General-Purpose Task Guidance from Natural Language in Augmented Reality using Vision-Language Models

Daniel J. Stover

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

A. Lynn Abbott, Chair

Doug A. Bowman, Co-chair

Chris Thomas

Creed Jones

May 1, 2024

Blacksburg, Virginia

Keywords: Augmented Reality, Machine Learning, Task Guidance

General-Purpose Task Guidance from Natural Language in Augmented Reality using Vision-Language Models

Daniel J. Stover

(ABSTRACT)

Augmented reality task guidance systems provide assistance for procedural tasks, which require a sequence of physical actions, by rendering virtual guidance visuals within the real-world environment. An example of such a task would be to secure two wood parts together, which could display guidance visuals indicating the user to pick up a drill and drill each screw. Current AR task guidance systems are limited in that they require AR system experts for use, require CAD models of real-world objects, or only function for limited types of tasks or environments. We propose a general-purpose AR task guidance approach and proof-of-concept system to generate guidance for tasks defined by natural language. Our approach allows an operator to take pictures of relevant objects and write task instructions for an end user, which are used by the system to determine where to place guidance visuals. Then, an end user can receive and follow guidance even if objects change location or environment. Guidance includes reusable visuals that display generic actions, such as our system's 3D hand animations. Our approach utilizes current vision-language machine learning models for text and image semantic understanding and object localization. We built a proof-of-concept system using our approach and tested its accuracy and usability in a user study. We found that all operators were able to generate clear guidance for tasks in an office room, and end users were able to follow the guidance visuals to complete the expected action 85.7% of the time without knowledge of their tasks. Participants rated that our system was easy to use to generate guidance visuals they expected.

General-Purpose Task Guidance from Natural Language in Augmented Reality using Vision-Language Models

Daniel J. Stover

(GENERAL AUDIENCE ABSTRACT)

Augmented Reality (AR) task guidance systems provide assistance for tasks by placing virtual guidance visuals on top of the real world through displays. An example of such a task would be to secure two wood parts together, which could display guidance visuals indicating the user to pick up a drill and drill each screw. Current AR task guidance systems are limited in that they require AR system experts for use, require detailed models of real-world objects, or only function for limited types of tasks or environments. We propose a new task guidance approach and built a system to generate guidance for tasks defined by written instructions. Our approach allows an operator to take pictures of relevant objects and write task instructions for an end user, which are used by the system to determine where to place digital visuals. Then, an end user can receive and follow guidance even if objects change location or environment. Guidance includes visuals that display generic actions, such as our system's 3D hand animations that mimic human hand actions. Our approach utilizes AI models for text and image understanding and object detection. We built a proof-of-concept system using our approach and tested its accuracy and usability in a user study. We found that all operators were able to generate clear guidance for tasks in an office room, and end users were able to follow the guidance visuals to complete the expected action 85.7% of the time without knowledge of the tasks. Participants rated that our system made it easy to write instructions and take pictures to create guidance visuals.

Dedication

I want to thank my fellow Virginia Tech 3DI lab-mates Christos L., Cory I., Alex G., Ibrahim T., Kylie D., Leo P., Logan L., and Shakiba D. for their assistance and company during the majority of my Master's degree, my roommates Evan A. and Siraj S. for being there throughout this journey, and of course all of my family and friends.

Contents

List of Figures viii

1 Introduction 1

1.1 Motivation . . . 1

1.2 Background . . . 5

1.3 Problem Statement . . . 7

1.4 Research Questions . . . 11

1.5 Approach . . . 12

1.6 Contributions . . . 15

2 Related Work 16

2.1 Manual Authoring Systems . . . 17

2.2 Semi-Automated Authoring Systems . . . 20

2.3 Automated Authoring Systems . . . 22

2.4 Enabling Automated General-Purpose Task Guidance Systems . . . 24

3 Approach/System Design 28

3.1 System Assumptions . . . 29

3.2 Predefined Guidance Visuals . . . 30

3.3 Operator Phase . . . 31
3.3.1 Operator Instructions and Pictures . . . 33
3.3.2 Semantic Understanding . . . 34
3.3.3 Localization Block . . . 39
3.3.4 Rendering Block . . . 42
3.4 End User Phase . . . 44
3.4.1 Scanning Objects . . . 44
3.4.2 Using Generated Task Guidance . . . 46
3.5 General-Purpose Task Guidance System Design . . . 48

4 User Study Design, Results, and Discussion 50

4.1 Study Design . . . 50
4.1.1 Apparatus . . . 51
4.1.2 Tasks . . . 52
4.1.3 Measures . . . 56
4.1.4 Participants . . . 58
4.1.5 Procedure . . . 58
4.2 Results . . . 59
4.3 Discussion . . . 65
4.3.1 General-Purpose System Validation . . . 65

4.3.2 System Usability by Non-Experts 68

4.3.3 Further System Analysis 71

5 Conclusion and Future Work **82**

5.1 Conclusion 82

5.2 Future Work 84

5.2.1 Future Evaluations 84

5.2.2 Future System Enhancements 85

Bibliography **90**

Code and Attributions **99**

Appendices **100**

Appendix A User Study Materials **101**

A.1 Consent Form 102

A.2 Pre-Study Questionnaire 106

A.3 Post-Study Interview Questions 108

A.4 Participant Recruitment Email 110

List of Figures

1.1 Approach Overview for Operator (1) and End User (2) for the task of starting the coffee machine. The yellow hands are being displayed as an example of generated guidance from the system. 13

2.1 GPT-4V input example using OpenAI's ChatGPT interface [2]. 26

2.2 GroundingDINO input examples from [16]. Colored bounding boxes are displayed in each image to detect objects from the text prompt below each image. 27

3.1 Predefined Action Visuals for Our System 30

3.2 System Design for Operator Phase 32

3.3 GPT-4V System Prompt 34

3.4 Earlier Iteration of GPT-4V System Prompt 36

3.5 Block Diagram for Semantic Understanding 37

3.6 Block Diagram for Localization with prompt "top screw hole of coffee machine". 40

3.7 Another example of outputs from the first and second pass of GroundingDINO in Localization Block with prompt "white coffee maker lid". 41

3.8 Block Diagram for Rendering 42

3.9 System Design for Scanning Step of End User Phase 45

3.10 Menus for End User During Scanning Step 45

3.11 Using AR Task Guidance Generated by Our System 47

4.1 Picture of the office used for the user study. 51

4.2 Pictures of objects used for user study 52

4.3 Mean accuracy of participants in the end-user phase completing the correct action from the previous operator's instructions 59

4.4 Distribution of the number of trials it took participants to obtain an accurate output for each instruction during the operator phase. The whiskers on each box plot show the minimum and maximum number of trials for the instruction related to the object on the x-axis, and the black cross marker shows the mean number of trials for the instruction. 61

4.5 Mean accuracy of all guidance outputs' visual and location (a; on top) or just action visual (b; on bottom) per trial number for all participants during the operator phase. Black error bars for each trial show the maximum and minimum output accuracy values from all of the participants during that trial. 62

4.6 Post-study questionnaire quantitative responses. The black cross marker for each question indicates the mean response value over all participants. An up arrow next to the question number signifies that a higher number is more favorable, while a down arrow means a lower number is more favorable. **Q1:** "Did the proposed system provide guidance visuals that you expected for your input text instructions?"
"Given the guidelines, how easy was it to write instructions to get an expected output?"
"Given the guidelines, how easy was it to take good pictures to get an expected output?"
"When needing to go back and revise instructions and/or pictures after an output was incorrect, was it easy to know what to change?"
"Did those changes make the output better?" 64

