. Yan, W. Y. Wang, and L. R. Petzold. Gpt-4v(ision) as a generalist evaluator for vision-language tasks, 2023. 1, 4

[37] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. 7