

agentAR: Creating Augmented Reality Applications with Tool-Augmented LLM-based Autonomous Agents

Chenfei Zhu*
School of Mechanical Engineering
Purdue University
West Lafayette, USA
zhu1237@purdue.edu

Ziyi Liu
School of Mechanical Engineering
Purdue University
West Lafayette, USA
liu1362@purdue.edu

Shao-Kang Hsia*
School of Mechanical Engineering
Purdue University
West Lafayette, USA
shsia@purdue.edu

Jingyu Shi
Elmore Family School of Electrical
and Computer Engineering
Purdue University
West Lafayette, USA
shi537@purdue.edu

Xiyun Hu*
School of Mechanical Engineering
Purdue University
West Lafayette, USA
hu690@purdue.edu

Karthik Ramani
School of Mechanical Engineering
Elmore Family School of Electrical
and Computer Engineering
Purdue University
West Lafayette, USA
ramani@purdue.edu

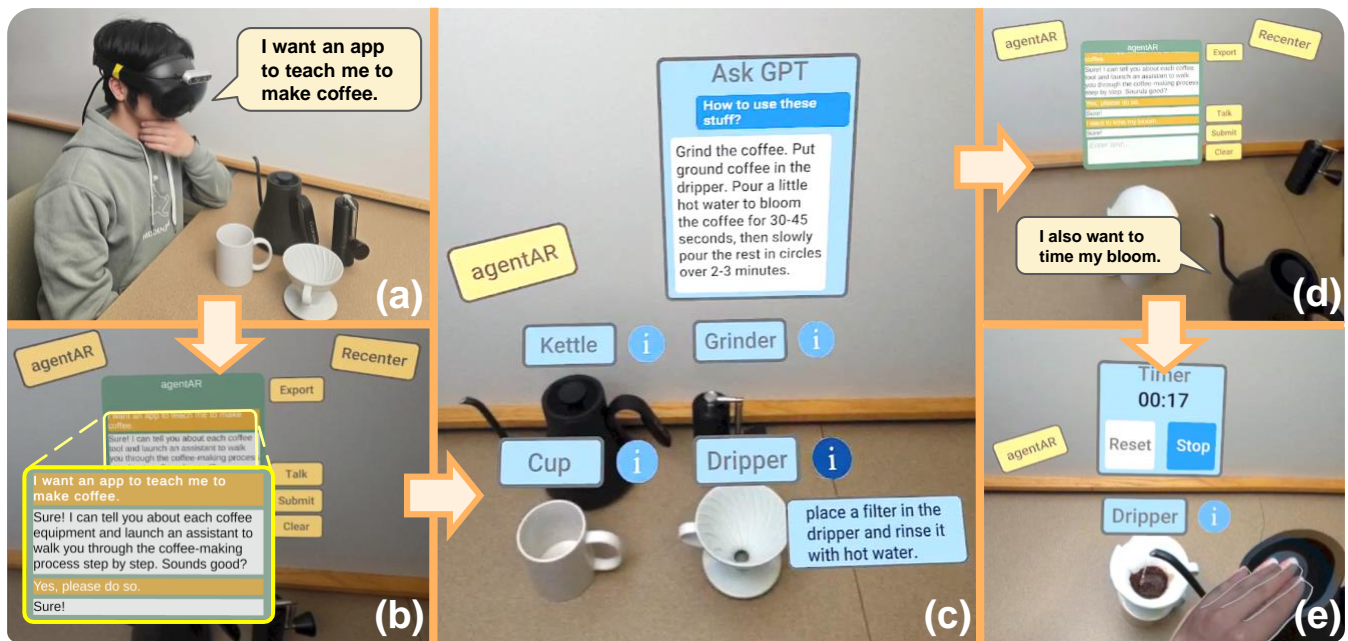


Figure 1: A walkthrough example of agentAR. (a) A user wants an AR application to teach him how to make coffee; (b) He clicks the agentAR button to activate the system and verbally requests it to create a coffee-making application; (c) agentAR creates an AR application that provides information about each coffee-making equipment and step-by-step brewing tutorials powered by GPT [63]; (d) The user then requests agentAR to add a timer to the application to track the bloom phase; (e) agentAR updates the AR application by adding a timer tool.

*Three authors contributed equally to this research.



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

UIST '25, Busan, Republic of Korea

© 2025 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2037-6/2025/09

<https://doi.org/10.1145/3746059.3747676>

Abstract

Creating Augmented Reality (AR) applications requires expertise in both design and implementation, posing significant barriers to entry for non-expert users. While existing methods reduce some of this burden, they often fall short in flexibility or usability for complex or varied use cases. To address this, we introduce agentAR, an AR authoring system that leverages a tool-augmented large language

model (LLM)–based autonomous agent to support end-to-end, in-situ AR application creation from natural language input. Built on an application structure and tool library derived from state-of-the-art AR research, the agent autonomously creates AR applications from natural language dialogue. We demonstrate the effectiveness of agentAR through a case study of six AR applications and a user study with twelve participants, showing that it significantly reduces user effort while supporting the creation of diverse and functional AR experiences.

CCS Concepts

• **Human-centered computing** → **Natural language interfaces; Interactive systems and tools; Interaction techniques; Mixed / augmented reality.**

Keywords

Large Language Model, Augmented Reality, Natural Language Interface, Autonomous Agent, Artificial Intelligence

ACM Reference Format:

Chenfei Zhu, Shao-Kang Hsia, Xiyun Hu, Ziyi Liu, Jingyu Shi, and Karthik Ramani. 2025. agentAR: Creating Augmented Reality Applications with Tool-Augmented LLM-based Autonomous Agents. In *The 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25), September 28–October 1, 2025, Busan, Republic of Korea*. ACM, New York, NY, USA, 23 pages. <https://doi.org/10.1145/3746059.3747676>

1 Introduction

Augmented Reality (AR) enhances perception, enriches user experiences, and fosters intuitive interactions by seamlessly integrating virtual content into physical environments. Its accessibility and immersive nature have led to widespread adoption in both general scenarios [14, 16, 35, 43, 76] and specialized domains such as sports [48, 52, 95] and education [12, 23, 27]. As AR becomes more prevalent in everyday contexts, there is a growing need to democratize its use, enabling end users to create more personalized experiences. To support this shift and promote broader participation, there has been a continuous pursuit of effective and intuitive methods to reduce the effort required for AR creation [62, 67, 74, 90]. Traditionally, creating AR applications requires substantial efforts in two key aspects: design and implementation. Users need to: 1) design the overall structure of the AR applications by determining the necessary techniques and how they should be integrated to achieve the desired functionality; 2) Implement the design into functional AR applications through programming. This process can be challenging for users with limited experience in AR design or programming, restricting their participation.

To address these challenges, previous works have introduced various methods to simplify AR creation. These methods can be broadly categorized into demonstration-based and visual programming methods. Demonstration-based methods [59, 89, 90] provide users with pre-designed application structures supported by trigger-action pairs and enabled implementation via physical demonstrations. However, such an authoring mechanism restricts users to limited, task-specific applications, which are insufficient to support the design of AR experiences for a wide range of scenarios in real-world contexts. In contrast, leveraging the modular nature of AR

applications, visual programming methods [21, 62] enabled block-based compositions of AR applications, supporting more flexible AR creation. Users can design AR applications by connecting blocks, then the system implements designs into functional applications. Despite simplifying implementation, visual programming methods still require considerable user effort in design (i.e., decomposing system behavior, selecting appropriate blocks, and coordinating nested structures to achieve the desired execution), especially for complex applications, which can be a barrier for inexperienced users. To this end, we identify the call for a comprehensive approach to reduce user efforts in both designing and implementing AR applications.

New opportunities to address this gap have been introduced by the emergence of tool-augmented large language model (LLM)–based autonomous agents [88] (tool-augmented agents, for short). Leveraging LLMs as central controllers and equipped with specialized tools, such as external APIs [68, 73] or knowledge bases [28, 40], such agents autonomously complete complex tasks through self-directed planning (i.e., designing the task workflow) and execution (i.e., utilizing tools to implement the plan), from solely natural language input. Tool-augmented agents have been widely employed in diverse fields, including software engineering [25, 66], industrial automation [97], product design [17], and natural science research [5]. The planning and execution stages of tool-augmented agents naturally align with the design and implementation aspects of AR creation, respectively. By autonomously performing both stages based on natural language input, these agents offer a promising approach to automating AR creation and significantly reducing user effort. Earlier attempts [86] utilized tool-augmented agents to generate interactive objects and scenes in AR (e.g., a virtual bathroom with a bathtub, toilet, etc.), based solely on textual descriptions. While these attempts highlight the potential of tool-augmented agents in AR creation, they are limited to virtual content generation and, therefore, lack the capabilities for designing and implementing comprehensive AR applications, thus falling short of supporting AR application creation.

To this end, we present agentAR, an AR authoring system that leverages a tool-augmented agent to autonomously create AR applications from natural language. To *plan* (i.e., design AR applications), agents must understand how to structure an application; to *execute* (i.e., implement the applications), they need access to the necessary tools. To inform agent-based creation, we performed a review of state-of-the-art AR research projects, based on which we derived a common structure for AR applications and constructed a tool library for AR application creation. Building on these insights, we developed a tool-augmented agent to enable AR application creation from natural language. Taking this agent as the backbone, agentAR further supports chat-based interactions and native deployment of AR applications, enabling an end-to-end and in-situ creation experience, reducing user effort to simple language input.

To evaluate the effectiveness of agentAR, we first conducted a case study. We selected six AR applications from our review and re-implemented them using agentAR. Our re-implementations successfully replicated the core functionalities of the original AR applications, demonstrating that agentAR is effective in both planning curated AR applications and executing them using the tool library. We then conducted a two-session user study with twelve participants to quantitatively assess the system's performance and

usability. Results show that agentAR significantly outperforms a baseline agent in AR application creation and enables users with limited AR experience to create various AR applications with minimal effort.

Following is a list of our contributions:

- A novel approach for creating AR applications from natural language using tool-augmented LLM autonomous agents.
- A review of state-of-the-art AR research to derive a common structure and tool library that supports agent-based AR application creation.
- agentAR, an AR authoring system that leverages a tool-augmented agent to enable end-to-end and in-situ AR application creation from natural language.
- A case study comprising six AR applications, and a user study with twelve novice users demonstrating the effectiveness of agentAR in creating various AR applications.

2 Related Works

2.1 Authoring Systems for Augmented Reality

Various prototyping tools have been proposed to facilitate the creation of augmented reality (AR) experiences. Traditional methods can be broadly classified into demonstration-based and visual programming-based methods.

Demonstration-based Methods enable users to define AR applications through hands-on demonstrations, thereby reducing the burden of design and implementation. Typically, the authoring system records user movement and narration and then maps them onto a predefined application structure [33, 74]. For instance, GhostAR [8] allows users to physically demonstrate movements in AR to program robot behaviors. Meanwhile, CAPtuAR [90] uses an in-situ program-by-demonstration approach to rapidly author context-aware applications based on previous user activities. Similarly, Teachable Reality [59] leverages vision-based interactive machine teaching techniques to transform real-world interactions into AR applications. TutorialLens [42] and InstruMentAR [49] provide an authoring environment for creating interactive AR tutorials through step-by-step narration and demonstration. At a higher level, ScalAR [67] integrates a VR-based demonstration workflow, automatically deploying applications across various AR scenarios after a single demonstration. While demonstration-based methods substantially simplify AR creation, the simplistic authoring mechanisms often limit them to specific scenarios. Consequently, these methods may not effectively support the creation of versatile and complex AR applications.

Visual Programming Methods offer greater flexibility by providing users with elementary functional blocks to build AR applications. These methods package and present these blocks through intuitive visual interfaces, relieving users from having to implement everything by manually writing code, while supporting the creation of more diverse AR experiences. Earlier attempt DART [53] provides an AR design toolkit that decomposed 3D content animation into DART behaviors, allowing designers to compose a variety of AR animations. More recent toolkit-based visual-programming platforms, such as ARGUS [9], SIGMA [3], and TOM [34], follow a similar template-driven approach to scaffold a wide range of AR applications, lowering the barrier for intermediate developers.

BlocklyAR [62], BlocklyXR [38], and LearnIoTVR [102] offer visual programming environments like Scratch [70] but are designed for creating AR and extended reality applications. Pronto [44] facilitates dynamic AR interaction design by integrating tablet-based video prototyping and 3D manipulation, while SpatialProto [60] further expands capabilities by enabling users to record and animate objects from real-world environments, subsequently placing and transforming them within AR scenes. In exchange for greater user control, visual programming methods require substantial effort in the design process, particularly as application complexity grows. This overhead can hinder inexperienced creators from creating various AR experiences effectively.

2.2 LLM-based Autonomous Agents

LLM-based autonomous agents have been widely adopted across various domains to tackle complex tasks through self-directed planning and action [88]. In computer science and software engineering, these agents have shown strong potential for automating processes such as coding, testing, debugging, and documentation generation. For example, MetaGPT [25] and ChatDev [66] utilize multi-agent frameworks, where agents with different roles collaborate through natural language conversations to complete end-to-end software development tasks. ChatEDA [93] applies an agent-based approach to electronic design automation (EDA), streamlining the workflow by integrating task planning, script generation, and execution.

Beyond software development, LLM-based agents have also been applied in industrial automation, robotics, and embodied AI. Xia et al. [97] integrate LLMs with digital twin systems, enabling agents to adapt to task-specific requirements for hierarchical production tasks. TaPA [96] introduces agents that align visual perception with task planning, improving execution quality in visually grounded tasks. Dasgupta et al. [13] propose a Planner-Actor-Reporter paradigm for embodied reasoning and sequential task planning. TidyBot [94] presents an embodied agent capable of learning user preferences for household object manipulation via textual examples, enabling personalized cleaning behaviors.

We explore the use of a tool-augmented LLM agent to empower AR application creation. By leveraging the agent's self-directed planning and execution capabilities, augmented by a specialized AR-focused tool library, we aim to automate the AR creation process and enable inexperienced users to intuitively create versatile AR applications using solely natural language input.

2.3 LLM in MR

LLMs have seen extensive application in MR, significantly enhancing spatial understanding, object manipulation capabilities, and language-based development.

Spatial Understanding and Scene Analysis. Recent research has explored how LLMs can enhance comprehension of 3D scenes and spatial contexts within MR environments. Models such as 3D-LLM [26], PointLLM [100], and ShapeLLM [65] enable LLMs to effectively interpret 3D objects and scenes, achieving tasks such as 3D captioning, question answering, and navigation, thereby enhancing accessibility and user comprehension within MR contexts.