4.7 Pictures of fan and light switch compared between first and final trials for one participant. The green dot shows the output from object detection during Localization. 70

4.8 Breakdown of causes of incorrect outputs for participants during the operator phase. 72

4.9 Initial operator instruction leading to incorrect GPT-4V output (top), compared to updated operator instruction leading to correct GPT-4V output (bottom) 75

4.10 Breakdown of causes of participants' performing incorrect tasks based on guidance visuals during the end-user phase. 77

4.11 Comparison of operator (a and c) and end-user images (b and d) that resulted in different object detection locations. The green dot in each image represents the output from object detection. 78

4.12 Picture of an operator image of the light switch compared to an end-user image of the light switch, which should try to be matched. The green dot in each image represents the output from object detection. 79

List of Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
AR	Augmented Reality
CAD	Computer-Aided Design
HWD	Head-Worn Display
ML	Machine Learning
RGB	Red-Green-Blue
RQ	Research Question
VLM	Vision-Language Model

Chapter 1

Introduction

1.1 Motivation

Across various domains, individuals have to perform difficult procedural tasks in a wide variety of environments.

Definition 1. A **procedural task** "involves people performing established sequences of activities [or actions], while interacting with objects in the physical environment, to accomplish specific goals" [13].

The term "task" will be used as a shorthand for procedural tasks.

Definition 2. An **environment** "refers to the physical context where users are interacting" [14].

Some examples of people who regularly perform procedural tasks include technicians, construction workers, carpenters, or even chefs. Consider technicians in manufacturing plants or production floors, who are tasked with a specific set of ordered tasks to complete during their shift. Examples of their tasks could include "test the electronics board" or "set up the production robot". Construction workers and carpenters also have tasks but in different environments, such as "attach the wooden support". Cooking is another example where the

cook may have the task of "add 2 cups of water to the pot", requiring them to interact with kitchen appliances and cooking utensils.

Typically, there are a few main approaches for training or guiding individuals with less experience through these tasks, including human one-on-one tutoring, video tutorials, or written manuals [6, 13, 14, 17]. Human one-on-one tutoring involves an expert in the task teaching the new individual typically by demonstrating the task in person to the trainee, whether this being how to use a machine for production, how to use a construction tool, or how to cook certain parts of a recipe [14]. Video tutorials involve transferring this knowledge from an expert to a trainee remotely through a pre-recorded 2D video without the expert needing to be there in person [17]. Written instructions, on the other hand, could be a technical manual for technicians with tons of written information, such as text or graphics, to teach them how to perform their tasks with specific machines or equipment [30]. This could also include a written recipe in a book to instruct a cook on how to complete each step. These methods can be used initially to train an individual, or continuously to guide individuals each time they need to perform the task.

Human one-on-one tutoring has the benefit of showing trainees exactly where and how to perform each step in space, and can adapt to learner uncertainties, but requires the human expert's time and presence for tutoring [14, 30]. Video tutorials have the benefit of being reusable, only needing to be created once by an expert, but "suffer from the lack of a spatial connection between the digital representation and the user's physical presence" [6, 14]. Written manuals are also reusable, but "can be difficult and time-consuming to interpret, and only show static information" [30]. A useful approach that attempts to merge these benefits is to use Augmented Reality to teach and guide users through tasks. Augmented Reality (AR) is an interface with the main characteristics of "(1) the combination of virtual and physical elements, (2) being interactive in real-time, and (3) being registered in 3D"

[11]. AR systems have the benefits of human onboarding since 3D visuals can be placed next to relevant real-world objects required for the task [6, 17, 30]. They also have the benefits of video tutorials because they are configured by an expert for multiple uses, so novice users can receive guidance at any time [17]. Thus, AR task guidance systems are applications that guide users through real-world tasks using relevant digital 3D visuals placed next to relevant real-world objects that the user needs to interact with. The 3D visuals should show the user how to interact with objects to complete their task [17].

Researchers have conducted studies finding that AR task guidance systems have advantages over other methods of task guidance. First, Gavish et al. found that users who were trained for the task of assembling an electronic actuator with an AR system had fewer unsolved errors during an assembly task than those trained using a pre-recorded video tutorial [10]. This encourages the future use of AR task guidance in training. Additionally, other researchers compared the effectiveness of AR task guidance with guidance from a traditional 2D LCD screen for the task of assembling an engine combustion chamber. They implemented an AR system to help guide users through an assembly task, and they found that "AR was faster and more accurate for psychomotor phase activities, was overwhelmingly preferred, and considered to be more intuitive" [13]. These findings help motivate further research and development of AR task guidance systems.

Despite these benefits, current AR task guidance systems have several primary limitations. One limitation is that some current AR systems typically only work for the specific environment and objects they are made for. This often includes a system expert manually placing guidance visuals on real-world objects in 3D software, such as Unity. For example, a system expert may create visuals showing an automotive technician where to place and use a wrench to tighten a bolt. However, if the apparatus containing the bolt moves locations, the guidance would have to be re-created, or the environment would need to be prepared

perfectly with several markers.

Another limitation includes systems requiring detailed Computer-Aided Design (CAD) models of relevant objects in the environment. For example, a technician may utilize a vibration testing machine and an AR task guidance system may be configured with a 3D CAD model of this machine. Hence, this same AR system will not understand any other machine, even one with the same buttons but in different locations. The final major limitation is that current systems are often limited to certain types of tasks or environments. For instance, guidance that can only be created for assembly tasks, or environments with specific machines.

These limitations are challenging to solve all at once because it requires a system that can understand beyond one scenario or environment, as explained by Grubert et al: "The biggest challenge is to build an AR interface that is actually context-controlled and moves away from the single purpose AR application ... This requires not only additional context sources but also more complex context modeling approaches to infer the current context" [11]. To address these limitations, AR task guidance systems need to be capable of understanding multiple types of tasks and objects without 3D models or markers, without needing a system expert to manually place guidance visuals in software.

To develop an AR task guidance system with this level of understanding in various contexts, Machine Learning (ML) techniques can be applied to "increase the adaptability and versatility of AR systems" [23]. ML is an advancing field of computer science that uses algorithms capable of providing results on inputs, such as images and text, that have never been seen before by training the ML algorithm (referred to as a "model"), which nowadays are typically deep networks using the Transformer architecture, on similar or general data [5, 28].

For example, current object detection ML models are capable of outputting bounding boxes around specified objects in an image, or localization tasks, by being trained on multitudes of

examples initially curated by humans [31]. The state-of-the-art in machine learning has been advancing quickly, with multimodal models capable of understanding both text and images. Moreover, ML models are becoming increasingly more general, capable of understanding larger sets of images and text [12, 16, 31]. These advancements have enabled the ability to solve previously infeasible problems in several fields.

Thus, an AR task guidance system with the benefits of more general language and scene understanding (through images) can be more robust to general environments and scenarios. Additionally, the use of these models can make AR task guidance systems more usable by individuals who are non-experts in the system since both the individual and the system can intuitively understand text and images. One motivational example driven by the benefits of using ML for AR applications is its use in "Industry 4.0", the next anticipated technological revolution in manufacturing. Reviews from researchers Sahija and Sahu, among others, validate the need for AR systems that use ML (AR+ML) for remote guidance in manufacturing among other use cases [22, 23]. They also address the limitations of current AR+ML systems and specify possible challenges for future systems to overcome, such as being reconfigurable and robust in a variety of environments, as well as not requiring several complex inputs to function.

1.2 Background

When developing an AR task guidance approach, AR hardware is important to consider. Surveys of current AR task guidance approaches show that AR HWDs are most commonly used. In 2015, a survey of AR use in "through-life engineering applications" found that the most popular hardware used in these applications was HWDs at 58% [8]. These applications were used in industries such as automotive and aerospace for tasks such as maintenance

and inspection. A similar survey conducted in 2018 on AR applications in maintenance also found that the most popular hardware used for these applications was HWDs at 30% [19].

To enable general-purpose AR systems, Grubert et al. proposed a "Pervasive Augmented Reality" taxonomy, which defines critical components of a context-aware AR system [11]. Since our proposed approach is a general-purpose task guidance system, it will need to consider aspects of their context-aware framework, specifically environmental context factors. In a task guidance system, the geometry of the objects relevant to completing the user's tasks is useful for a system to understand in order to display guidance. Most modern AR systems include spatial mapping capabilities, which typically use a depth sensor to build up a 3D geometrical representation of the space around the user. For instance, the Microsoft HoloLens 2 includes a front-facing infrared time-of-flight depth sensor for spatial mapping [27]. In addition to a geometrical representation of the environment, a general-purpose task guidance system will also need to know what the environment actually looks like, such as specific objects required for tasks. Current AR systems also typically include front-facing RGB cameras, such as the HoloLens 2's PhotoVideo camera [27].

Several types of AR task guidance systems exist for a variety of tasks. Examples of various types of tasks in manufacturing include assembly and disassembly, repair or setup, and training. The survey above found that out of these common task types, current task guidance applications are being used 33% of the time for assembly/disassembly tasks, 26% for repair, and 15% for training [19]. Assembly/disassembly includes putting objects together, repair or setup includes interacting with objects in specific ways depending on the object, and training includes how to use an object for the first time. Each of these tasks requires the AR task guidance system to have visuals set up by someone who knows how to perform the tasks correctly, who is titled the "operator".

As discussed previously, machine learning could be useful for AR task guidance systems.

Two such ML fields include Natural Language Processing (NLP) and Computer Vision (CV), which enable software to understand language and 2D images, which are fundamental modes of information to humans. The advancement of NLP using current ML models has enabled the integration of AI chatbots in modern products, such as OpenAI's ChatGPT and GPT-4 [18]. Advancements in CV with modern ML models have enabled the previously infeasible task of self-driving cars working in general scenarios on the road, such as Waymo's autonomous vehicles.

In general, the usability of ML models, such as object detection models, is increasing since they are shifting more towards becoming text promptable with open-set vocabulary. This means that any free-form text prompt is a valid input to the model [31]. ML researchers explain that recent model architectures internally fuse their language representations with their vision representations (encodings), which allow text prompts to affect many layers of the model, which they state can have comparable effects on the model's weights as actually fine-tuning the model [16]. Models that accept both text and images are called vision-language models, and they are well-suited for enhancing the usability and generality of AR task guidance systems since they are trained to handle general data that is simply understood by humans [12].

1.3 Problem Statement

The primary goal of an AR task guidance system is to display relevant visuals to guide the end user through their tasks. As part of this goal, an operator determines what the end user's tasks are and configures the system. Thus, such a system typically has two users that have to be considered: the operator and the end user. One important aspect of an AR task guidance system is its usability. A critical component of usability for both the

operator and the user of an AR task guidance system is that the system does not require the use of technical skills such as computer programming, modeling, and animation to be set up, used, and reconfigured, which is attributed as a limitation of current AR applications [8]. A system that does not require these technical skills will be referred to as usable by "non-experts", as stated in Definition 3. The generality of an AR task guidance system is also critical, as expressed by Pervasive AR [11]. Thus, we have broken down the necessary components of a general-purpose AR task guidance system below. The goal of this thesis is to demonstrate a proof-of-concept AR task guidance system that is usable by non-experts and is general purpose.

Definition 3. Usable by non-experts means that a system does not require the use of technical skills, such as "computer programming, modeling, and animation" [19].

In a general-purpose AR task guidance system, several components should be present (in no particular order):

Components of a General-Purpose AR Task Guidance System:

- **Component 1:** The system should be capable of generating guidance visuals for tasks about which it has no a priori knowledge.
- **Component 2:** It should be capable of generating guidance visuals in arbitrary environments without special preparation of the environment.
- **Component 3:** It should require little to no complex technical input or setup to generate guidance visuals.
- **Component 4:** It should be able to determine which visuals to place and where to place them.

- **Component 5:** It should be robust to objects moving locations or environments.
- **Component 6:** It should produce clear guidance that is understandable by a naive user.

Definition 4. General tasks are defined as procedural tasks about which a system has no a priori knowledge.

Definition 5. General environments are defined as arbitrary environments with no special preparation.

Component 1 states that these systems should be capable of understanding general tasks. Because these tasks are not known by the system, it will need to be capable of understanding the semantics from any specified task. Thus, an input modality is necessary that can represent any task. Our approach uses natural language to specify open-vocabulary tasks. Open vocabulary means that the operator configuring the system can write free-form task descriptions, also called instructions, in a manner similar to instructing another person. For example, a system should be able to accept and understand instructions with similar semantics but different forms, such as "Add salt to the pot" or "Pick up the salt shaker and shake it over the pot."

Component 2 focuses on general environments instead of tasks. Multiple environments may contain variations in real-world objects and real-world settings. For example, a system instruction may be to "Microwave for 1 minute". However, microwave button panels vary widely, such that the '1', '0', and "start" buttons would be at different locations on many microwaves. Additionally, some microwaves have a single button that quickly sets it to 1

minute. Thus, systems should be capable of being configured by non-experts for different microwave types.

Moreover, **Component 5** considers that the location of the microwave on a kitchen counter may move over time. The system should be initially configurable by non-experts at any location. If an object changes location after operator configuration, the end user should be able to continue to use the system and still receive relevant guidance for their tasks, which should also be the case for the same microwave in entirely different kitchens or environments.

Component 3 discusses that the system should have minimal to no reliance on technical inputs or setup to function, which is highlighted by a review of future AR+ML systems [23]. For example, to make an AR task guidance system usable by non-experts, the system should not rely on needing 3D CAD models for every new object or having to re-train an ML model on pictures of new objects, since developing these CAD or ML models requires technical knowledge that most people do not have. Thus, to provide 3D visual guidance, a system could either automatically generate 3D visuals or have a predefined library of 3D visuals that the system reuses in general settings.

Component 4 then references the capability of the system to automatically determine the placement of guidance visuals, as opposed to a system expert manually placing them. Once a visual is either generated or selected from a predefined library, it needs to be positioned correctly, most likely near the real-world object the task is referencing. For example, the instruction to "Microwave for 1 minute" could lead the system to output a hand visual pressing the correct microwave button or buttons, which should be aligned in front of the correct button and pointing towards the button.

Furthermore, as **Component 6** states, a system with the above capabilities should not compromise the clarity of its outputs for the overall goal of providing guidance for the end user's

tasks. Current task guidance approaches have limitations that have kept them from enabling a general-purpose system with all six components above. This is a challenging problem because such a general-purpose system requires an understanding of general tasks, which we specify using natural language, and their semantics given any general environment without detailed CAD models, while automatically determining where to place visuals without a system expert. We propose an approach and proof-of-concept system designed with these components in mind, using current vision-language ML models to overcome these challenges.

1.4 Research Questions

The main research questions that this research plans to answer are the following:

- **RQ1:** How can we design a general-purpose AR task guidance system with the desired Components 1-6 outlined in Section 1.3?
- **RQ2:** How can this general-purpose AR task guidance system be designed such that it can be set up and used by non-expert users?