Table 1: Examples of reviewed AR research

Short Title	Core Functionality	Year	Conference
RealitySketch [82]	AR interface for sketching interactive graphics	2020	UIST
Radi-Eye [76]	Use gaze & head-crossing to select widgets	2021	CHI
Free-Throw [48]	Visualizations in basketball free-throw training	2021	CHI
LightPaintAR [91]	AR interface with virtual light traces to create light paintings	2021	CHI EA
Opportunistic Interfaces [16]	Enable virtual interfaces on everyday objects	2022	CHI EA
Augmented Reading [84]	AR-based implementations of Bionic Reading and Spritz	2022	SVR
AR Museum [85]	Mobile AR virtual guidance system for museum	2022	SVR
ARCAM [45]	AR mobile video shooting camera movement guidance system	2022	MHCI
Augmented Math [12]	AR explanations by augmenting static equations	2023	UIST
BeeAR [55]	AR navigation system with always-visible digital landmarks	2023	MHCI
DeAR [32]	Hybrid desktop-AR for synchronized data visualization and interaction	2023	SVR
Breaking the Plane [18]	Visualize mathematical functions in 3D with handwritten input	2024	CHI EA
GazePointAR [43]	Use gaze and point gesture to enhance speech queries	2024	CHI
SonifyAR [80]	Generates context-aware sound effects with LLM in AR	2024	UIST
XR-Objects [14]	Enable digital interaction with real-world objects	2024	UIST
avaTTAR [52]	Provide visual cues for table tennis stroke training	2024	UIST

Human-Agent Interaction. LLMs also play a crucial role in enriching human-agent interactions within MR. Wan et al. [87] introduce an LLM-based AI agent specifically designed and evaluated for enhancing user-agent interaction in Virtual Reality. ClassMeta [50] employs GPT-4 [63]-powered embodied avatars capable of interacting naturally with instructors and students through both verbal and gestural communications, significantly boosting engagement in virtual classroom settings.

Embedded MR Assistants. LLMs have also been integrated into various MR environments as embedded assistants, providing context-aware support tailored to specific scenarios. SocialMind [101] presents an LLM-based proactive AR system that provides in-situ social assistance. ARAS [36] develops an operational AR assistant that leverages LLMs to facilitate intuitive and natural communications between surgeons and AR guidance systems during surgical procedures. OmniActions [35] establishes a pipeline based on LLMs to process multi-modal sensory inputs and predict actionable responses, while Augmented Object [14] combines LLMs with object segmentation and classification, enabling interactive digital augmentation of physical objects.

Object Manipulation and Creation. Further applications of LLMs in MR have been focused on object manipulation, creation, and behavior design. sMoRe [99] is an MR application that utilizes LLMs to assist users in creating, positioning, and managing virtual objects within physical environments. DreamCodeVR [20] proposes a system that translates natural language commands into executable code, assisting users in defining object behaviors within VR applications dynamically. LLMR [86] presents an LLM-driven architecture integrating planning, scene analysis, and interactive object creation, enabling users to build AR scenes using language instructions. Moreover, Social Conjuring [41] introduces an AI-enhanced framework for dynamic 3D scene co-creation, enabling real-time collaborative construction and modification of virtual worlds by multiple users.

While language-based methods, especially agent-based methods like LLMR [86], show considerable promise by integrating design

and implementation seamlessly and intuitively, existing solutions are still limited in integrating virtual content with physical environments. We present agentAR, an AR authoring system that embeds a tool-augmented LLM agent to support in-situ and end-to-end AR-native authoring and deployment. This agent can be seen as an embedded assistant that supports AR application creation. Furthermore, during creation, agentAR leverages the spatial and scene understanding capabilities of LLMs and provides interactive communication to enhance user experience and facilitate intuitive AR application development.

3 agentAR System Design

To enable agent-based creation, we first conducted a review of state-of-the-art AR application research, from which we derived a common structure for AR applications and identified foundational building blocks that serve as tools for the LLM agent. Building on these insights, we developed a tool-augmented agent by crafting its planning and execution mechanisms to automate the design and implementation of AR applications. Taking this agent as the backbone, we presented agentAR, an AR authoring system that enables end-to-end and in-situ AR application creation from simple language input.

In the following sections, we elaborate on our review of AR research, then introduce the workflow of agentAR, followed by a detailed explanation of the planning and execution mechanisms of the tool-augmented agent, as well as agentAR user interface.

3.1 AR Research Review

We conducted a review of state-of-the-art AR application research to inform agent-based creation, focusing on how the agent can design and implement an AR application. Specifically, we collected and analyzed prior AR literature by examining their system structures and identifying foundational building blocks. The structures of AR applications inform the agent on how to design them, while the foundational building blocks serve as tools that the agent can use during implementation. Given the large volume of AR research

and the diversity of its application domains, our review focused on three representative domains: *General-purpose*, *Sports*, and *Education*. This selection was made to achieve a balance between general and specialized use cases, enabling the agent to handle general-purpose AR creation while also equipping it with some domain-specific capabilities. Based on our review, we derived a common structure for AR applications and constructed a tool library tailored to AR creation.

3.1.1 Review Process. To gather representative AR research within our selected domains for analysis, we reviewed AR-related papers published in major HCI venues, including CHI, UIST, MobileHCI, SIGGRAPH, and their associated workshop sessions. In our brief review, we focus on research that was published later than 2020. Rather than compiling an exhaustive literature corpus, our goal was to survey a representative set of AR applications. To achieve this, we first conducted a keyword search for "Augmented Reality", and then examined the titles and abstracts of all publications to identify research related to the selected domains. In this process, we excluded review articles, studies, and research focused on hardware or human collaboration, concentrating on AR software for individual use. Following this filtering, we collected a total of 42 papers.

We further reviewed the system designs and relevant sections of all collected publications to analyze their overall workflows or system structures, core functionalities, and the individual components involved. This analysis allowed us to understand the common application structure and foundational building blocks used across the collected cases. We present some examples of the collected research in Table 1 and the full list in Appendix. A.8.

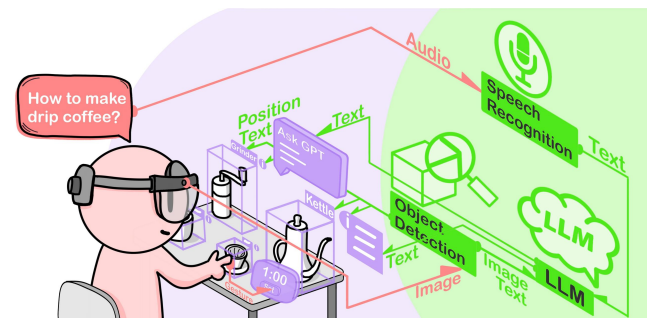


Figure 2: Structure of the coffee making application shown in Figure. 1(e). Frontend Layer is shown in purple, and Backend Layer is shown in green. Arrows represent data flow. Red arrows indicate inputs to the AR application.

3.1.2 Structuring AR Applications. Based on our review, we derived a common structure for AR applications to support agent-based creation. Drawing from the nature of AR applications, the structure of each application can be divided into two layers: a *Backend Layer* and a *Frontend Layer*. *Backend Layer* takes input from sensory devices (e.g., cameras, microphones) and is responsible for user and environmental perception, reasoning, and data processing. In contrast, *Frontend Layer* is responsible for rendering visual content, presenting user interfaces, and supporting other interactive elements that

users directly perceive and engage with. These two layers interact bidirectionally, i.e., *Frontend Layer* can visualize the data processed by *Backend Layer*, while user interactions captured through the *Frontend Layer* can influence the behavior and processing flow of *Backend Layer*.

Each layer is composed of several building blocks. These blocks serve as tools for the agent in AR application creation and are thus referred to as tools in our structure. Tools within the same layer can operate independently or be connected to support more complex functionalities. Additionally, tools from different layers can also interconnect to exchange information. To reflect their distinct roles, we categorize these tools into *Backend Tools* and *Frontend Tools*, corresponding to their respective layers.

This structure enables an agent to design various AR applications in a unified and structured format. For example, the coffee-making application shown in Figure. 1(e) can be represented as Figure. 2.

Table 2: Tools collected from our review of AR application research.

Tool	Example
Backend Tools	
Object Detection & Tracking	[14, 27, 43]
Human Pose Estimation	[52, 95]
Hand Gesture Detection	[16, 27]
Hand Tracking	[71]
Plane Detection	[80]
Gaze Tracking	[24, 43, 76]
Speech Recognition	[14, 16, 43]
Speech Synthesis	[7, 43]
OCR	[12, 23, 35]
LLM	[14, 24, 35]
Math Simulator	[12, 18]
Physics Simulator	[23, 82]
Frontend Tools	
Data Visualizer	[14, 18, 52, 82]
Animator	[23, 27, 80, 82]
User Interface	[7, 14, 76, 80]
2D & 3D Assets	[27, 80, 99]

3.1.3 Constructing Tool Library. We aimed to construct a generalizable yet flexible tool library rather than compile an exhaustive inventory. To this end, we identified and collected commonly used functional building blocks based on their usage across the reviewed literature, while excluding highly customized features that rarely appear elsewhere, such as the custom graph detection algorithm in [12]. These collected building blocks are built as tools for our tool library. In this tool library, we focus on capturing the essential functionality of tools rather than distinguishing between specific implementations. For instance, Augmented Math [12] and Breaking the Plane [18] utilize 2D and 3D data visualizers, respectively, for mathematical equations, while avaTTAR [52] employs a 3D visualizer for human body poses. Despite differences in data dimension and use contexts, all these tools fundamentally serve to visualize

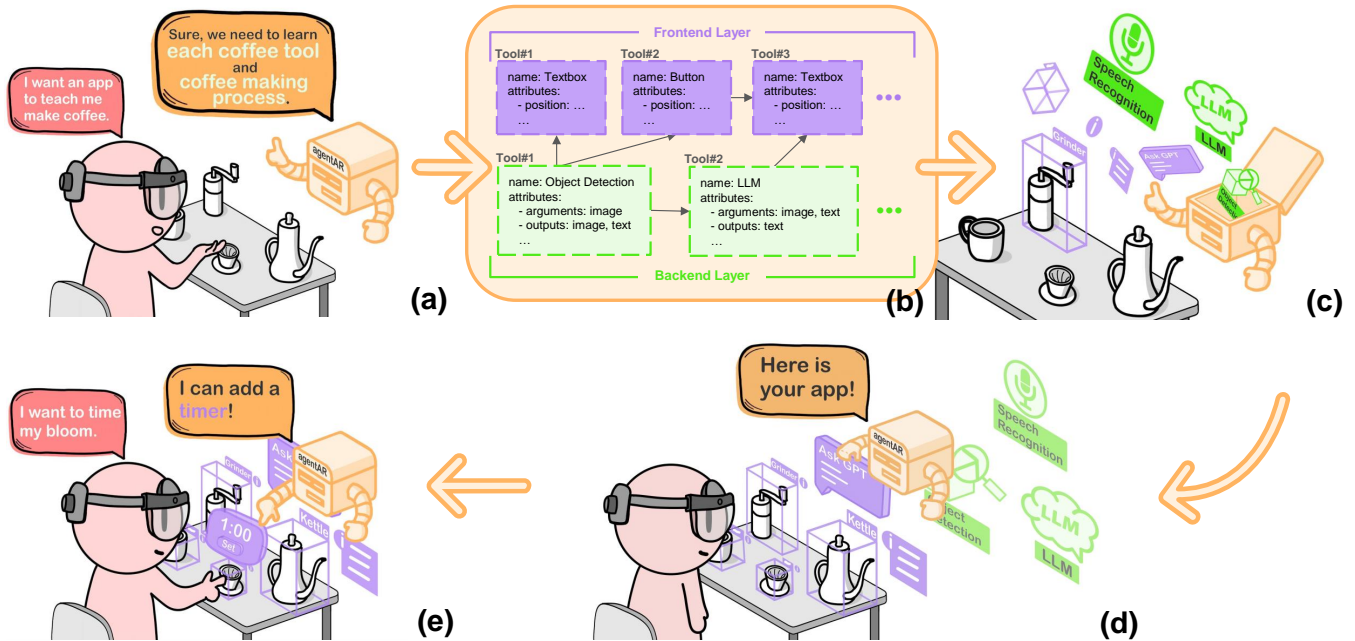


Figure 3: agentAR workflow. (a) Users describe the desired AR applications via natural language dialogue; (b) Planning stage: agentAR interprets user requests and designs the application structure; (c)-(d) Execution stage: agentAR further leverages a tool library to implement the design, generate and deploy a functional AR application; (e) The creation process is iterative—users can continue chatting with agentAR to edit or refine the application.

discrete data points. Therefore, we collectively categorize these tools under the *Data Visualizer* category. We further categorized collected tools into *Backend Tools* and *Frontend Tools*, as discussed in Sec. 3.1.2. A summary of the collected tools, accompanied by representative examples from the reviewed literature, is provided in Table 2.

To better inform planning and enhance execution, we structured each tool into a unified format by defining a set of attributes for each tool that clearly specified its usage. For *Frontend Tools*, we define the following three attributes:

- **Pose:** The spatial position and orientation of the tool in the environment.
- **Activation:** The activation of the tool.
- **Interface:** The interaction interface of the tool.

These attributes determine how the *Frontend Tools* are presented and interacted with in the AR environment.

For *Backend Tools*, we define a separate set of attributes:

- **Tool Description:** An explanation of the tool’s core functionality.
- **Arguments:** The input parameters required by the tool.
- **Outputs:** The data or results returned after tool utilization.

Further details and an example tool definition can be found in Appendix. A.2. Notably, our representation and tool library are derived to support the planning and execution steps of the agent during AR application creation. While they do not aim to cover all possible AR scenarios, we believe they capture the most common

cases within the selected domains and provide sufficient flexibility for the agent to construct a wide range of AR applications.

3.2 agentAR workflow

The workflow of agentAR is shown in Fig. 3. Fundamentally, agentAR enables users to create AR applications through a natural language dialogue. Users simply describe the applications they want, and agentAR interprets these requests to plan the application and then executes the plan to generate a functional AR experience. This process is in-situ and end-to-end. The generated application is compiled and deployed at runtime within agentAR, allowing users to immediately experience and use it in AR. Moreover, this workflow is iterative—users can continue chatting with the system to edit or refine the application. Each time, agentAR revisits the planning and execution steps with awareness of the current context, enabling dynamic adaptation to different needs or environments.

3.3 Planning

During the planning stage, agentAR generates a semi-structured plan for the AR application, based on the application structure introduced in Sec. 3.1.2.

To support AR application creation from simple natural language input and reduce user effort, particularly for inexperienced users, agentAR supports planning from vague, high-level requests such as “I want an AR application to teach me how to make coffee”, without requiring users to specify the application structure or implementation details. To handle such inputs, agentAR first analyzes the user request. If it lacks important information to generate a concrete

plan, the system actively engages the user through the dialogue, offering suggestions or requesting clarifications to gather the missing details needed to produce an actionable application plan, as shown in Figure. 1(b).

To ensure effective planning, agentAR incorporates specification-based instruction and demonstration parsing in its prompt, following the approach introduced in [73]. Additionally, we enable the agent to access multimodal contextual information to support context-aware planning. We elaborate on each of these features below.

Specification-based Instruction. Task specification provides a standardized template for planning, enabling the agent to systematically design an application through slot filling. Following Sec. 3.1.2, we define distinct specifications for *Backend Layer* and *Frontend Layer*.

For *Backend Layer*, we define:

- *Arguments*: Specifies the exact inputs required by the backend layer.
- *Returns*: Describes the outputs of the backend layer, including both data structures and their natural language interpretations.
- *Tools Involved*: Lists all tools used in constructing the backend layer of the AR application.
- *Structure*: Describes the logic, architecture, and data flow of the backend layer using pseudo code. It elaborates on how tools are utilized and interconnected within the layer.

For *Frontend Layer*, we define:

- *Tools Involved*: Lists all tools used in constructing the frontend layer of the AR application.
- *Backend Layer Arguments*: Specifies the arguments from the backend layer that the frontend program should utilize.
- *Structure*: Describes the logic, architecture, and data flow of the frontend layer using pseudo code. This includes specifying the attributes for each tool involved.

Demonstration Parsing. agentAR leverages in-context learning for more effective task parsing and planning. By injecting several demonstrations into the prompts, agentAR allows the agent to better understand the intention and criteria for planning. Each demonstration is a group of input and output on planning—the user’s request and the expected plan to be parsed out. Furthermore, these demonstrations, consisting of dependencies between tools parsed from the user’s request, effectively aid the agent in understanding the logical relationships between tools and determining the application structure and data flow. We provide an example demonstration in Table 3. Following [73], we adopt six demonstrations in our implementation. All the demonstrations are drawn from the reviewed AR applications presented in Section 5.

Context-aware Planning. Context awareness plays a critical role in effective planning. agentAR incorporates two kinds of contextual information to enhance the agent’s planning capability. The first is environmental context. When a user initiates a request, the agent captures an image of the surrounding environment. This visual input complements the user’s language description by providing environmental context, allowing the agent to interpret user intent more accurately and reduce ambiguity in natural language

Table 3: An example demonstration provided to the agent during planning.

Input	Detect the object being looked at, retrieve information, and initiate a chat box.
Backend Layer	<p>Arguments: None</p> <p>Returns: dict with field 'ListOfObjects': list of objects with 'Label' and 'Response'</p> <p>Tools Involved: Object Detection, Gaze Tracking, LLM, Speech Synthesis</p> <p>Structure:</p> <p>Image ← Camera() Gaze ← GazeTracking() Label, Location ← ObjectDetection(Gaze, Image) Response ← LLM(Image, Location) Speech ← SpeechSynthesis(Response)</p>
Frontend Layer	<p>Tools Involved: User Interface (Textbox)</p> <p>Backend Layer Arguments: Response</p> <p>Structure:</p> <p>Define Detect: Response ← Backend() Create Textbox Textbox.Content ← Response Create Button Button.Select ← Detect()</p>

instructions. The second is planning history. When editing or updating existing applications, the agent is provided with access to previous plans. This allows it to make informed updates, ensuring that the requested changes are addressed without unintentionally modifying unrelated components of the application.

We provide the meta-prompt for planning in Appendix. A.1.

3.4 Execution

Once a plan is generated, agentAR proceeds to utilize the specified tools to compose a functional AR application. Specifically, based on the semi-structured plans for *Backend Layer* and *Frontend Layer* generated during the planning stage, agentAR produces the corresponding programs. This process involves invoking the specified tools, setting their attributes, and establishing data flows and dependencies to ensure coherent integration, according to the plans. In this step, we also adopt a demonstration-based parsing approach, where the agent is given a list of all available tools along with several example outputs (i.e., the final programs) to guide the composition of AR applications using the necessary tools. In addition to calling tools from the library, the agent also generates code for simple logic and basic integration operations, according to the plan. We provide an example *Backend* and *Frontend* program generated by agentAR in Appendix. A.7. By composing both *Backend* and *Frontend* programs, agentAR produces a complete AR application.

agentAR then compiles the generated application at runtime. To ensure successful compilation, agentAR adopts a feedback loop across the planning and execution stages. If compilation fails, the agent is provided with debug messages generated by the compiler. These messages help identify errors in the generated application, enabling the agent to refine its plan until a functional AR application

is deployed. This runtime deployment along with the feedback loop mechanism makes agentAR an AR native deployment system, providing an end-to-end creation experience—from natural language requests to complete, functional AR applications.

Moreover, although agentAR is designed for rapid in-situ prototyping, it also allows users to export the source code of generated applications at runtime, providing a foundation for further customization and development. Thanks to the unified tool format, shared attributes, and consistent application structure, the generated programs can be extended with minimal effort—developers need only adjust attribute values, refine connections between tools, or introduce new tools following the same schema to evolve quick prototypes into more sophisticated applications.

3.5 User Interface

agentAR is an AR native deployment system. It allows users to create AR applications, as well as experience the created applications in AR. Users can easily switch between the creation mode and generated applications using the attach-to-view buttons **agentAR** and **Recenter**. In creation mode, agentAR offers a simplistic user interface to help users create AR applications. This interface consists of a dialogue panel with a set of buttons: **Talk**, **Submit**, **Clear**, **Export**, as shown in Fig. 4(a).

- **Talk** lets users speak to the agent.
- **Submit** sends their spoken request to the agent.
- **Clear** removes the current transcription in case of input errors.
- **Export** allows users to save and export the generated applications for further development.

In addition, to enhance explainability, agentAR generates a **User Manual** upon execution to guide users on how to use the AR application. We present an example *User Manual* in Figure. 4(b).

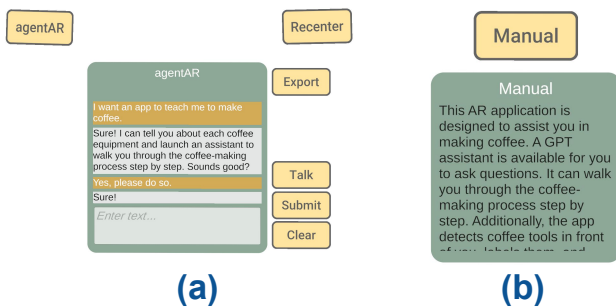


Figure 4: (a) User interfaces of agentAR; (b) User Manual for the coffee-making application in Figure. 1(e).

4 Hardware and Software Implementation

4.1 Hardware and Software Setup

We set up agentAR using a customized Oculus Quest Pro [57] head-mounted display (HMD) paired with an Intel RealSense depth camera D435i [30] to provide additional input, as shown in Fig. 1(a). The depth camera and the Oculus Quest’s built-in camera were calibrated in advance to ensure accurate spatial alignment. The

depth camera and Oculus headset were connected to a local PC (Intel core i7-9700K CPU, 3.60 GHz, 32 GB RAM, Nvidia GeForce GTX 2080 GPU) via Type-C cables. The depth camera captured both RGB and depth frames at a resolution of (640, 360), the color and depth frames are aligned with the same frame rate of (30 Hz). agentAR is developed on Unity 3D (2022.3.20f1) and implemented on the PC. We used GPT-4o [63] model as the controller of our agent. We used Roslyn C# compiler [31] to compile generated AR applications at runtime, following the approach used in [72].

4.2 Tool Library Implementation

We implemented all tools in our tool library. Our implementations aimed to balance performance and efficiency to meet the wide range of use scenarios of AR applications. Therefore, rather than focusing exclusively on computational performance, we prioritized lightweight, compatible, and scalable implementations that better support AR applications and seamless integration of tools. While this approach may limit system capabilities to some extent, it preserves the core functionalities necessary for building a wide range of AR applications. Moreover, for tools with diverse design options, such as user interfaces or math simulators, whose specific behaviors vary across different applications, we implemented flexible yet simplified versions that provide the core functionalities. In addition to the tool library, we also implemented a set of supplementary tools for basic object behaviors, I/O operations, and data processing, such as *Get Microphone Input* and *Crop Image*. We believe our simplified yet modular implementations provide sufficient flexibility for agentAR to support the creation of diverse AR applications. Implementation details for all the tools are provided in Appendix. A.3.

5 Case Study

We conducted a case study to assess the potential of agentAR in supporting AR application creation. Specifically, to examine the system’s capability in planning curated AR applications from natural language and utilizing tools to execute them, we re-implemented six AR applications across the three selected domains from the reviewed literature, aiming to reproduce their core functionalities using only natural language input.

5.1 Selected AR Applications and Our Re-implementations

We selected six AR applications from our review and re-implemented them using agentAR. An overview of these applications and our re-implementations is shown in Fig. 5, with detailed descriptions provided below.

XR-Objects [14] (Figure. 5(a)). XR-Objects enables analog objects to initiate digital actions, such as querying for details or executing tasks. It achieves seamless integration of physical objects as interactive digital entities by utilizing object detection with multi-modal LLMs. Our re-implementation uses the object detection tool to identify and localize objects. For each detected object, the application places buttons around it for users to initiate various actions, such as checking info, opening a notepad, a timer, or activating an LLM assistant for discussions.

avaTTAR [52] (Figure. 5(b)). avaTTAR is an AR application for table tennis stroke training. It captures and reconstructs the

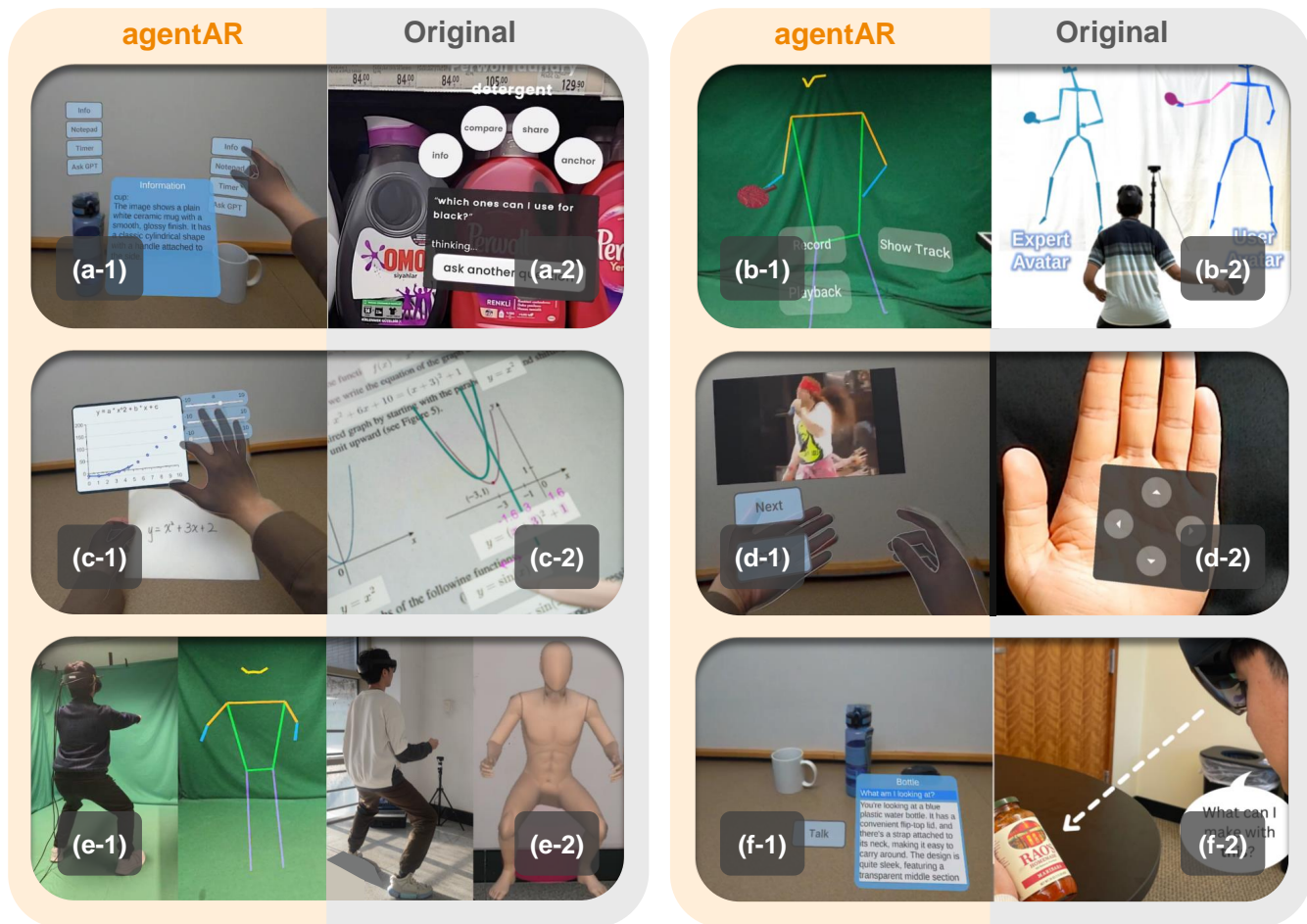


Figure 5: Selected AR applications and our re-implementations in the case study. (a-2)-(f-2) Selected AR applications; (a-1)-(f-1) Our re-implementations. (a) XR-Object [14]; (b) avaTTAR [52]; (c) Augmented Math [12]; (d) AhUI [16]; (e) AR-Enhanced Workouts [95]; (f) GazePointAR [43].

3D human pose and paddle orientation during practice to provide both first-person and third-person view visual cues, enabling users to visualize target strokes and correct their attempts effectively. Our re-implementation employs the human pose estimation tool to track the instructor's body movements and uses the controller to simulate a table tennis paddle. A virtual paddle model is retrieved and attached to the controller. The system also incorporates a record/playback tool to capture and replay motion sequences. During playback, users can 'walk into' the body pose visualizer for a first-person perspective or observe it from the side for a third-person view.

Augmented Math [12] (Figure. 5(c)). Augmented Math is a machine learning-based approach to authoring explorable math explanations in AR. It augments static math textbooks by identifying mathematical formulas with OCR, manipulating them, and creating interactive animations. Our re-implementation employs the OCR tool to identify math equations in front of users and visualize the

equation with the math simulator and data visualizer. Additionally, users can adjust the coefficients with sliders, exploring how variations in coefficient values affect the math equation.

Ad hoc UI (AhUI) [16] (Figure. 5(d)). AhUI is a prototyping toolkit that leverages real-time tracking, voice activation, mid-air gestures, etc., to empower users to transform everyday objects into opportunistic interfaces on the fly. In our re-implementation, we use hand tracking to attach a set of buttons to the user's hand, allowing them to move with hand motion. These buttons serve as controls for a video player, enabling intuitive and mobile interaction.