RQ1 and RQ2 are the research questions we plan to address by the proposed system and user study. RQ1 focuses on how such a system can be architected to achieve ideal characteristics for a general AR task guidance system while maintaining clear guidance visuals. RQ2 focuses on how such a system can be usable by anybody, which would allow non-expert operators and users in a variety of domains to take advantage of powerful AR task guidance.

1.5 Approach

To address our two research questions, a proposed AR task guidance system was designed, implemented, and tested. Our system was designed with the desired characteristics of a general AR task guidance system in mind while ensuring it is usable by non-experts. It is an AR application used on the Microsoft HoloLens 2 for two users: an operator and an end user. The operator sets up the system by writing text instructions on how to complete the tasks and taking pictures of the relevant objects in each instruction. Once complete, the system will generate 3D guidance for each instruction that the end user can follow. Additionally, even if the locations of objects in the real world change, the end user can continually use the system without needing the operator to reconfigure it by being guided to take pictures of the same objects the operator took pictures of.

An example use case of the implemented approach is shown in Figure 1.1. As shown in the top diagram labeled (1), the operator begins setting up the system by determining what tasks the end user should receive guidance for, then they can write free-form text instructions as input to the task guidance system. In this example, the instruction is "Start coffee machine". Then, the operator takes pictures of the relevant objects for the current instruction, which is the coffee machine in this case. Next, the system will generate a 3D visual as guidance for the instruction. Specifically, our system selects the most relevant 3D visual out of a library of predefined hand actions, including hand pressing, twisting, pulling, and picking up/placing animations. To start the coffee machine, the system may select a hand-pressing visual. Then, the system determines where to place the guidance visual. For this task, it should place the visual such that the hand is pressing the correct start button for the coffee machine. This is done for all of the instructions. Once the operator is done setting up the system, meaning that they believe that the generated guidance visuals are clear enough to

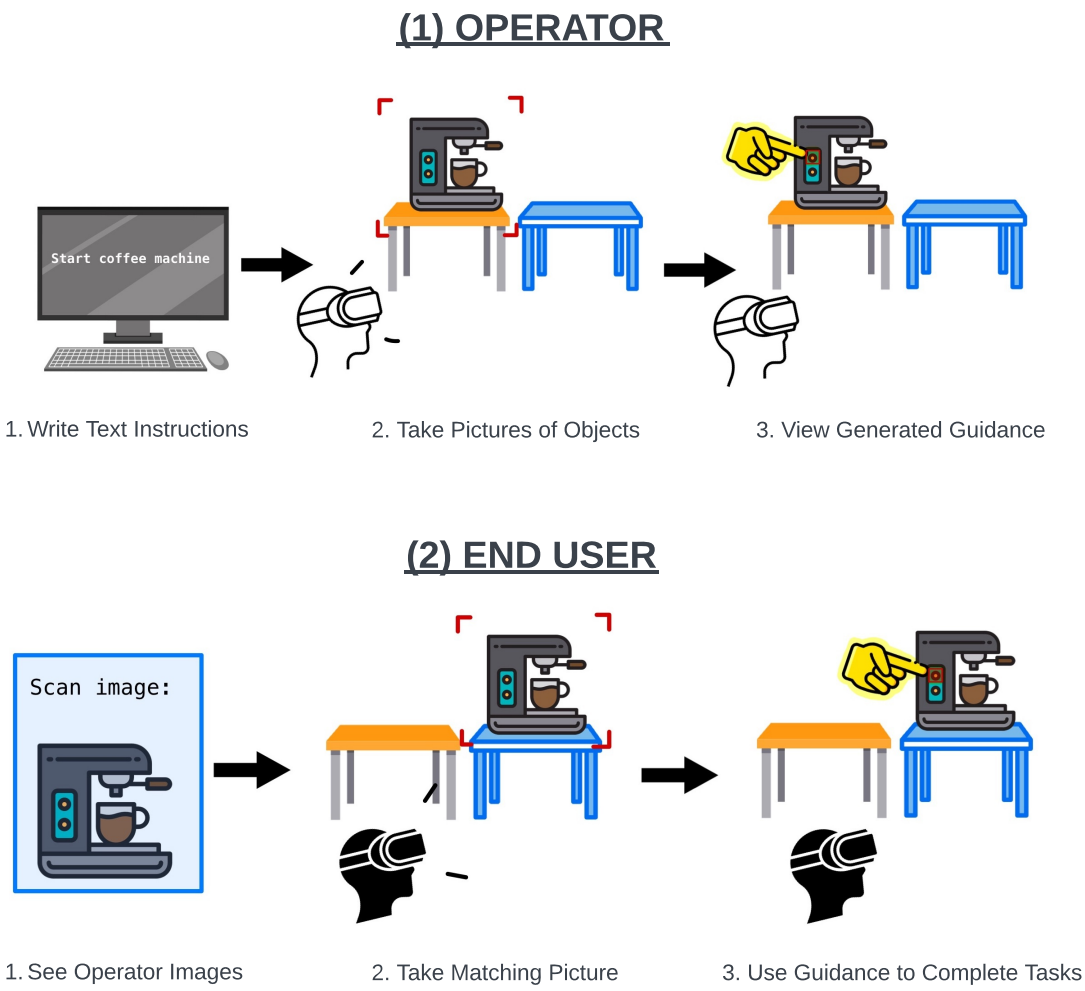


Figure 1.1: Approach Overview for Operator (1) and End User (2) for the task of starting the coffee machine. The yellow hands are being displayed as an example of generated guidance from the system.

help the end user with each task, the system saves information from the operator phase.

Now, the end user is ready to use the system to receive guidance on their tasks, such as how to start the coffee machine, as shown in Figure 1.1 in the bottom diagram labeled (2). First, the end user begins by seeing an image of an object that the operator took. Then, they are instructed to match the operator's image as closely as possible by taking a picture. In this example, the end user will take a matching picture of the same coffee machine. The location

of the object may have changed, but the end user will still take a picture of the matching object. Once all objects have been photographed for the current instruction, the system uses information from the operator's setup to place the guidance visual for the instruction. This is done for all of the images the operator took. Thus, the example user would be able to view the hand-pressing visual on the coffee machine start button, guiding them to press it.

In order to interpret open-vocabulary text instructions and help detect where to place the guidance visual, our system uses current state-of-the-art Vision-Language Models (VLM). The use of VLMs enables the task guidance system to generate guidance with general tasks, such as "Start coffee machine" and general environments, such as different types of coffee machines. Additionally, using VLMs allows the system to interpret simple inputs, such as text and images, which helps the task guidance system be usable by non-experts.

To test the proposed task guidance approach and answer the research questions, a user study was designed and conducted. 12 participants tested the usability of the approach related to RQ2, as well as the effectiveness of the system related to RQ1. The study was conducted in an office room with five overall tasks to complete: making coffee, muting an alarm clock, opening a cabinet drawer, turning on a fan, and turning off the lights. Each participant began as the end user by using the previous participants' operator setup to generate guidance and followed the guidance visuals to complete tasks unknown to them. Then, the participant became the operator, writing instructions and taking pictures of the relevant objects over multiple operator trials until they reported all generated instruction guidance was clear. Results were obtained and analyzed from the system's output for each participant, as well as responses from a post-study interview.

1.6 Contributions

The following are the contributions from this research (in no particular order):

1. An AR task guidance approach and proof-of-concept system using modern vision-language machine learning models to automatically place AR visuals for tasks unknown to the system a priori in arbitrary environments without special preparation. To our knowledge, this is the first AR system using vision-language models for general-purpose task guidance.