AR-Enhanced Workouts [95] (Figure. 5(e)). AR-Enhanced Workouts employs an AR-based application that provides visual cues for workouts. AR-Enhanced Workouts captured users' exercise performance with human body pose estimation technology and provided feedback via AR visual cues. The re-implementation of this app incorporates the human pose estimation tool for tracking user movement. The visualized movement is displayed in front

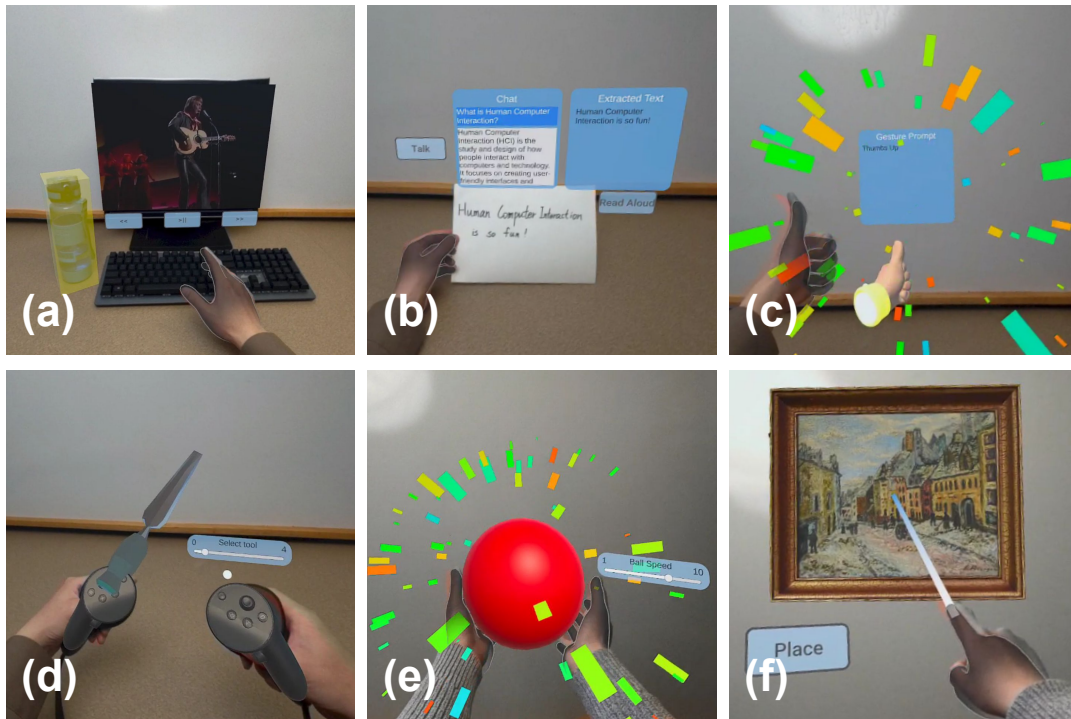


Figure 6: Additional AR applications created by agentAR. (a) Immersive Video Player; (b) Interactive Paper; (c) Gesture Learning Assistant; (d) Hand Tool Tutorial; (e) Ball Catching Game; (f) ARTPlace.

of the user, providing visual cues for users to understand their movements.

GazePointAR [43] (Figure. 5(f)). GazePointAR is an LLM-based context-aware visual assistant for wearable AR that leverages eye gaze and conversation history to disambiguate speech queries. Our re-implementation leverages the gaze tracking tool, the object detection tool, and the LLM tool. When users trigger the *Talk* button, it acquires users' gaze direction, detects the object being observed, leverages an LLM to query to retrieve details, and allows for further inquiries.

We provide the prompts used to create the above applications in Appendix A.4. When generating each application with agentAR, we first removed that application from the agent's demonstration set to prevent data leakage and ensure an unbiased evaluation of the system's capabilities.

5.2 Results

In this case study, agentAR successfully re-implemented six AR applications from prior research, reproducing their core functionalities from natural language descriptions. This demonstrated agentAR's potential to create a diverse set of AR applications across use scenarios and in different forms, enabling comprehensive AR application creation via solely natural language.

6 Additional Examples

To further showcase the capabilities of agentAR in AR application creation, we include additional examples generated by the system.

Each application was proposed by an LLM instructed to describe AR application ideas in natural language; the meta-prompt used to guide the LLM is provided in Appendix A.5. The resulting applications are shown in Figure 6, with detailed descriptions below.

Immersive Video Player (Figure. 6(a)). Immersive Video Player allows users to place a video player at a location where the users touch the physical screen in the surroundings. Users can play or pause the video by tapping the table. Additionally, the app detects bottles in the environment and turns them into controllers. Users can switch to the next video by poking the detected bottle.

Interactive Paper (Figure. 6(b)). Interactive Paper recognizes text in front of the user, reads it aloud, and allows the user to make further inquiries about the content through an LLM assistant.

Gesture Learning Assistant (Figure. 6(c)). Gesture Learning Assistant randomly selects a predefined gesture, displays its name, and shows a 3D model for users to mimic. If the user performs the gesture correctly, the app provides visual feedback with confetti and a successful sound, and then proceeds to the next gesture. If the gesture is incorrect, it plays a metallic sound and stays on the current one.

Hand Tool Tutorial (Figure. 6(d)). Hand Tool Tutorial displays and attaches various virtual hand tools to the user's left controller. A slider allows users to switch between different tools, including a file, groove pliers, a hand saw, needle-nose pliers, and a screwdriver.

Ball Catching Game (Figure. 6(e)). Ball Catching Game spawns a virtual ball in front of the user and launches it every 5 seconds.

If the user successfully catches the most recent ball, it is removed from the scene.

ARtPlace (Figure 6(f)). ARtPlace detects nearby surfaces and lets users place virtual paintings by pointing at their desired location. When the “Place” button is tapped, the selected painting is anchored to the detected surface.

7 User Study

We conducted a two-session user study to evaluate the effectiveness and usability of our system. We recruited twelve users (8 self-identified as male and 4 self-identified as female), aged between 19 and 30 years (AVG=25.25, SD=3.03). Of the twelve participants, eight were familiar with AR applications on smartphones, tablets, or head-mounted devices, while the remaining four had had prior exposure to AR or VR technologies. None of the users has developed any AR or VR applications before. All the users were familiar with LLMs. The entire study lasted approximately one hour per participant, and each participant received compensation in the form of a \$20 e-gift card. Before diving into the study, participants were explained the study and asked to sign the consent form. Then, participants completed a warm-up session using a demo AR application, where they were guided to experience each tool provided by our system. This session was designed to familiarize participants with the AR-HMD and its interactions. There are two sessions in our study. In the first session, participants were asked to replicate an assigned AR application, while in the second session, they were instructed to build their own applications using the system. After each session, users completed a 5-point Likert-type questionnaire (Strongly Disagree; Slightly Disagree; Neutral; Slightly Agree; Strongly Agree) about the creation experience. At the end of the user study, each participant was interviewed and completed the standard System Usability Scale (SUS) questionnaire.

7.1 Procedure

7.1.1 Session 1: Quantitative Evaluation. In this session, we aimed to quantitatively evaluate the performance of agentAR in enabling AR application creation via natural language. In agentAR, planning plays a pivotal role in the whole workflow, since it determines the overall structure of the AR application. Therefore, we deem that the quality of planning can be utilized to evaluate the capability of agentAR and focus on evaluating the planning stage in this session. We selected six AR applications from our case study, *XR-Objects (re-implementation)*, *avaTTAR (re-implementation)*, *Augmented Math (re-implementation)*, *Immersive Video Player*, *Gesture Learning Assistant*, and *Ball Catching Game*. This selection of applications is to ensure full coverage of all the selected domains and all tools in our tool library to achieve a comprehensive evaluation of system performance. Each participant was assigned one application and tasked with recreating it using only natural language. Two participants were assigned to each application. To help participants understand the task goal, participants were instructed to explore the application directly in AR. We did not provide the *User Manual* or any textual descriptions of the application, as they may serve as potential inputs to the system and create bias in the task. Participants were asked to create the application without making further edits.

We adopt the following metrics to quantitatively evaluate system performance:

- **GPT-4 Score.** Following [73], we use GPT-4 as a judge to assess the overall correctness of the agent’s planning results. For each user request, GPT-4 is prompted to evaluate whether the agent has properly planned an AR application that satisfies the user intent. Each response is marked as correct (1) or incorrect (0). We provide the meta-prompt for GPT-4 in Appendix. A.6. The *GPT-4 Score* is computed as the percentage of correct responses:

$$\text{GPT-4 Score} = \frac{1}{N} \sum_{i=1}^N \text{Correct}_i$$

- **Precision.** Precision measures how many tools utilized by the agent are actually in the ground-truth plan:

$$\text{Precision} = \frac{|\text{Selected} \cap \text{GroundTruth}|}{|\text{Selected}|}$$

- **Recall.** Recall measures how many tools required by the ground-truth plan are utilized by the agent:

$$\text{Recall} = \frac{|\text{Selected} \cap \text{GroundTruth}|}{|\text{GroundTruth}|}$$

- **F1 Score.** F1 Score is the harmonic mean of precision and recall, reflecting the balance between them:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

7.1.2 Session 2: Overall System Usability. In this session, we aimed to evaluate the overall system usability of agentAR. Participants were encouraged to freely utilize the system to explore their own ideas on new AR applications. To provide a more engaging creation experience and stimulate participants’ creativity, this session was conducted in an office with a variety of physical objects (i.e., monitor, bottle, keyboard, etc.) to interact with. During this session, participants were allowed to edit the generated application up to five times if the initial output did not match their intent.

7.2 Results

7.2.1 Session 1 Results. To evaluate the agent design in agentAR, we used a raw GPT-4o model as a baseline during post-processing and compared its performance with that of agentAR. For the baseline agent, we excluded key features of the agentAR agent, and only instructed it to plan AR applications based on our application structure and tool library. Results are shown in Table. 4.

Table 4: User Study Session 1 results. * denotes p<0.05.

Agent	GPT-4 Score ↑	Precision* ↑	Recall ↑	F1* ↑
Baseline	91.67	77.42	61.54	68.57
agentAR	100.00	89.71	78.21	83.56

We performed the Shapiro–Wilk test on all paired data and found that most samples exhibited evidence of non-normality. Consequently, we employed the Wilcoxon Signed-Rank Test, a widely used non-parametric test for paired samples, to assess whether there were statistically significant differences between the two agents

across continuous evaluation metrics, including F1, Precision, and Recall. For binary GPT-4 scores, we used McNemar’s test, which is designed for evaluating differences between two classifiers on paired binary outcomes.

With respect to GPT-4 scores, although agentAR received perfect ratings across all test cases, McNemar’s test did not reveal a statistically significant difference between the two systems ($p=1.0$). This is likely due to the ceiling effect inherent in binary scoring, where both systems were generally able to generate plans deemed acceptable by the large language model.

On the tool utilization side, agentAR achieved significantly better performance than the baseline in both F1 ($W=3.0, p=0.0076$) and Precision ($W=2.0, p=0.015$). The improvement in Recall was not statistically significant ($W=16.0, p=0.130$), though the mean value for agentAR was still higher. All three metrics collectively reflect the system’s capability in accurately and completely identify necessary tools during the planning process.

The statistically significant gains in both F1 and Precision indicate that agentAR not only reduces incorrect tool usage but also better captures the relevant tools required to execute the intended AR application. Although Recall did not show significance, its upward trend suggests the potential for improved completeness. Overall, the improvement in F1—a harmonic mean of Precision and Recall—demonstrates that agentAR achieved more reliable and balanced tool utilization, underscoring its effectiveness in translating natural language into executable AR plans.

7.2.2 Session 2 Results. We present four applications created by participants during session 2 in Figure 7. The overall system Likert results collected are shown in Fig. 8. In general, users prefer creating AR applications using natural language (Q2: $AVG=4.3, SD=0.5$): “I think it’s amazing that you can just say what you want in natural language, and the system can make it happen. It’s so much easier than building everything step by step by yourself. (P2)” and admire our system’s ability to create (Q1: $AVG=4.3, SD=0.6$) and edit (Q4: $AVG=4, SD=0.8$) AR applications: “I said I wanted to add some control features, and it totally understood what I meant. It even gave me a few different version options—I thought that was pretty impressive. (P1)” and “I could just say I wanted to change the model, and it would replace it automatically. Super convenient. (P7)” Users also acknowledged chat-based communication with agentAR (Q3: $AVG=4.4, SD=0.5$). “The communication is easy, and it really makes me want to keep talking to it. The natural conversation is pretty engaging too. (P12)”

We also asked users about each feature of the agent. Most of the users thought agentAR can understand the context when creating or editing AR applications (Q6: $AVG=4.4, SD=0.5$). “It used what I said earlier as context and kept going from there—I think that’s a really important feature. (P8)” and “Even though I didn’t say much, it kind of guessed what I wanted to do based on the scene—it felt like it had some environmental awareness. (P7)” Besides, most users also admire the system’s effort in helping them understand how created applications work (Q5: $AVG=4.5, SD=0.7$): “After it generated the app, it explained how it was put together—like what I said at the beginning, and what modules it used. I really liked that. (P1)”

In general, users believe our system can support novice users to create AR applications (Q7: $AVG=4.2, SD=0.7$). “You don’t need to learn any AR editors or tools—just talk, and it creates things. It’s way

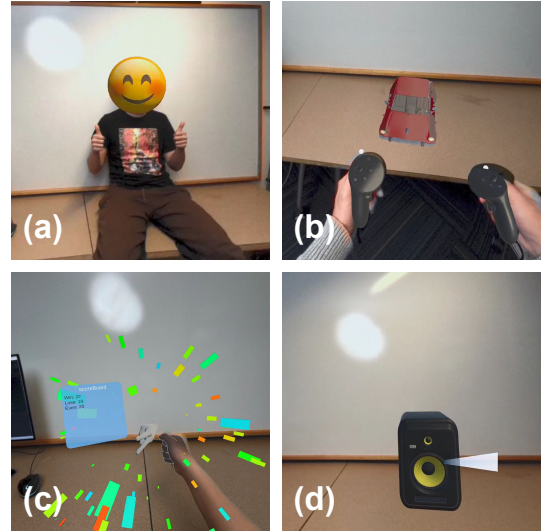


Figure 7: Example AR applications created by participants in User Study Session 2. (a) Emoji Face (P11). Displays an emoji that aligns with and tracks the head position of the person in front of the user; **(b) Toy Car (P6).** Enables users to control a virtual toy car using the left controller; **(c) Rock-Paper-Scissors (P12).** Allows users to play rock-paper-scissors with the app via hand gesture recognition, with visualized virtual gestures; **(d) Gaze-Controlled Music (P1).** Plays music when the user looks at a virtual speaker and pauses it when they look away, using eye gaze interaction.

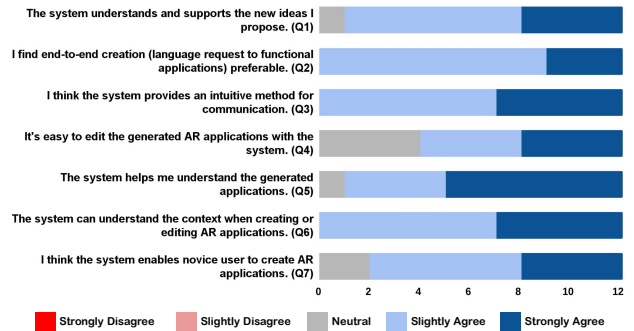


Figure 8: User Study Session 2 Likert-type questionnaire results.

“simpler than the tools I’ve used before. (P4)” For the system usability, the users reported an $M = 79.79$ and $SD = 7.27$ SUS. This score is promising and indicates the high usability of the system.

8 Limitations and Discussion

In this section, we reflect on the current boundaries of agentAR and outline directions for future work. We first summarize the failure modes observed in our studies, then discuss the broader challenges

the system faces, and propose potential solutions to advance agent-based AR authoring. We conclude by envisioning how these advances could pave the way toward more general and intelligent mixed-reality assistants.

8.1 Errors and Failures Found in Studies

While agentAR enables diverse AR application creation through natural language alone, it may produce errors or encounter failure cases due to design limitations. In our studies, we observed three primary categories of failures.

Incapacity. The system is incapable of resolving the user’s request. This typically occurs when the agent lacks the necessary tool, relevant knowledge, or when the task complexity exceeds its operational scope. Such breakdowns stem from fundamental design constraints—as noted in Sec. 3.1, agentAR’s capabilities are grounded in three literature-derived domains. Consequently, it struggles with scenarios beyond this scope (e.g., sign-language recognition or multi-user applications).

Misalignment. The generated application functions, but does not fully align with the user’s intent. These failures often involve misplaced UI elements or occasional misinterpretation of user commands. Misalignment mainly arises from the inherent error-proneness of LLMs, which remains challenging—especially in contextualized AR environments.

Ergonomic Violations. agentAR occasionally overlooks 3D layout principles that professional AR designers apply instinctively—e.g., menus appearing outside the neutral gaze cone, panels occluding physical objects, or targets rendered too small or distant for comfortable interaction. This is primarily because the current focus of agentAR is on rapid functional prototyping; detailed spatial heuristics—such as optimal gaze zones and occlusion-aware placement—are not yet encoded. Integrating these ergonomic rules represents a promising direction for future iterations of agent-based AR authoring.

8.2 Interpretability

While tool-augmented agents offer great potential for lowering the barrier to AR creation, challenges remain in their interpretability. These challenges are twofold: users may require greater transparency over the creation process, and they also need to understand the behavior of the AR applications generated by the agent [75].

In agentAR, we address these challenges by enabling the agent to engage in interactive dialogue with users—asking for additional information when needed and explaining its planning and decision-making processes. Furthermore, the system provides a User Manual to help users understand the generated AR applications.

Although these features improve the system’s interpretability and usability, textual communication alone may not be sufficient—especially for complex applications or scenarios involving spatial elements that are difficult to convey through text. To address this, there is a clear need for more intuitive and multimodal communication mechanisms that better align with the nature of AR applications. For instance, when placing user interfaces in a 3D space, the agent could utilize visual cues to suggest candidate positions or demonstrate interactions during the creation process.

Enhancing interpretability is essential for supporting non-expert users, increasing trust in agent decisions, and improving the overall usability of agent-based AR authoring systems.

8.3 Specialized and Context-Aware Planning

In agentAR, we employ a GPT-4o model as the LLM controller of the agent. While it possesses substantial internal knowledge and strong general reasoning capabilities, it is not inherently designed for domain-specific tasks such as AR application creation [74]. To equip it with the necessary capabilities, we leverage its few-shot and in-context learning abilities. Specifically, we provide demonstration examples that illustrate how to utilize tools from our tool library and adopt a specification-based parsing strategy to guide the creation process through structured slot-filling.

This approach offers several advantages: it is efficient, requires no additional model training, and is easy to deploy in practice. However, it also presents clear limitations. Since the agent learns plan and tool usage patterns only through a small number of examples, its ability to generalize to complex, highly customized, or novel AR applications is restricted. In such cases, the agent may fail to generate valid plans or appropriate tool compositions, especially when it encounters use cases that deviate significantly from the provided demonstrations. To better support AR creation, it is essential to develop LLMs tailored specifically for this task.

In addition, to support more robust and intelligent creation, the agent must also exhibit strong context awareness. In our current implementation, we achieve basic context awareness by capturing real-time environment images and maintaining access to planning history. These mechanisms allow the agent to understand and adapt to user intent more effectively. Nonetheless, this level of contextual grounding remains limited. The environment is only represented as 2D images, and the history consists of textual records, which may not be rich or structured enough for deeper reasoning.

To advance the agent’s capabilities, more comprehensive forms of contextual understanding are needed. For example, incorporating 3D spatial representations of the environment could significantly enhance the agent’s ability to reason about spatial layouts, object placements, and interaction possibilities. Integrating advanced multimodal reasoning models—such as 3D-LLM [26] or PointLLM [100]—could further improve spatial perception and affordance recognition within mixed reality (MR) environments. Additionally, establishing a tighter closed-loop creation workflow, where the agent can inspect, simulate, or even test components of the generated AR application (e.g., UI placement, tool behavior), would help identify potential issues and refine the output before deployment.

8.4 Bridging Automation and Control

While our system provides an intuitive and flexible way to create AR applications, there are important trade-offs between agent-based creation and manual development.

Agent-based methods enable dynamic, in-situ creation through natural language, lowering the barrier for inexperienced users and allowing rapid adaptation across environments. This approach aligns well with in-situ AR design principles, making it easy for users to prototype and iterate within the actual usage context.

However, this simplicity comes at the cost of control—agent-based creation currently lacks the fine-grained customization that manual development affords. In contrast, manual development allows precise control over application structure and behavior, enabling more complex and tailored AR experiences, though it demands greater effort and expertise.

To balance these strengths, agentAR supports exporting generated applications as editable code. This allows users to treat the initial output as a prototype, which can be further refined through manual development. In doing so, we aim to bridge the gap between ease of use and expressiveness, supporting both accessible creation and advanced customization.

8.5 Limitations in Available Tools

The tool library in our system was derived from a review of previous AR research, covering the most commonly used ones in three selected domains. This tool library is limited in both domain coverage and tool diversity. While it effectively supports a range of scenarios, it is not comprehensive enough to address the full breadth of AR applications. For instance, surgical applications often require precise object pose estimation or real-time physical simulation, which are not supported by our tool library. In such cases, agent-based creation via our system would be impossible. Moreover, some AR applications rely on highly customized methods that are difficult to generalize. For example, Augmented Math [12] utilizes a specialized algorithm for detecting and extracting mathematical graphs. These kinds of bespoke techniques often require expert design and adaptation for specific use cases. Supporting agent-based AR creation in such professional or technical scenarios would thus require the development of dedicated, task-specific tool libraries.

Nevertheless, recent advances in modular and multimodal tools offer a promising direction. For instance, state-of-the-art multimodal LLMs can accept diverse forms of input (text, images, shape, etc.), broadening their applicability across contexts. As tools become increasingly integrated and powerful, they can support more flexible and combinatorial use cases, enabling a wider range of AR applications without the need for extensive customization. This trend suggests a promising path toward simplifying the tool space for agent-based AR creation while enhancing its overall expressiveness and utility.

8.6 Towards General Agent Intelligence for MR

While we focus on AR application creation, the broader potential of agent-based approaches lies in evolving toward general intelligent assistants for MR. Such agents would move beyond single-purpose tasks and become always-present collaborators embedded in the MR environment. These agents could perceive the surrounding space, understand spatial layouts, track user behavior, and reason about both the virtual and physical worlds in context.

Equipped with multimodal perception—such as speech [99], vision [10], gaze [43], and motion [29]—they could interpret user intent more accurately and respond in natural, conversational ways. This opens the door for agents to assist with a wide variety of activities, from interactive tutorials and contextual reminders to on-the-fly content creation and real-time support for complex workflows.

Rather than requiring users to operate predefined apps, future MR agents could generate and adapt experiences dynamically, based on what users are doing and where they are. This vision positions the agent not just as a tool, but as a proactive, intelligent companion—capable of enhancing productivity, creativity, and everyday interaction across physical and digital boundaries.

9 Conclusion

In this paper, we introduce a novel approach to automating AR creation from natural language by leveraging tool-augmented LLM agents, and present agentAR, an agent-based AR system for in-situ, end-to-end AR application creation. To inform agent-based AR creation, we reviewed state-of-the-art AR research and derived a common structure and tool library for building AR applications. Guided by these insights, we developed a tool-augmented LLM agent—the backbone of agentAR—that creates various AR experiences from simple natural language input, thereby significantly lowering creation barriers. We evaluated agentAR through a case study in which we re-implemented six curated AR applications proposed in prior research, using only natural language. In addition, we conducted a user study to quantitatively assess the system’s performance in enabling inexperienced users to build AR applications and to evaluate its usability. Results from both studies show that agentAR not only plans and executes curated AR applications but also allows novice users to create diverse AR experiences with minimal effort—often with just a few sentences. These findings underscore the promise of agent-based approaches for AR creation, and we anticipate that, as supporting technologies advance, such methods will further democratize AR development and enable users to produce increasingly personalized AR experiences.

Acknowledgment

We wish to thank all the reviewers for their invaluable feedback. This work is partially supported by the NSF under the Future of Work at the Human-Technology Frontier (FW-HTF) 1839971 and NSF Partnerships for Innovation Technology Transfer (PFI-TT) 2329804. We also acknowledge the Feddersen Distinguished Professorship Funds. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the funding agency.

References

- [1] Vitor Barth, Adriano Vogel, Dalvan Griebler, and Marcio Sarroglia Pinho. 2024. Open Source Augmented Reality in Data Center Infrastructure Maintenance. In *Proceedings of the 24th Symposium on Virtual and Augmented Reality* (Natal, RN, Brazil) (SVR '22). Association for Computing Machinery, New York, NY, USA, 151–156. <https://doi.org/10.1145/3604479.3604527>
- [2] Rudieri Dietrich Bauer, Thiago Luiz Watambak, Salvador Sergi Agati, Marcelo da Silva Hounsell, and Andre Tavares da Silva. 2022. Development and Evaluation of a PCB’s Manual Assembly system using Augmented Reality and Total Quality. In *Proceedings of the 23rd Symposium on Virtual and Augmented Reality* (Virtual Event, Brazil) (SVR '21). Association for Computing Machinery, New York, NY, USA, 24–32. <https://doi.org/10.1145/3488162.3488231>
- [3] Dan Bohus, Sean Andrist, Nick Saw, Ann Paradiso, Ishani Chakraborty, and Mahdi Rad. 2024. Sigma: An open-source interactive system for mixed-reality task assistance research—extended abstract. In *2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, 889–890.
- [4] Angelica M. Bonilla Fominaya, Rong Kang (Ron) Chew, Matthew L. Komar, Jeremia Lo, Alexandra Slabakis, Ningjing (Anita) Sun, Yunyi (Joyce) Zhang, and David Lindlbauer. 2022. MoonBuddy: A Voice-based Augmented Reality User Interface That Supports Astronauts During Extravehicular Activities. In *Adjunct*

- Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) (UIST '22 Adjunct). Association for Computing Machinery, New York, NY, USA, Article 4, 4 pages. <https://doi.org/10.1145/3526114.3558690>
- [5] Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2023. ChemCrow: Augmenting large-language models with chemistry tools. arXiv:2304.05376 [physics.chem-ph] <https://arxiv.org/abs/2304.05376>
 - [6] Eugenie Brasier, Emmanuel Pietriga, and Caroline Appert. 2021. AR-enhanced Widgets for Smartphone-centric Interaction. In *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction* (Toulouse & Virtual, France) (*MobileHCI '21*). Association for Computing Machinery, New York, NY, USA, Article 32, 12 pages. <https://doi.org/10.1145/3447526.3472019>
 - [7] Runze Cai, Nuwan Janaka, Yang Chen, Lucia Wang, Shengdong Zhao, and Can Liu. 2024. PANDALens: Towards AI-Assisted In-Context Writing on OHMD During Travels. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI '24*). Association for Computing Machinery, New York, NY, USA, Article 1053, 24 pages. <https://doi.org/10.1145/3613904.3642320>
 - [8] Yuanzhi Cao, Tianyi Wang, Xun Qian, Pawan S Rao, Manav Wadhawan, Ke Huo, and Karthik Ramani. 2019. GhostAR: A time-space editor for embodied authoring of human-robot collaborative task with augmented reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 521–534.
 - [9] Sonia Castelo, Joao Rulff, Erin McGowan, Bea Steers, Guande Wu, Shaoyu Chen, Iran Roman, Roque Lopez, Ethan Brewer, Chen Zhao, et al. 2023. Argus: Visualization of ai-assisted task guidance in ar. *IEEE Transactions on Visualization and Computer Graphics* 30, 1 (2023), 1313–1323.
 - [10] Rui-Che Chang, Yuxuan Liu, and Anhong Guo. 2024. Worldscribe: Towards context-aware live visual descriptions. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–18.
 - [11] Alan Y. Cheng, Meng Guo, Melissa Ran, Arpit Ranasaria, Arjun Sharma, Anthony Xie, Khuyen N. Le, Bala Vinathirathan, Shihe (Tracy) Luan, David Thomas Henry Wright, Andrea Cuadra, Roy Pea, and James A. Landay. 2024. Scientific and Fantastical: Creating Immersive, Culturally Relevant Learning Experiences with Augmented Reality and Large Language Models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI '24*). Association for Computing Machinery, New York, NY, USA, Article 275, 23 pages. <https://doi.org/10.1145/3613904.3642041>
 - [12] Neil Chulpongsatorn, Mille Skovhus Lunding, Nishan Soni, and Ryo Suzuki. 2023. Augmented Math: Authoring AR-Based Explorable Explanations by Augmenting Static Math Textbooks. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) (UIST '23). Association for Computing Machinery, New York, NY, USA, Article 92, 16 pages. <https://doi.org/10.1145/3586183.3606827>
 - [13] Ishita Dasgupta, Christine Kaeser-Chen, Kenneth Marino, Arun Ahuja, Sheila Babayan, Felix Hill, and Rob Fergus. 2023. Collaborating with language models for embodied reasoning. arXiv:2302.00763 [cs.LG] <https://arxiv.org/abs/2302.00763>
 - [14] Mustafa Doga Dogan, Eric J Gonzalez, Karan Ahuja, Ruofei Du, Andrea Colaço, Johnny Lee, Mar Gonzalez-Franco, and David Kim. 2024. Augmented Object Intelligence with XR-Objects. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) (UIST '24). Association for Computing Machinery, New York, NY, USA, Article 19, 15 pages. <https://doi.org/10.1145/3654777.3676379>
 - [15] Fiona Draxler, Audrey Labrie, Albrecht Schmidt, and Lewis L. Chuang. 2020. Augmented Reality to Enable Users in Learning Case Grammar from Their Real-World Interactions. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI '20*). Association for Computing Machinery, New York, NY, USA, 1–12. <https://doi.org/10.1145/3313831.3376537>
 - [16] Ruofei Du, Alex Olwal, Mathieu Le Goc, Shengzhi Wu, Danhang Tang, Yinda Zhang, Jun Zhang, David Joseph Tan, Federico Tombari, and David Kim. 2022. Opportunistic Interfaces for Augmented Reality: Transforming Everyday Objects into Tangible 6DoF Interfaces Using Ad hoc UI. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (*CHI EA '22*). Association for Computing Machinery, New York, NY, USA, Article 183, 4 pages. <https://doi.org/10.1145/3491101.3519911>
 - [17] Runlin Duan, Chenfei Zhu, Yuzhao Chen, Yichen Hu, Jingyu Shi, and Karthik Ramani. 2025. DesignFromX: Empowering Consumer-Driven Design Space Exploration through Feature Composition of Referenced Products. In *Proceedings of the 2025 ACM Designing Interactive Systems Conference (DIS '25)*. Association for Computing Machinery, New York, NY, USA, 1040–1060. <https://doi.org/10.1145/3715336.3735824>
 - [18] Liam Franco Esparraguera, Kristoffer Selberg, Brian Lou, Jenny Sun, Beza Desta, Andrés Monroy-Hernández, and Parastoo Abtahi. 2024. Breaking the Plane: Exploring Real-Time Visualization of 3D Surfaces in Augmented Reality with Handwritten Input. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI EA '24*). Association for Computing Machinery, New York, NY, USA, Article 62, 9 pages. <https://doi.org/10.1145/3613905.3651032>
 - [19] Danilo Gasques, Janet G. Johnson, Tommy Sharkey, Yuanyuan Feng, Ru Wang, Zhuoqun Robin Xu, Enrique Zavala, Yifei Zhang, Wanze Xie, Xinming Zhang, Konrad Davis, Michael Yip, and Nadir Weibel. 2021. ARTEMIS: A Collaborative Mixed-Reality System for Immersive Surgical Telementoring. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) (*CHI '21*). Association for Computing Machinery, New York, NY, USA, Article 662, 14 pages. <https://doi.org/10.1145/3411764.3445576>
 - [20] Daniele Giunchi, Nels Numan, Elia Gatti, and Anthony Steed. 2024. Dream-CodeVR: Towards Democratizing Behavior Design in Virtual Reality with Speech-Driven Programming. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. 579–589. <https://doi.org/10.1109/VR58804.2024.00078>
 - [21] Terrell Glenn, Ananya Ipsita, Caleb Carithers, Kylie Pepler, and Karthik Ramani. 2020. StoryMakAR: Bringing Stories to Life With An Augmented Reality & Physical Prototyping Toolkit for Youth. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI '20*). Association for Computing Machinery, New York, NY, USA, 1–14. <https://doi.org/10.1145/3313831.3376790>
 - [22] Ivan Grishchenko, Valentin Bazarevsky, Andrei Zanfir, Eduard Gabriel Bazavan, Mihai Zanfir, Ruijia Yee, Luuk van der Meer, Timofey Ignatov, Matthias Grundmann, and Cristian Sminchisescu. 2022. BlazePose GHUM Holistic: Real-time 3D Human Landmarks and Pose Estimation. arXiv preprint arXiv:2206.11678 (2022). <https://arxiv.org/abs/2206.11678>
 - [23] Aditya Gunturu, Yi Wen, Nandi Zhang, Jarin Thundathil, Rubaiat Habib Kazi, and Ryo Suzuki. 2024. Augmented Physics: Creating Interactive and Embedded Physics Simulations from Static Textbook Diagrams. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) (UIST '24). Association for Computing Machinery, New York, NY, USA, Article 144, 12 pages. <https://doi.org/10.1145/3654777.3676392>
 - [24] Jiajing Guo, Andrew Benton, Nan Tian, William Ma, Nicholas Feffer, Zhengyu Zhou, and Liu Ren. 2023. EyeClick: A Robust Two-Step Eye-Hand Interaction for Text Entry in Augmented Reality Glasses. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) (UIST '23 Adjunct). Association for Computing Machinery, New York, NY, USA, Article 91, 4 pages. <https://doi.org/10.1145/3586182.3615814>
 - [25] Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiaowu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. arXiv:2308.00352 [cs.AI] <https://arxiv.org/abs/2308.00352>
 - [26] Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 2023. 3D-LLM: Injecting the 3D World into Large Language Models. arXiv:2307.12981 [cs.CV] <https://arxiv.org/abs/2307.12981>
 - [27] Chia Hsu, Yu Chen, Yu-Jung Liu, Yu-Cheng Chang, and Min-Jui Lee. 2023. Spelland: Situated Language Learning with a Mixed-Reality Spelling Game through Everyday Objects. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) (*CHI EA '23*). Association for Computing Machinery, New York, NY, USA, Article 597, 6 pages. <https://doi.org/10.1145/3544549.3583830>
 - [28] Chenxu Du, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. ChatDB: Augmenting LLMs with Databases as Their Symbolic Memory. arXiv:2306.03901 [cs.AI] <https://arxiv.org/abs/2306.03901>
 - [29] Xiyun Hu, Dizhi Ma, Fengming He, Zhengzhe Zhu, Shao-Kang Hsia, Chenfei Zhu, Ziyi Liu, and Karthik Ramani. 2025. GesPrompt: Leveraging Co-Speech Gestures to Augment LLM-Based Interaction in Virtual Reality. In *Proceedings of the 2025 ACM Designing Interactive Systems Conference*. 59–80.
 - [30] Intel. 2025. Intel® RealSense™ Depth Camera D435i - Product Specifications. <https://www.intel.com/content/www/us/en/products/sku/190004/intel-realsense-depth-camera-d435i/specifications.html>
 - [31] Trivial Interactive. 2019. Roslyn C#. <https://forum.unity.com/threads/released-roslyn-c-runtime-c-compiler.651505/>
 - [32] Yhonatan Iquiapaza, Jorge Wagner, and Luciana Nedel. 2024. DeAR: Combining Desktop and Augmented Reality for Visual Data Analysis. In *Proceedings of the 25th Symposium on Virtual and Augmented Reality* (Rio Grande, Brazil) (*SVR '23*). Association for Computing Machinery, New York, NY, USA, 233–237. <https://doi.org/10.1145/3625008.3625021>
 - [33] Rahul Jain, Jingyu Shi, Runlin Duan, Zhengzhe Zhu, Xun Qian, and Karthik Ramani. 2023. Ubi-TOUCH: Ubiquitous Tangible Object Utilization through Consistent Hand-object interaction in Augmented Reality. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–18.
 - [34] Nuwan Janaka, Shengdong Zhao, David Hsu, Sherisse Tan Jing Wen, and Chun Keat Koh. 2024. Demonstrating TOM: A Development Platform for Wearable Intelligent Assistants in Daily Activities. In *Adjunct Proceedings of the 26th International Conference on Mobile Human-Computer Interaction*. 1–6.
 - [35] Hamraz Javaheri, Omid Ghamarnejad, Paul Lukowicz, Gregor Alexander Stavrou, and Jakob Karolus. 2024. ARAS: LLM-Supported Augmented Reality Assistance System for Pancreatic Surgery. In *Companion of the 2024 on ACM*

- International Joint Conference on Pervasive and Ubiquitous Computing* (Melbourne VIC, Australia) (*UbiComp '24*). Association for Computing Machinery, New York, NY, USA, 176–180. <https://doi.org/10.1145/3675094.3677543>
- [36] Hamraz Javaheri, Omid Ghamarnejad, Paul Lukowicz, Gregor Alexander Stavrou, and Jakob Karolus. 2024. ARAS: LLM-Supported Augmented Reality Assistance System for Pancreatic Surgery. In *Companion of the 2024 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Melbourne VIC, Australia) (*UbiComp '24*). Association for Computing Machinery, New York, NY, USA, 176–180. <https://doi.org/10.1145/3675094.3677543>
- [37] Glenn Jocher. 2023. *Ultralytics YOLOv8*. <https://docs.ultralytics.com/models/yolov8/>
- [38] Kwanghee Jung, Vinh T. Nguyen, and Jaehoon Lee. 2021. BlockyXR: An Interactive Extended Reality Toolkit for Digital Storytelling. *Applied Sciences* 11, 3 (2021). <https://doi.org/10.3390/app11031073>
- [39] Seokbin Kang, Ekta Shokeen, Virginia L. Byrne, Leyla Norooz, Elizabeth Bon-signore, Caro Williams-Pierce, and Jon E. Froehlich. 2020. ARMath: Augmenting Everyday Life with Math Learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI '20*). Association for Computing Machinery, New York, NY, USA, 1–15. <https://doi.org/10.1145/3313831.3376252>
- [40] Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tenenholz. 2022. MRKL Systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. arXiv:2205.00445 [cs.CL] <https://arxiv.org/abs/2205.00445>
- [41] Amina Kobenova, Cyan DeVeaux, Samyak Parajuli, Andrzej Banburski-Fahey, Judith Amores Fernandez, and Jaron Lanier. 2024. Social Conjuring: Multi-User Runtime Collaboration with AI in Building Virtual 3D Worlds. arXiv:2410.00274 [cs.HC] <https://arxiv.org/abs/2410.00274>
- [42] Junhan Kong, Dena Sabha, Jeffrey P Bigham, Amy Pavel, and Anhong Guo. 2021. TutorialLens: Authoring Interactive Augmented Reality Tutorials Through Narration and Demonstration. In *Proceedings of the 2021 ACM Symposium on Spatial User Interaction* (Virtual Event, USA) (*SUI '21*). Association for Computing Machinery, New York, NY, USA, Article 16, 11 pages. <https://doi.org/10.1145/3485279.3485289>
- [43] Jaewook Lee, Jun Wang, Elizabeth Brown, Liam Chu, Sebastian S. Rodriguez, and Jon E. Froehlich. 2024. GazePointAR: A Context-Aware Multimodal Voice Assistant for Pronoun Disambiguation in Wearable Augmented Reality. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI '24*). Association for Computing Machinery, New York, NY, USA, Article 408, 20 pages. <https://doi.org/10.1145/3613904.3642230>
- [44] Germán Leiva, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2020. Pronto: Rapid Augmented Reality Video Prototyping Using Sketches and Enaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI '20*). Association for Computing Machinery, New York, NY, USA, 1–13. <https://doi.org/10.1145/3313831.3376160>
- [45] Jiefeng Li, Yingying She, Fang Liu, Chun Yu, Xiaoli Wang, and Yuxin Xu. 2022. Augmented Reality Based Video Shooting Guidance for Novice Users. *Proc. ACM Hum.-Comput. Interact.* 6, MHCI, Article 215 (Sept. 2022), 20 pages. <https://doi.org/10.1145/3546750>
- [46] Jiajia Li, Zixia Zheng, Xiemin Wei, and Guanyun Wang. 2021. FaceMe: An Augmented Reality Social Agent Game for Facilitating Children's Learning about Emotional Expressions. In *Adjunct Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) (*UIST '21 Adjunct*). Association for Computing Machinery, New York, NY, USA, 17–19. <https://doi.org/10.1145/3474349.3480216>
- [47] Lawrence Lim, Wei-Yee Goh, Mara Downing, and Misha Sra. 2021. A Spatial Music Listening Experience in Augmented Reality. In *Adjunct Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) (*UIST '21 Adjunct*). Association for Computing Machinery, New York, NY, USA, 23–25. <https://doi.org/10.1145/3474349.3480218>
- [48] Tica Lin, Rishi Singh, Yalong Yang, Carolina Nobre, Johanna Beyer, Maurice A. Smith, and Hanspeter Pfister. 2021. Towards an Understanding of Situated AR Visualization for Basketball Free-Throw Training. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) (*CHI '21*). Association for Computing Machinery, New York, NY, USA, Article 461, 13 pages. <https://doi.org/10.1145/3411764.3445649>
- [49] Ziyi Liu, Zhengzhe Zhu, Enze Jiang, Feichi Huang, Ana M Villanueva, Xun Qian, Tianyi Wang, and Karthik Ramani. 2023. InstruMentAR: Auto-Generation of Augmented Reality Tutorials for Operating Digital Instruments Through Recording Embodied Demonstration. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) (*CHI '23*). Association for Computing Machinery, New York, NY, USA, Article 32, 17 pages. <https://doi.org/10.1145/3544548.3581442>
- [50] Ziyi Liu, Zhengzhe Zhu, Lijun Zhu, Enze Jiang, Xiyun Hu, Kylie A Pepler, and Karthik Ramani. 2024. ClassMeta: Designing Interactive Virtual Classmate to Promote VR Classroom Participation. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI '24*). Association for Computing Machinery, New York, NY, USA, Article 659, 17 pages. <https://doi.org/10.1145/3613904.3642947>
- [51] Mathias N. Lystbæk, Ken Pfeuffer, Tobias Langlotz, Jens Emil Sloth Grønbaek, and Hans Gellersen. 2024. Spatial Gaze Markers: Supporting Effective Task Switching in Augmented Reality. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI '24*). Association for Computing Machinery, New York, NY, USA, Article 633, 11 pages. <https://doi.org/10.1145/3613904.3642811>
- [52] Dizhi Ma, Xiyun Hu, Jingyu Shi, Mayank Patel, Rahul Jain, Ziyi Liu, Zhengzhe Zhu, and Karthik Ramani. 2024. avaTTAR: Table Tennis Stroke Training with Embodied and Detached Visualization in Augmented Reality. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) (*UIST '24*). Association for Computing Machinery, New York, NY, USA, Article 35, 16 pages. <https://doi.org/10.1145/3654777.3676400>
- [53] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. 2004. DART: a toolkit for rapid design exploration of augmented reality experiences. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (Santa Fe, NM, USA) (*UIST '04*). Association for Computing Machinery, New York, NY, USA, 197–206. <https://doi.org/10.1145/1029632.1029669>
- [54] Andrii Matvienko, Sebastian Günther, Sebastian Ritzenhofen, and Max Mühlhäuser. 2022. AR Sightseeing: Comparing Information Placements at Outdoor Historical Heritage Sites using Augmented Reality. *Proc. ACM Hum.-Comput. Interact.* 6, MHCI, Article 194 (Sept. 2022), 17 pages. <https://doi.org/10.1145/3546729>
- [55] Bartosz Mazurkiewicz, Marcelo de Lima Galvão, and Ioannis Giannopoulos. 2023. BeeAR: Augmented Reality Beeline Navigation for Spatial Knowledge Acquisition. *Proc. ACM Hum.-Comput. Interact.* 7, MHCI, Article 199 (Sept. 2023), 17 pages. <https://doi.org/10.1145/3604246>
- [56] Meta. 2023. Interaction SDK v72. <https://developers.meta.com/horizon/reference/interaction/v72/>
- [57] Meta. 2025. Meta Quest Pro Headset For Business. <https://forwork.meta.com/quest/quest-pro/>
- [58] Meta. 2025. Mixed Reality Utility Kit Overview. <https://developers.meta.com/horizon/documentation/unity/unity-mr-utility-kit-overview/>
- [59] Kyzyl Monteiro, Ritik Vatsal, Neil Chulpongatorn, Aman Parnami, and Ryo Suzuki. 2023. Teachable Reality: Prototyping Tangible Augmented Reality with Everyday Objects by Leveraging Interactive Machine Teaching. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) (*CHI '23*). Association for Computing Machinery, New York, NY, USA, Article 459, 15 pages. <https://doi.org/10.1145/3544548.3581449>
- [60] Leon Müller, Ken Pfeuffer, Jan Gugenheimer, Bastian Pflöging, Sarah Prange, and Florian Alt. 2021. SpatialProto: Exploring Real-World Motion Captures for Rapid Prototyping of Interactive Mixed Reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) (*CHI '21*). Association for Computing Machinery, New York, NY, USA, Article 363, 13 pages. <https://doi.org/10.1145/3411764.3445560>
- [61] Tobias Müller, Mark Colley, Gülsemin Dogru, and Enrico Rukzio. 2022. AR4CAD: Creation and Exploration of a Taxonomy of Augmented Reality Visualization for Connected Automated Driving. *Proc. ACM Hum.-Comput. Interact.* 6, MHCI, Article 177 (Sept. 2022), 27 pages. <https://doi.org/10.1145/3546712>
- [62] Vinh T. Nguyen, Kwanghee Jung, and Tommy Dang. 2020. BlockyAR: A Visual Programming Interface for Creating Augmented Reality Experiences. *Electronics* 9, 8 (2020). <https://doi.org/10.3390/electronics9081205>
- [63] OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2023). <https://arxiv.org/abs/2303.08774>
- [64] OpenAI. 2025. OpenAI Text-to-Speech API. <https://platform.openai.com/docs/guides/text-to-speech>
- [65] Zekun Qi, Runpei Dong, Shaochen Zhang, Haoran Geng, Chunrui Han, Zheng Ge, Li Yi, and Kaisheng Ma. 2024. ShapeLLM: Universal 3D Object Understanding for Embodied Interaction. arXiv:2402.17766 [cs.CV] <https://arxiv.org/abs/2402.17766>
- [66] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative Agents for Software Development. arXiv:2307.07924 [cs.SE] <https://arxiv.org/abs/2307.07924>
- [67] Xun Qian, Fengming He, Xiyun Hu, Tianyi Wang, Ananya Ipsita, and Karthik Ramani. 2022. SealAR: Authoring Semantically Adaptive Augmented Reality Experiences in Virtual Reality. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (*CHI '22*). Association for Computing Machinery, New York, NY, USA, Article 65, 18 pages. <https://doi.org/10.1145/3491102.3517665>
- [68] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. arXiv:2307.16789 [cs.AI] <https://arxiv.org/abs/2307.16789>