2. An AR task guidance approach and proof-of-concept system that is usable by individuals without AR expertise (non-experts) using only natural language text and picture inputs with a predefined library of 3D guidance visuals. To our knowledge, this is the first system of its kind that uses natural language text inputs for task guidance.

3. An AR task guidance approach and proof-of-concept system that is usable by the end user even if the real-world objects move locations or environments, without requiring an operator to reconfigure the system.

4. Empirical evidence from a user study that the proof-of-concept AR task guidance system achieves its goals of generality and usability.

Chapter 2

Related Work

The method of setting up an AR task guidance system to generate guidance visuals is called "authoring". Task guidance systems can be broken down by their method of authoring. A survey of AR in manufacturing categorized current applications by their authoring solutions, which were classified into manual, semi-automated, and automated authoring methods. Only looking at solutions that are used for 3D guidance visuals, they found that 64% of applications used manual authoring, 24% used semi-automated authoring, and the last 4% used automated authoring [19]. Below is a summary of each of these authoring methods:

- **Manual Authoring:** System expert manually determines the placement and alignment of guidance visuals in the 3D space.
- **Semi-Automated Authoring:** System determines the pose of guidance visuals automatically from operator setup but requires complex data to be created by system experts for new tasks and environments.
- **Automated Authoring:** System generates guidance visuals with minimal input or setup for new tasks and environments.

As stated above, automated authoring includes automatically placing 3D guidance visuals. The survey referenced also considers systems that extract guidance from 3D CAD models as automated authoring systems. However, utilizing a 3D CAD model for every object requiring

guidance is complex data that must be created by a modeling expert, and fits more as a semi-automated authoring solution based on the above definition. A true automated authoring method will be defined as one that generates guidance visuals with minimal input or setup for new tasks and environments.

2.1 Manual Authoring Systems

As described above, the majority of current AR applications for manufacturing use manual authoring. The main limitation of these systems for task guidance is that they are only capable of providing guidance for the single use-case they are configured for. Once any components move or the tasks change, the system has to be reconfigured to account for these adaptations. Additionally, reconfiguring these systems requires technical knowledge of AR technology to actually place the 3D visuals, presumably done in a 3D software such as Unity.

Henderson and Feiner had users complete the tasks of locating, positioning, and aligning components of airplane engine combustion chambers [13]. They developed a prototype AR task assistance system on a head-worn display for these tasks, as well as a similar system that displayed guidance on a 2D LCD screen. Their system was capable of guiding users to the correct location of components of interest using 2D and 3D arrows. Then, the system rendered curved paths showing the user where to move components. For alignment, the system displayed 3D rotation arrows along the components that changed size and color as the rotation was conducted. Additionally, the system highlighted the holes needed to align the combustion chamber parts. They ran a user study to compare the completion time and accuracy of the tasks between the HWD AR system, and the 2D LCD system. Results from their user study showed that users completed the alignment task significantly faster and with

fewer errors on average with the AR system than when using the 2D LCD guidance. The system created for this study tracked all relevant components, including the AR headset, using outside-in tracking with several infrared cameras and markers. Thus, guidance visuals were manually authored based on where each combustion chamber component currently is located in space and where it should be next.

Another AR system created for aircraft maintenance and training was developed by De Crescenzio et al [7]. They developed a system that can display 3D task guiding visuals, typically from 3D CAD models, for specific tasks, which they tested on an oil change procedure for an aircraft. They performed a case study and found that users were satisfied with the system and rated low physical and mental workloads during the tasks. To create the visuals, their system enables a creator to place 3D visuals in a virtual coordinate system, which is linked to the real-world coordinate system attached to the aircraft apparatus using a markerless feature tracking approach. This manual authoring method of placing the 3D visuals in a virtual coordinate frame has the limitation of requiring technical expertise in a 3D CAD interface.

More recent work on a self-aware training and assistant AR system was created to guide a user through an assembly task they had no prior experience performing [26]. The user had to utilize multiple tools, such as a screwdriver and drill, to assemble a desktop carving machine in a stationary workspace. This task guidance system determines the user's current state from a trained activity recognition model to decide which task guidance step to display to them. They ran a case study and found that users had faster completion times and lower errors on average when using the AR system compared to doing the task without it. The 3D guidance visuals were manually placed for each step based on a tracked AR marker by the system expert.

Park et al. developed a system as part of their AR task assistance method that generates

3D guidance visuals by using deep learning to detect and segment the object of interest and extract either its 3D shape or 2D segmentation image for use as the guidance visual instead of needing a 3D CAD model [20]. They conducted a user study where users had to perform a 3D printer maintenance task with their proposed task guidance system using deep learning and with one that used traditional AR markers to place simple visuals. Their proposed system used object detection to determine where each relevant object was located and help the user find them when needed. Additionally, their system displayed task guidance visuals using the actual extracted 3D or 2D replicas to help guide users through their 3D printer maintenance tasks. Overall, they found that when using their proposed task guidance system, users completed the tasks quicker on average, and the researchers attributed this primarily to the users not having to search for tools since the system guides the user to the correct location. To generate the 3D guidance visuals, a remote system expert can use a 3D software tool to manually place each visual relative to each other in the AR environment.

Upon scenario changes, such as different tasks or a new environment, our approach does not require system experts to determine where to place visuals since it only requires text instructions and pictures of objects to be re-authored. For example, if the system is set up for the task of "open machine door", but the physical machine the system was configured for has been replaced by a new one with a different door location, the operator of our system simply needs to take a picture of the new machine to reconfigure the output task guidance visual. In this example, current manually authored systems would need a system expert to manually move the visual to the new door location on the machine. Additionally, in a case where our system is set up for task guidance with this machine, and the machine moves location in the environment, our approach would not require an operator to reconfigure the guidance. The user would simply take a picture of the machine, and the guidance visuals would update as expected. In current manually authored systems, moving an object such as this machine

would require a system expert to replace guidance visuals, unless the environment is set with AR markers, requiring expert setup of its own in every environment.

Thus, our proposed AR task guidance approach addresses the main limitation of manually authored task guidance systems by eliminating the need for the system to be reconfigured by a system expert upon task or environment changes.

2.2 Semi-Automated Authoring Systems

Semi-automated authoring is different from manual authoring in that a system expert does not need to manually place visuals, typically the system determines their placement using discrete inputs from the system user. The inputs for this type of authoring system typically address the need for authoring to be more usable by non-experts, attempting to remove the need for an expert in AR technology to set up the system. However, semi-automated authoring systems are still limited in that they typically require complex data, such as CAD models, to generate guidance visuals in new tasks or environments.

Zogopoulos et al. proposed an authoring tool that can generate 3D visual instructions for common manufacturing tasks, including assembly/disassembly and surface treatment (such as gluing) [32]. Their tool requires a CAD model input for the object involved in the task, then it can automatically extract assembly steps from the model to display to the user. Additionally, the authoring tool allows for additional operator input, such as where to drill holes, intermediate steps between assembly, or surface treatment path and width. The system then generates guidance visuals from this tool, displaying views of the 3D model before and after assembly, where parts are located, or the exact surface treatment path for example. While guidance is extracted from CAD model assemblies and other operator input

in a usable authoring tool, the system has the limitation of requiring a full 3D CAD model for real-world objects, which is complex data that needs to be created by a system expert for each new object.