- [69] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Whisper: Robust Speech Recognition via Large-Scale Weak Supervision. <https://github.com/openai/whisper>
- [70] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (2009), 60–67.
- [71] Mark Richardson, Fadi Botros, Yangyang Shi, Pinhao Guo, Bradford J Snow, Linguang Zhang, Jingming Dong, Keith Vertanen, Shugao Ma, and Robert Wang. 2024. StegoType: Surface Typing from Egocentric Cameras. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) (*UIST '24*). Association for Computing Machinery, New York, NY, USA, Article 83, 14 pages. <https://doi.org/10.1145/3654777.3676343>
- [72] Jasmine Roberts, Andrzej Banburski-Fahey, and Jaron Lanier. 2022. Steps towards prompt-based creation of virtual worlds. arXiv:2211.05875 [cs.HC] <https://arxiv.org/abs/2211.05875>
- [73] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face. arXiv:2303.17580 [cs.CL] <https://arxiv.org/abs/2303.17580>
- [74] Jingyu Shi, Rahul Jain, Seungeun Chi, Hyungjun Doh, Hyung-gun Chi, Alexander J Quinn, and Karthik Ramani. 2025. CARING-AI: Towards Authoring Context-aware Augmented Reality INstruction through Generative Artificial Intelligence. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–23.
- [75] Jingyu Shi, Rahul Jain, Hyungjun Doh, Ryo Suzuki, and Karthik Ramani. 2023. An HCI-centric survey and taxonomy of human-generative-AI interactions. arXiv preprint arXiv:2310.07127 (2023).
- [76] Ludwig Sidenmark, Dominic Potts, Bill Bapisch, and Hans Gellersen. 2021. Radi-Eye: Hands-Free Radial Interfaces for 3D Interaction using Gaze-Activated Head-Crossing. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) (*CHI '21*). Association for Computing Machinery, New York, NY, USA, Article 740, 11 pages. <https://doi.org/10.1145/3411764.3445697>
- [77] Avinash Kumar Singh, Jia Liu, Carlos A. Tirado Cortes, and Chin-Teng Lin. 2021. Virtual Global Landmark: An Augmented Reality Technique to Improve Spatial Navigation Learning. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) (*CHI EA '21*). Association for Computing Machinery, New York, NY, USA, Article 276, 6 pages. <https://doi.org/10.1145/3411763.3451634>
- [78] Sketchfab. 2023. Sketchfab Data API v3. <https://sketchfab.com/developers/data-api/v3>
- [79] Aimee Sousa Calepso, Natalie Hube, Noah Berenguel Senn, Vincent Brandt, and Michael Sedlmair. 2022. cARdLearner: Using Expressive Virtual Agents when Learning Vocabulary in Augmented Reality. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) (*CHI EA '22*). Association for Computing Machinery, New York, NY, USA, Article 245, 6 pages. <https://doi.org/10.1145/3491101.3519631>
- [80] Xia Su, Jon E. Froehlich, Eunye Koh, and Chang Xiao. 2024. SonifyAR: Context-Aware Sound Generation in Augmented Reality. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) (*UIST '24*). Association for Computing Machinery, New York, NY, USA, Article 128, 13 pages. <https://doi.org/10.1145/3654777.3676406>
- [81] Xia Su, Han Zhang, Kaiming Cheng, Jaewook Lee, Qiaochu Liu, Wyatt Olson, and Jon E. Froehlich. 2024. RASSAR: Room Accessibility and Safety Scanning in Augmented Reality. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI '24*). Association for Computing Machinery, New York, NY, USA, Article 968, 17 pages. <https://doi.org/10.1145/3613904.3642140>
- [82] Ryo Suzuki, Rubaiat Habib Kazi, Li-yi Wei, Stephen DiVerdi, Wilmot Li, and Daniel Leithinger. 2020. RealitySketch: Embedding Responsive Graphics and Visualizations in AR through Dynamic Sketching. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) (*UIST '20*). Association for Computing Machinery, New York, NY, USA, 166–181. <https://doi.org/10.1145/3379337.3415892>
- [83] Xiao Tang, Xiaowei Hu, Chi-Wing Fu, and Daniel Cohen-Or. 2020. GrabAR: Occlusion-aware Grabbing Virtual Objects in AR. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) (*UIST '20*). Association for Computing Machinery, New York, NY, USA, 697–708. <https://doi.org/10.1145/3379337.3415835>
- [84] Joao Marcelo X N Teixeira and Julia D T De Souza. 2024. Augmented Reading. In *Proceedings of the 24th Symposium on Virtual and Augmented Reality* (Natal, RN, Brazil) (*SVR '22*). Association for Computing Machinery, New York, NY, USA, 55–61. <https://doi.org/10.1145/3604479.3604511>
- [85] Natalia Teixeira, Bianca Lahm, Fabiana Frata Furlan Peres, Claudio Roberto Marquette Mauricio, and Joao Marcelo Xavier Natario Teixeira. 2022. Augmented Reality on Museums: the Ecomuseu Virtual Guide. In *Proceedings of the 23rd Symposium on Virtual and Augmented Reality* (Virtual Event, Brazil) (*SVR '21*). Association for Computing Machinery, New York, NY, USA, 147–156. <https://doi.org/10.1145/3488162.3488219>
- [86] Fernanda De La Torre, Cathy Mengying Fang, Han Huang, Andrzej Banburski-Fahey, Judith Amores Fernandez, and Jaron Lanier. 2024. LLMR: Real-time Prompting of Interactive Worlds using Large Language Models. arXiv:2309.12276 [cs.HC] <https://arxiv.org/abs/2309.12276>
- [87] Hongyu Wan, Jinda Zhang, Abdulaziz Arif Suria, Bingsheng Yao, Dakuo Wang, Yvonne Coady, and Mirjana Prpa. 2024. Building LLM-based AI Agents in Social Virtual Reality. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI EA '24*). Association for Computing Machinery, New York, NY, USA, Article 65, 7 pages. <https://doi.org/10.1145/3613905.3651026>
- [88] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (March 2024). <https://doi.org/10.1007/s11704-024-40231-1>
- [89] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Yuanzhi Cao, and Karthik Ramani. 2021. GesturAR: An Authoring System for Creating Freehand Interactive Augmented Reality Applications. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) (*UIST '21*). Association for Computing Machinery, New York, NY, USA, 552–567. <https://doi.org/10.1145/3472749.3474769>
- [90] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Ke Huo, Yuanzhi Cao, and Karthik Ramani. 2020. CAPTurAR: An Augmented Reality Tool for Authoring Human-Involved Context-Aware Applications. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) (*UIST '20*). Association for Computing Machinery, New York, NY, USA, 328–341. <https://doi.org/10.1145/3379337.3415815>
- [91] Tianyi Wang, Xun Qian, Fengming He, and Karthik Ramani. 2021. LightPaintAR: Assist Light Painting Photography with Augmented Reality. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) (*CHI EA '21*). Association for Computing Machinery, New York, NY, USA, Article 237, 6 pages. <https://doi.org/10.1145/3411763.3451672>
- [92] Guande Wu, Jing Qian, Sonia Castelo Quispé, Shaoyu Chen, João Rulff, and Claudio Silva. 2024. ARTIST: Automated Text Simplification for Task Guidance in Augmented Reality. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (*CHI '24*). Association for Computing Machinery, New York, NY, USA, Article 939, 24 pages. <https://doi.org/10.1145/3613904.3642772>
- [93] Haoyuan Wu, Zhuolun He, Xinyun Zhang, Xufeng Yao, Su Zheng, Haisheng Zheng, and Bei Yu. 2024. ChatEDA: A Large Language Model Powered Autonomous Agent for EDA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43, 10 (Oct. 2024), 3184–3197. <https://doi.org/10.1109/icaad.2024.3383347>
- [94] Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. 2023. TidyBot: personalized robot assistance with large language models. *Autonomous Robots* 47, 8 (Nov. 2023), 1087–1102. <https://doi.org/10.1007/s10514-023-10139-z>
- [95] Yihong Wu, Lingyun Yu, Jie Xu, Dazhen Deng, Jiachen Wang, Xiao Xie, Hui Zhang, and Yingcai Wu. 2023. AR-Enhanced Workouts: Exploring Visual Cues for At-Home Workout Videos in AR Environment. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) (*UIST '23*). Association for Computing Machinery, New York, NY, USA, Article 121, 15 pages. <https://doi.org/10.1145/3586183.3606796>
- [96] Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. 2023. Embodied Task Planning with Large Language Models. arXiv:2307.01848 [cs.CV] <https://arxiv.org/abs/2307.01848>
- [97] Yuchen Xia, Manthan Shenoy, Nasser Jazdi, and Michael Weyrich. 2023. Towards autonomous system: flexible modular production system enhanced with large language model agents. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 1–8. <https://doi.org/10.1109/etfa54631.2023.10275362>
- [98] Chang Xiao, Ryan Rossi, and Eunye Koh. 2022. iMarker: Instant and True-to-scale AR with Invisible Markers. In *Adjunct Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) (*UIST '22 Adjunct*). Association for Computing Machinery, New York, NY, USA, Article 32, 3 pages. <https://doi.org/10.1145/3526114.3558721>
- [99] Yunhao Xing, Que Liu, Jingwu Wang, and Diego Gomez-Zara. 2024. sMoRe: Enhancing Object Manipulation and Organization in Mixed Reality Spaces with LLMs and Generative AI. arXiv:2411.11752 [cs.HC] <https://arxiv.org/abs/2411.11752>
- [100] Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. 2024. PointLLM: Empowering Large Language Models to Understand Point Clouds. arXiv:2308.16911 [cs.CV] <https://arxiv.org/abs/2308.16911>
- [101] Bufang Yang, Yunqi Guo, Lilin Xu, Zhenyu Yan, Hongkai Chen, Guoliang Xing, and Xiaofan Jiang. 2024. SocialMind: LLM-based Proactive AR Social Assistive System with Human-like Perception for In-situ Live Interactions. arXiv:2412.04036 [cs.AI] <https://arxiv.org/abs/2412.04036>

- [102] Zhengzhe Zhu, Ziyi Liu, Youyou Zhang, Lijun Zhu, Joey Huang, Ana M Vilanueva, Xun Qian, Kylie Pepler, and Karthik Ramani. 2023. LearnIoTVR: An End-to-End Virtual Reality Environment Providing Authentic Learning Experiences for Internet of Things. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) (*CHI '23*). Association for Computing Machinery, New York, NY, USA, Article 447, 17 pages. <https://doi.org/10.1145/3544548.3581396>

A Appendix

A.1 System Meta Prompt

Table 5: The meta-prompt for agentAR.

Your job is to understand users' requests and generate structured plans for building various AR applications on a prototyping platform. This platform consists of a frontend layer and a backend layer. The backend layer takes input from sensory devices and is responsible for user and environmental perception, reasoning, and data processing, and the frontend layer is responsible for rendering visual content, presenting user interfaces, and supporting other interactive elements that users directly perceive and engage with. The plans you generate are intended to guide the subsequent program composition using a dedicated tool library.

For the frontend plan, you must specify the tools involved, arguments passed from the backend, and pseudo-code that outlines the structure of the layer. For the backend plan, you must define the arguments, return values, tools involved, and also pseudo-code outlines the structure of the layer.

You must refer to the available tools in {{Available Tool List}} when composing your plans. Do NOT specify tools that do not exist. Use only the available tools to fulfill the user's requests. If a requested functionality is not supported by the current toolset, you should either reject the request or find a feasible alternative that achieves the same purpose, and notify the user accordingly. To support more accurate task parsing and planning, you are provided with several demonstrations:

{{Demonstrations}}

Each demonstration pairs a user request with the expected output plan, illustrating how to interpret user intent, determine tool dependencies, and construct appropriate application structures and data flows. You should refer to these demonstrations to guide your planning. You are also provided with an environment image and the planning history {{Planning History}}. These contextual cues should be incorporated into your reasoning to generate more coherent and adaptive application plans.

Be mindful of the system's limitations. If a feature cannot be supported by existing tools, you must reject the request. If an alternative method can fulfill the intent, propose it to the user. If further clarification is needed, ask the user for more information. Only when all necessary details are collected should you proceed with generating the plans.

Your final output must follow the specified template:

Plan for backend:

- Arguments:
- Returns:
- Tools involved:
- Structure:

Plan for frontend:

- Tools involved:
- Backend arguments:
- Structure:

A.2 Tool Format and Definition

Table 6: Unified format for *Frontend* and *Backend* tools.

<i>Frontend</i> Tool	<i>Backend</i> Tool
{Tool name}	{Tool name}
Position	Description
Orientation	Arguments
Activation (Visibility)	Returns
{Custom interfaces}	

Table 7: Example definition of a *Backend* tool.

```
class DetectionHelper:
    """Handless object detection functionalities."""
    def __init__(self):
        self.model = YOLO("yolov8n.pt")
    def DetectFrame(self, **kwargs) -> dict:
        """
        Performs object detection on an input image and returns a
        list of detected objects, their bounding boxes, and labels.
        Arguments:
            ColorImage (numpy.ndarray): The input image.
        Returns:
            dict: A dictionary containing:
            - 'ListOfObjects' (list[dict]): A list of detected objects.
              Each object is represented as:
            - 'LABEL' (str): The class label of the detected object.
            - 'U1' (int): The x-coordinate of the top-left corner
              of the bounding box.
            - 'V1' (int): The y-coordinate of top-left.
            - 'U2' (int): The x-coordinate of bottom-right.
            - 'V2' (int): The y-coordinate of bottom-right.
        """
        res = {}
        res["ListOfObjects"] = []
        results = self.model(kwargs["ColorImage"])[0]
        for result in results.bboxes:
            u1, v1, u2, v2 = map(int, result.xyxy[0])
            if result.conf[0]>0.5:
                cls = int(result.cls[0])
                label = f"{self.model.names[cls]}"
                oneObject = {'LABEL':label, 'U1':u1, 'V1':v1, \
                    'U2':u2, 'V2':v2}
                res["ListOfObjects"].append(oneObject)
        return res
```

A.3 Tool Implementation Details

Object Detection & Tracking. We adopt Ultralytics YOLOv8 [37] for object detection and tracking. YOLOv8 supports real-time object

Table 8: Example definition of a *Frontend* tool. All *Frontend* tools inherit from the abstract class *AbstractHelper*.

```
// Abstract base class for handling UI-related functionalities. The
// default position is set to global (0,0,0). The default orientation is
// to track the headset.
public abstract class AbstractHelper: MonoBehaviour
{
    // Gets whether the GameObject is currently visible (active)
    // in the scene.
    public bool IsVisible;
    // Enables or disables visibility.
    public void SetVisibility(bool tf);
    // Enables position tracking, making the GameObject follow
    // another object's position with an optional offset.
    public virtual void SetFollowingPosition(
        GameObject followPositionObject, Vector3? offset = null);
    // Enables orientation tracking, making the GameObject's
    // Z-axis face a target object with an optional rotation offset.
    public virtual void SetFollowingOrientation(
        GameObject followOrientationObject,
        Quaternion? offset = null);
    // Disables position tracking and sets a fixed position.
    public virtual void SetStaticPosition(Vector3 staticPosition,
        Vector3? offset = null);
    // Disables orientation tracking and sets a fixed rotation.
    public virtual void SetStaticOrientation(
        Quaternion staticOrientation, Quaternion? offset = null);
}

// A helper class for a UI slider.
public class SliderHelper: AbstractHelper
{
    // Gets and sets the current value of the slider.
    public float Value;
    // Sets the title text displayed above the slider.
    public void SetTitle(string title);
    // Sets the minimum value and updates the label.
    public void SetMin(float min);
    // Sets the maximum value and updates the label.
    public void SetMax(float max);
    // Registers a callback function for slider value changes.
    public void AddOnValueChangedCallback(
        UnityAction<float> action);
```

detection and tracking with high precision. Its lightweight architecture and ease of deployment make it well-suited for integration into dynamic AR scenarios.

Human Pose Estimation. Our human pose estimation tool supports both third-person and first-person pose estimation. For third-person pose estimation, we implemented the Mediapipe BlazePose GHUM 3D [22] model. For first-person pose estimation, we leveraged the built-in body pose detection function of Quest Pro AR-HMD provided by Meta.

Hand Gesture Detection & Hand Tracking. We utilize the built-in hand-tracking function provided by Meta Interaction SDK [56] for hand gesture detection and hand tracking. These tools leverage the headset's onboard cameras to analyze users' hand gestures and movements in real-time.

Gaze Tracking. We utilized the built-in eye-tracking function provided by Meta Interaction SDK. This tool enables acquiring eye direction for downstream tasks.

Speech Recognition & Speech Synthesis. We implemented the lightweight, real-time Whisper [69] model for speech recognition and TTS-1 [64] for speech synthesis.

LLM. We utilized GPT-4o as our LLM tool. Our implementation enables text and image input to GPT-4o to enable versatile usage in AR applications.

OCR. We implement both general text OCR and math OCR using GPT-4o, guided by specialized prompts tailored for each task.

Math Simulator. We implemented a math simulator leveraging GPT-4o. Our implementation for the math simulator is designed to interpret monomial functions, identify their coefficients, and generate sampled data.

Physics Simulator. We implemented a physics simulator powered by GPT-4o. Our implementation generates simulated 3D movement data of virtual objects based on natural language descriptions.

Controller. We use the paired controllers of the Meta Quest Pro HMD to provide additional input, supporting precise tracking of their position and orientation. This enables more versatile and interactive AR applications through controller-based interactions.

Data Visualizer. We implement a data visualizer to render two types of data in AR applications: body poses and mathematical plots. Our implementation uses multiple forms and colors to distinguish different data elements and enhance clarity and comprehension.

Animator. We leveraged Unity's built-in Animator to implement basic animations such as scaling, translation, and rotation. These animations are defined using animation clips and managed through Animator Controllers, enabling smooth playback and easy integration into AR application workflows.

User Interface. We designed a set of user interface elements to enhance user interaction with virtual content in AR. While there are numerous custom user interfaces in AR applications, we implemented the three most commonly used options for our tool library: buttons, sliders, and text boxes. These interfaces provide versatile interaction mechanisms for different AR applications.

3D Assets. We leveraged the open-source asset library Sketchfab [78] to support the utilization of versatile 3D assets in AR applications. To enable on-demand asset creation, we implemented an object retrieval tool using Sketchfab API to search for assets based on textual input. It then downloads and imports them into AR applications.

Supplementary Tools. We implemented several supplementary tools to support more versatile AR application creation. These include a timer, a record/playback tool, a video player, and a scene

interaction tool. The record/playback tool allows users to record and replay object movements and tracked poses. The video player supports playing three local videos in the AR application. The scene interaction tool, backed by Meta's MRUK [58], allows attaching trigger functions to scene objects, detected objects, or designated volume.

A.4 Case Study Prompts

Augmented Object [14]. "Create an AR app where users can press a 'Detect' button to find nearby objects. The system takes a picture, identifies the objects, gets their 3D positions, and uses AI to describe them. For each object, show tools like an info button to show the description, a notepad, a timer, and a chat button to talk with AI. All tools face the user and stay hidden until turned on."

avaTTAR [52]. "I want to create an AR application to track and visualize the person's body pose in front of me. A button 'Show Track' follows the headset and toggles the visibility of the human pose visualizer. There should be a ping pong paddle model attached to the right controller, which I will give to the person in front of me in order to track his hand movement. A record/replay system is also needed. It records/replays the body pose and the paddle."

Augmented Math [12]. "Detect the math expression in front of the user. When an expression is detected, an x-y plot is created to visualize the math expression, and the user is able to adjust the coefficients in the math expression. Here is some descriptions of the UI elements. The "Detect" button should be hidden upon the successful detection of math equations."

AhUI [16]. "I want to create an AR application where a video player appears about a meter in front of me when I press a 'Start' button. Then, floating above my left hand, there should be a 'STOP' button to play/pause the video and a 'Next' button stacked above it to skip to the next video. The UI should follow my left hand, and the video player should face me."

AR-Enhanced Workouts [95]. "I want an application that can record and replay the coach's body pose in front of me. Also, I want to record and playback my own body pose so that I can check whether my body pose is aligned with the coach's body posture."

GazePointAR [43]. "Develop an AR application that allows users to interact with objects in their environment using eye tracking and gesture recognition. When a user does a left thumb up gesture, the system should detect the object they are looking at, retrieve relevant information, and provide a chatbot that the user can talk with about the detected object. The application should integrate a speech-to-text for the user to interact with the chatbot and a text-to-speech for the chatbot's responses."

A.5 Additional Examples Details

Interactive Paper. "I want an app that lets users capture text around them, shows it in a scrollable box, and reads it out loud. Users can also edit the text for notes. It's great for classrooms, offices, or on the go."

Gesture Learning Assistant. "The app shows a gesture like thumbs up or stop. When users try it, the system checks their hand pose and gives instant feedback—success sound and confetti if correct, retry sound if not."

Ball Catching Game. "I want an app where users catch virtual balls with their right hand. It tracks their movement to see if they

grab it. A new ball comes every 5 seconds, and a slider changes the speed."

Table 9: The meta-prompt used to create new AR applications.

You are a novice user with no programming or AR design experience. Your task is to communicate your ideas for AR applications using natural language in a conversation with an AI system. You should describe what you want the application to do in terms of goals, user experience, or intended behavior. Keep the idea simple to work in a single scene, with no advanced logic or multiple steps.

Avoid low-level design instructions such as specific object coordinates, technical terminology, or implementation details. Instead, talk about things you want to happen, like "I want to catch a ball" or "It should show me some text."

Here are some examples to guide you:

{{Demonstration}}

Now, describe your own AR app idea:

A.6 GPT-4 Score Prompt

Table 10: The meta-prompt for GPT-4 Score evaluation.

As a critic, your task is to assess whether the LLM agent has properly planned an AR application based on the user's request. To do so, carefully examine both the user's request and the agent's output, and then provide a decision using either "Yes" or "No" ("Yes" indicates accurate planning, and "No" indicates inaccurate planning). Additionally, provide a rationale for your choice using the following structure: {"choice": "yes"/"no", "reason": "Your reason for your choice"}.

Please adhere to the following guidelines: 1. The tool must be selected from the following options: {{ Available Tool List }}. 2. Please note that there exists a logical relationship and order between the tools. 3. Simply focus on the correctness of the planning without considering the tool arguments.

Current user request: {{Input}} AI assistant's output: {{Output}}

Your judgment:

A.7 Example Code Generated by agentAR

A.8 Full List of Reviewed AR Research

Table 11: Example *Frontend* program.

```

public class DerivedClass : BaseClassForRoslynCode
{
    public void GazeInteractionWithChatbot()
    {
        EyeTrackingHelper eyeTrackingHelper = GameObject.Instantiate(eyeTrackingPrefab).GetComponent<EyeTrackingHelper>();
        GestureDetectionHelper gestureDetectionHelper =
            GameObject.Instantiate(gestureDetectionPrefab).GetComponent<GestureDetectionHelper>();
        gestureDetectionHelper.AddWhenSelectFunction(SupportedGestures.ThumbsUpLeft, () => DetectGazeRoutine());
        void DetectGazeRoutine()
        {
            Vector3 eyeDirection = eyeTrackingHelper.GetAverageDirection();
            Vector3 eyeDirectionOpenCV = externalCamera.TransformDirectionUnity2OpenCV(eyeDirection);
            UDPArgumentConstructor constructor = new UDPArgumentConstructor();
            constructor.AddXYZ(eyeDirectionOpenCV);
            string SENM = pythonRoutineHelper.ExecutePythonRoutine("DetectObjectByGaze", constructor.GetArgumentString());
            pythonRoutineHelper.AddPythonRoutineOnFinishCallbackBySENM(SENM, DetectObjectByGazeCallback);
        }
        void DetectObjectByGazeCallback(UDPResult res)
        {
            ListOfObjects detectedObject = UDPResult.DeserializeResult<ListOfObjects>(res.REST);
            ObjectData obj = detectedObject.listOfObjects[0];
            VoiceChatHelper voiceChatHelper = GameObject.Instantiate(voiceChatPrefab).GetComponent<VoiceChatHelper>();
            voiceChatHelper.SetFollowingPosition(HMD, new Vector3(0, -0.15f, 0.4f));
            voiceChatHelper.SetFollowingOrientation(HMD);
            voiceChatHelper.SetTitle(obj.LABEL);
            voiceChatHelper.SetMetaPrompt("The user is looking at a " + obj.LABEL + ". The system description is: " + obj.Response +
                " Please engage in a conversation around it.");
        }
    }
}

```

Table 12: Example *Backend* program.

```

camHelper = CameraHelper()
detHelper = DetectionHelper()
imageUtilsHelper = ImageUtilsHelper()
visionHelper = VisionHelper()
def DetectObjectByGaze(**kwargs):
    res = {}
    res['ListOfObjects'] = []
    while True:
        convertedUV = camHelper.GetUVofDirection(**kwargs)
        frame = camHelper.GetFrame()
        DetectResult = detHelper.DetectFrame(**frame)
        if not DetectResult['ListOfObjects']:
            continue
        nearestObject = imageUtilsHelper.FindNearestObjectByUV(**convertedUV, **DetectResult)
        an_obj = nearestObject['ListOfObjects'][0]
        QueryResult = visionHelper.ProcessImageQuery(**frame, **an_obj)
        res['ListOfObjects'].append(**an_obj, **QueryResult)
    yield res
    break

```

Table 13: Full list of reviewed AR research.

Short Title	Core Functionality	Year	Conference
ARMath [39]	AR system that helps children explore math through everyday objects	2020	CHI
Case Grammar [15]	Handheld AR app teaches case grammar using nearby objects	2020	CHI
GrabAR [83]	Bare-hand interaction with virtual objects by predicting occlusion masks	2020	UIST
RealitySketch [82]	AR interface for sketching interactive graphics	2020	UIST
Radi-Eye [76]	Use gaze & head-crossing to select widgets	2021	CHI
Free-Throw [48]	Visualizations in basketball free-throw training	2021	CHI
ARTEMIS [19]	Remote surgical mentoring using real-time guidance and asynchronous video	2021	CHI
LightPaintAR [91]	AR interface with virtual light traces to create light paintings	2021	CHI EA
Virtual Landmark [77]	AR landmarks enhance navigation and spatial learning	2021	CHI EA
Spatial Music [47]	AR app spatializes music for immersive, customizable live-like listening	2021	UIST
FaceMe [46]	Virtual agent and tangible tools to help children with ASD learn emotions	2021	UIST Adjunct
AR Widget [6]	AR-enhanced smartphone widgets that offload interface elements into the air	2021	MHCI
cARdLearner [79]	AR flashcards use emotion-linked holograms to support vocabulary learning	2022	CHI EA
Opportunistic Interfaces [16]	Enable virtual interfaces on everyday objects	2022	CHI EA
Augmented Reading [84]	AR-based implementations of Bionic Reading and Spritz	2022	SVR
AR Museum [85]	Mobile AR virtual guidance system for museum	2022	SVR
AR DCIM [1]	AR application that display essential information for data center maintenance	2022	SVR
AR4CAD [61]	AR visualizations for infrastructure-supported automated driving	2022	MHCI
AR Sightseeing [54]	Presenting historical AR content at outdoor heritage sites	2022	MHCI
ARCAM [45]	AR mobile video shooting camera movement guidance system	2022	MHCI
iMarker [98]	Invisible AR markers enable accurate, low-cost, single-camera tracking	2022	UIST
MoonBuddy [4]	Voice-enabled AR app aiding communication, navigation, and documentation	2022	UIST Adjunct
InstruMentAR [49]	AR system auto-generates tutorials from multimodal user demonstrations	2023	CHI
Spelland [27]	Enable situated language learning using everyday objects	2023	CHI EA
EyeClick [24]	Eye-hand text entry for AR HMDs utilizing a modified keyboard	2023	UIST Adjunct
Augmented Math [12]	AR explanations by augmenting static equations	2023	UIST
BeeAR [55]	AR navigation system with always-visible digital landmarks	2023	MHCI
DeAR [32]	Hybrid desktop-AR for synchronized data visualization and interaction	2023	SVR
RAMM [2]	AR system for manual PCB assembly that reduces error and improves quality	2023	SVR
Breaking the Plane [18]	Visualize mathematical functions in 3D with handwritten input	2024	CHI EA
PANDALens [7]	Proactive AI narrative documentation assistant on AR	2024	CHI
OmniActions [35]	Processes sensory inputs and predicts follow-up actions with LLMs	2024	CHI
Scientific & Fantastical [11]	AR app uses narrative and LLMs to teach science	2024	CHI
GazePointAR [43]	Use gaze and point gesture to enhance speech queries	2024	CHI
RASSAR [81]	Use LiDAR camera to detect and visualize indoor accessibility issues	2024	CHI
ARTIST [92]	AR system auto-simplifies text to reduce cognitive load	2024	CHI
Gaze Markers [51]	AR markers to infer POI after attention shift	2024	CHI
SonifyAR [80]	Generates context-aware sound effects with LLM in AR	2024	UIST
XR-Objects [14]	Enable digital interaction with real-world objects	2024	UIST
StegoType [71]	Enable surface typing with camera-based hand-tracking	2024	UIST
Augmented Physics [23]	Embedded interactive physics simulations from textbook diagrams	2024	UIST
avaTTAR [52]	Provide visual cues for table tennis stroke training	2024	UIST