AdapTutAR is an adaptive AR tutorial system that was developed to guide users through tasks, such as machine tasks of twisting knobs and turning levers [14]. The system provides various guidance visuals at different levels of detail, including a virtual avatar, 3D animations and arrows, and text. To generate these visuals, the system requires a virtual CAD model of the machine or object that is physically aligned with the real machine. Then, the operator records each action to perform, and the system generates a virtual avatar that follows the actions of the operator, as well as animations from the CAD model that are aligned with the real-world action. Their system also determines when actions are complete using a state recognition convolutional neural network that they trained on images of their knobs at each setting for instance. After the operator finishes, a user can follow the guidance generated by the system to help perform their tasks. The authoring solution is usable by non-experts since it only requires an operator recording phase, but requiring an aligned CAD model is complex data that needs to be created by an expert for each new machine or environment.

Erkoyuncu et al. developed a task guidance system called ARAUM (Augmented Reality Authoring for Maintenance) focused on streamlining the authoring process [9]. Their system enabled operators to use an authoring platform to define fields such as which operation to perform and warnings to trigger from certain events. From these inputs, as well as a CAD model of the object of interest, the system determines how to place the 3D model for the operation using markerless feature-based tracking, as well as how to animate the visual. This authoring method is usable by operators without AR technical expertise, and they found that using their authoring method to generate visuals is significantly quicker than manually placing visuals using the Vuforia tracking SDK. However, a limitation of ARAUM's authoring

approach is that it requires a CAD model of the object being manipulated. Similar to other semi-automated authoring systems, this complex data needs to be created by an expert for each new environment.

Current semi-automated authoring systems make task guidance systems more usable by non-experts, as shown. However, they still have the main drawback of typically requiring the creation of CAD models to generate their guidance visuals. Needing a CAD model of every object for the user's tasks takes lots of time and expertise. Moreover, these models would have to be created by an expert each time the environment changes. This requirement limits the task guidance system's applicability in new tasks that involve different objects.

Thus, we address the limitations of current semi-automated authoring methods by making our authoring approach model-free to ensure that reconfiguring the system for new tasks and environments does not require the creation of 3D models. Therefore, our approach allows operators who are non-experts to configure the system without needing to create additional complex technical data.

2.3 Automated Authoring Systems

Automated authoring systems encompass the ability to generate task-guiding visuals with minimal or no input from the user, even for some tasks and environments that the system has never seen before. Requiring minimal or no input makes these authoring solutions completely usable and easily configurable by non-experts, which is a major goal for AR task guidance systems. However, current systems with automated authoring solutions can only generate visuals for a subset of task types or objects, limiting the viability of these systems on tasks or environments outside of their subsets.

InstruMentAR is a system designed to streamline the authoring setup for providing task guidance for instruments with common elements, such as buttons, knobs, sliders, switches, and touchscreens [17]. This system allows an operator to go through an authoring mode where they record each action to perform to use the instrument. By wearing a pressure-sensing finger attachment, the system uses a decision-tree algorithm to determine which action the operator is conducting and uses the output to determine which arrow visual to place. For example, if the operator twists a knob, an arrow is automatically generated around that knob with the same distance it was twisted by. This allows the operator to set up the system without requiring technical AR experience. Their approach is reusable for any instrument with generic elements and simply requires the operator to perform the tasks for automatic authoring, which is usable by any non-expert. Then, the generated guidance can be followed by a user in a tutorial or access mode, providing feedback on whether the user is performing the task correctly or not, advancing automatically if the task is completed. InstruMentAR has a usable authoring solution by non-experts that does not require complex models but is limited in that it can only be used to generate guidance for the instrument components it was designed for, and only with the finger gestures the decision tree algorithm was tuned for.

Stanescu et al. developed a task guidance system for assembly and disassembly tasks of rigid components [25]. Their system has a recording phase where the operator performs the assembly/disassembly task, and the system records volumetric changes in the workspace. For an example of using a stool assembly, placing a leg of the stool results in a volume change in the workspace that the system records and saves. Then, the order of these volume changes forms a directed graph of assembly steps. Additionally, the recorded volume change, such as the volume that represents the stool leg, is used as a visual to guide the user to place the correct component. The volume is also used to detect whether the user places the cor-

rect component during that step and allows the system to provide feedback. Overall, their approach is model-free since it uses volumetric changes to generate guidance visuals instead of a pre-made 3D model, allowing it to work on assembly/disassembly tasks with new rigid objects. The main limitation of this approach is that it only works for assembly/disassembly tasks and requires the object of interest to have large enough sub-components to have detectable volume changes.

Automated authoring solutions overcome the limitation of requiring system experts to set up the system or create any required complex data, such as CAD models, when presented with new tasks. However, current systems are unable to attempt to provide guidance visuals for general tasks and environments since they are limited to the types of tasks or objects they are built to understand.

Our proposed approach addresses the limitations of current automated authoring systems by using vision-language ML models that are trained on general data, allowing the approach to interpret general tasks and environments.

2.4 Enabling Automated General-Purpose Task Guidance Systems

To summarize, the primary limitations of current authoring methods include:

- **Limitation 1:** Requiring experts in AR technology to place visuals.
- **Limitation 2:** Requiring 3D CAD models for real-world objects of interest.
- **Limitation 3:** Only functioning for limited types of tasks or environments.

To enable an automated general-purpose task guidance system following Components 1-6 that is not limited by these factors, the system needs to be usable by non-experts and capable of understanding general tasks and environments without requiring complex new CAD models. An enabling technology for such a system is the use of current vision-language ML models.

Some current AR systems have demonstrated using VLMs to make decisions in general contexts that were previously challenging. While we found none using VLMs for AR task guidance, one system developed in 2024 called OCTO+ determines where to place objects naturally in an AR scene [24]. Given text input, such as "cupcake", and an image, such as a scene with a table containing a plate, the system outputs a 3D location to place the cupcake naturally, such as outputting the location of the plate in the scene. This system uses a pipeline of ML models to accomplish this task. First, it uses a model that outputs a list of text objects present in an image, which are sent to a bounding box detection model to filter the objects based on the size of the object, then a large language model receives a list of the filtered objects and determines which could be used to place the input object, and lastly a segmentation model is used to obtain the center of the object. Any open-vocabulary text description of an object can be input to this system, and it will attempt to output where to place the object in the scene. This system demonstrates the usability of modern VLMs for AR applications.

One such vision-language model is OpenAI's GPT-4V, which was trained on multitudes of text data and images from the Internet and other sources and can take a mixture of text and images as input and output a text response [18]. The model was trained to "predict the next token in a document" and was fine-tuned using reinforcement learning with human feedback to "produce responses better aligned with the user's intent" [18]. This "Foundation Model", meaning that it was trained on lots of general data and is capable of doing several diverse

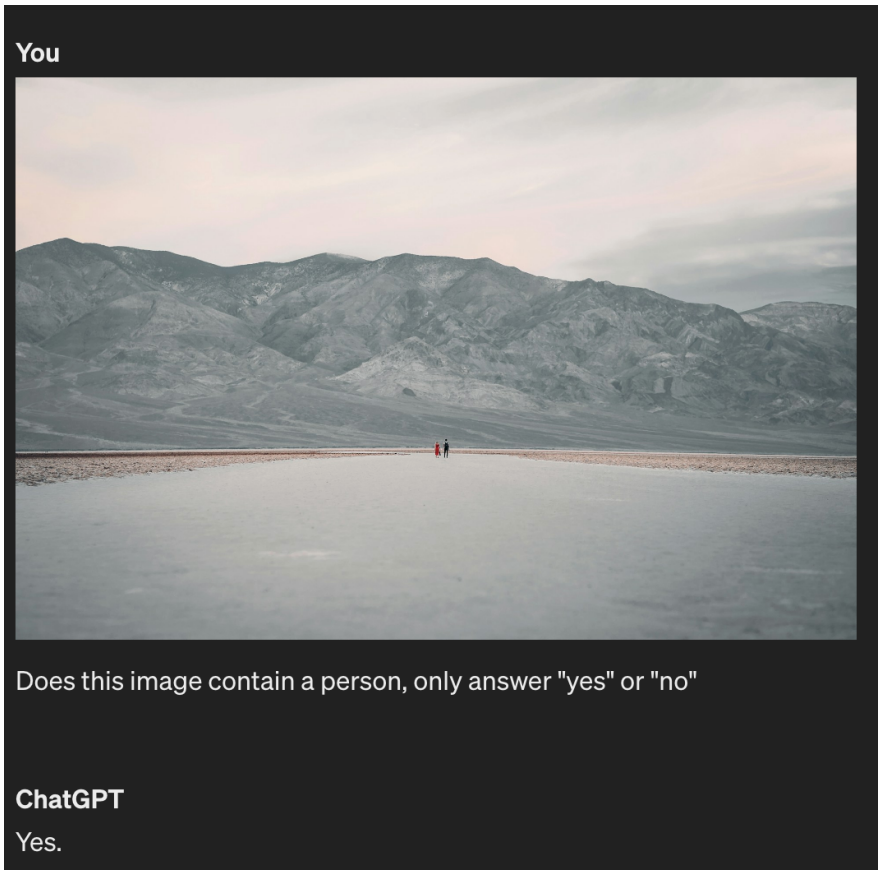


Figure 2.1: GPT-4V input example using OpenAI's ChatGPT interface [2].

tasks, can take a text prompt and an image and will output according to the input prompt it received, referencing the image for any needed information. For example, inputting the prompt "Does this image contain a person, only answer yes or no" along with an image to GPT-4V would typically produce either the output "yes" if a person exists in the image or "no" if not, as shown in Figure 2.1.

Another vision-language model with the specific task of bounding box/object detection is called GroundingDINO, which was trained on several large object detection datasets [16]. GroundingDINO was trained to "detect arbitrary objects given texts as queries" and outputs multiple bounding boxes locating objects from the text prompt in the provided image, as shown by the examples in Figure 2.2 [16]. The model uses several novel architecture choices,

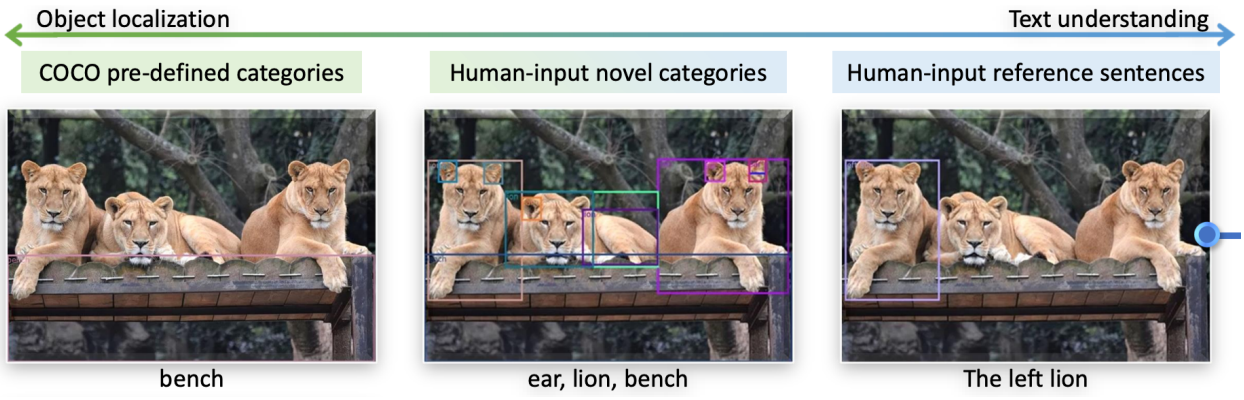


Figure 2.2: GroundingDINO input examples from [16]. Colored bounding boxes are displayed in each image to detect objects from the text prompt below each image.

such as improved text and image fusion blocks, to increase open-set object detection accuracy. It was also trained on referring expression comprehension tasks, which ask for a specific object, such as the "left lion" in Figure 2.2.

To display GPT-4V and GroundingDINO's usefulness in AR applications, The described OCTO+ system used GPT-4 as the LLM to determine which object could be used to place the input object. GPT-4V is similar to the LLM version GPT-4 but also accepts images. OCTO+ also used GroundingDINO as the bounding box detection model. They used these ML models to enable their system to accept open-vocabulary text input for general object names and found that their approach output a natural location with an accuracy of 70%. Their evaluation method consisted of placing 15 common indoor objects in indoor scenes from a dataset with depth data that the system had never seen before, showing the approach's accuracy in general environments. This result shows that their approach significantly outperforms a random placement of these objects with the performance of current ML models. Extending the application of VLMs to an AR task guidance system could also improve the accuracy of generating relevant guidance visuals for general tasks and environments.

Chapter 3

Approach/System Design

The goal of our approach is to generate useful guidance for an end user's tasks. Because this end user requires guidance, it is possible they do not know how to perform the tasks. Thus, our approach accommodates this by having a first user, an operator who is experienced with the tasks, define the end user's tasks and set up the system. Therefore, the system has a different phase for each user. As shown in the operator phase (1) in Figure 1.1, the operator writes text instructions, takes pictures of the objects to manipulate, and tests the system's generated guidance. Once the operator is satisfied with the generated visuals, the system is set up for the given tasks and is ready for the end user, which is shown in diagram (2) in Figure 1.1. Now, whenever the end user wants to be guided through their tasks, they will take matching images of all of the operator images, and the system will generate the guidance visuals for their tasks based on the current environment.

Thus, the operator phase is the authoring method for our approach and sets the system up for the end user's tasks using only text input and RGB images. In the end-user phase, the user simply takes matching pictures of the operator images, which allows them to receive task guidance without requiring knowledge of how to perform the tasks or what the objects do.

Our AR task guidance approach was implemented in a proof-of-concept system on the Microsoft HoloLens 2, which is an AR optical see-through HWD. Some components of the system ran as an application on the HoloLens itself, developed mainly using Unity and Mi-

crosoft MRTK3, while other algorithms ran on a server that could communicate with the AR HWD. For our system, the server was hosted on a personal laptop on the same WiFi network as the HoloLens. No additional hardware other than the HoloLens 2 and personal laptop were used.

3.1 System Assumptions

The following are the assumptions made for our system:

- **Assumption 1:** All objects required for the tasks exist in the environment.
- **Assumption 2:** The system has a predefined library of action visuals.
- **Assumption 3:** Visuals do not update with objects moving in real time.
- **Assumption 4:** Each text instruction outputs only one of the predefined action visuals.

Each of these assumptions contributed to decisions that were made when implementing our system. Assumption 1 is made to restrict the types of tasks that are valid. For example, "pick up the drill" would not be a valid task if the environment does not include a drill. Assumption 2 is made with consideration of Limitation 2 of current systems. To address this limitation of not requiring CAD models for real-world objects, the system either needs to automatically create 3D visuals of objects from the environment or reuse visuals representing generic actions from a predefined library. We chose the latter approach, as discussed later in this section. Assumption 3 is made to loosen the latency requirements of ML model inference, and we show that guidance visuals can still be useful even without updating in

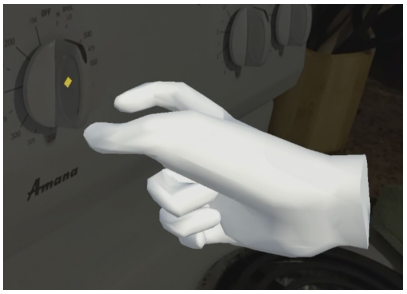
real time. Lastly, Assumption 4 references what type of text input, called an instruction, is valid for our system, which is defined below.

Definition 6. An **instruction** is a text description that requires a single action as part of a procedural task.

Definition 7. In our system, a **valid instruction** is an instruction requiring an action that is contained in the predefined library of action visuals.

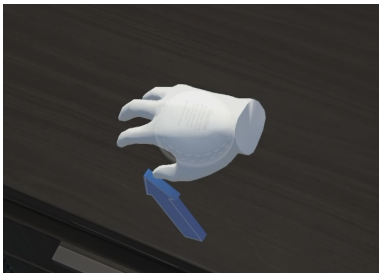
3.2 Predefined Guidance Visuals

(a) Press visual



(b) Twist visual

(c) Pull visual



(d) Pickup/Place visual start

(e) Pickup/Place visual

(f) Place/Place visual end

Figure 3.1: Predefined Action Visuals for Our System

As specified in Assumption 2, our system expects a predefined library of action visuals. One of these action visuals will be the output of the system for each valid instruction. These can include any 3D visuals that are generic and reusable for several types of tasks and their

instructions. For our system, we chose to implement the following action visuals: “press”, “pull”, “twist”, and “pickup/place”. We chose to create each action visual as a 3D hand animated such that it mimics the action as a person’s hand would, as shown in Figure 3.1.

The set of actions chosen, including press, pull, twist, and pickup/place, are generic actions that can accommodate a large range of tasks, sufficient for our proof-of-concept system. For example, “press” can be used to guide the user to press any button, which is common for tasks ranging from using kitchen and household appliances to manufacturing machines. “Pull” is a common action that can be used for tasks ranging from opening a drawer or door containing tools to pulling a lever. “Twist” is another common action for tasks ranging from opening a lid to twisting a knob on machines or appliances. Lastly, “pickup/place” is a common action that can be used to guide a user to move an object to another location.

To create the library of predefined guidance visuals, such as our action hand visuals, a system expert will have to initially create these 3D animations. The library of visuals should contain all possible actions for the end user’s tasks. For example, if the end user is expected to do the task “Add vegetable broth to the pot”, an action visual for “pour” would need to be created. Thus, once a system expert creates all action visuals for the expected set of end-user tasks, the system can reuse these visuals to generate task guidance, and the system expert would not be required anymore. Ideally, this approach allows such systems to include a large number of action visuals, encompassing all types of actions required.

3.3 Operator Phase

Our authoring and operator phase is shown in Figure 3.2. The Operator Phase consists of the entire diagram. Referencing the top diagram (1), the operator first defines what tasks the end user should receive guidance for, and they write valid instructions as text input to the

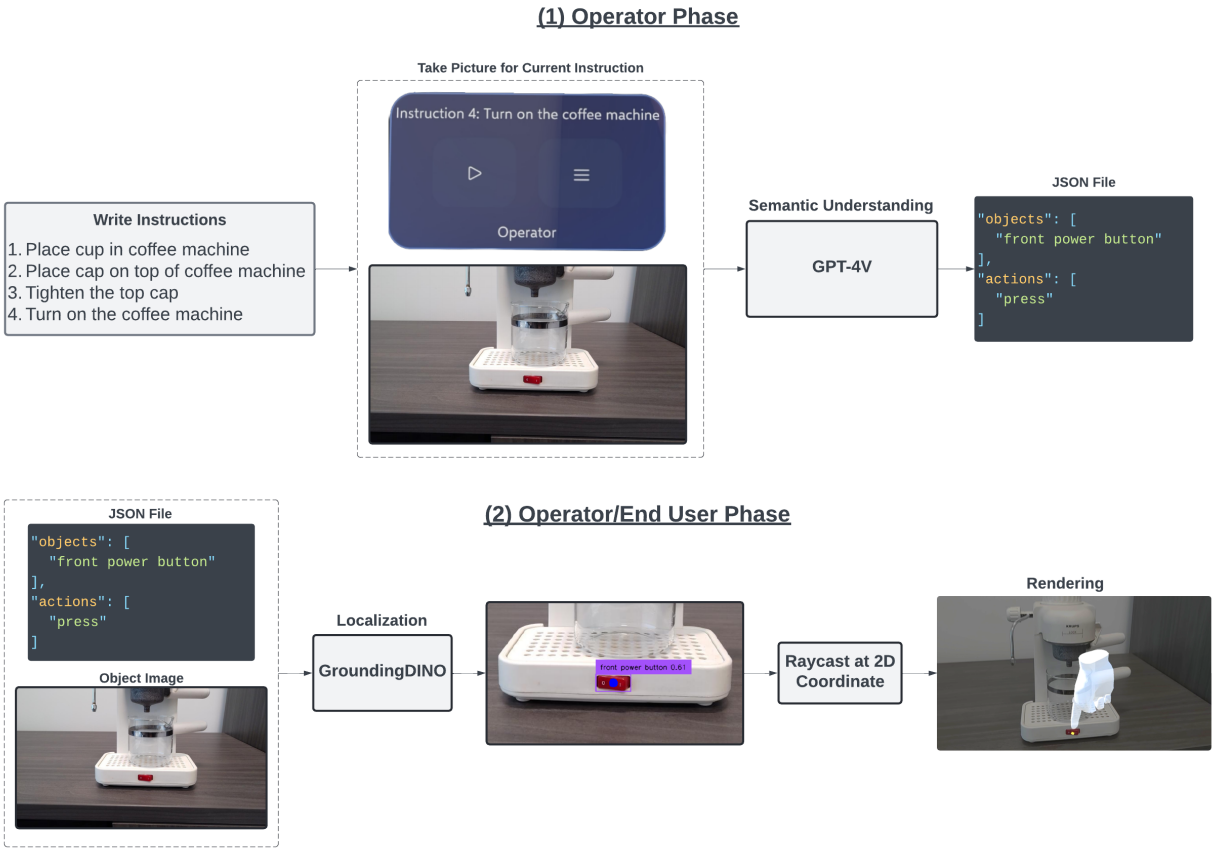


Figure 3.2: System Design for Operator Phase

system. Then, they take pictures of the relevant objects for each of their instructions. For each instruction, the image and instruction text are input to the Semantic Understanding Block, which uses the vision-language model GPT-4V to parse the inputs. The Semantic Understanding Block outputs a description of the relevant object, as well as the action, for the instruction in a JSON file.

Referencing the bottom diagram (2) in Figure 3.2, the Localization Block uses the object description as a text prompt for object detection using the bounding box detection VLM GroundingDINO. Using the output 2D coordinate of the relevant object location in the image, the system determines the 3D coordinate by raycasting and finding the intersection

with the 3D spatial map tracked by the HoloLens 2. Lastly, the Rendering Block displays the visual stored in the JSON file at the intersected location on the 3D spatial map. Diagram (2) in Figure 3.2 is the same sequence of components used by the End User Phase, explained in Section 3.4.

3.3.1 Operator Instructions and Pictures

The inputs from the operator during this phase are text instructions and images. The operator begins by writing the text instructions in a text file, which are used by the rest of the system. As stated in Assumption 4, a valid instruction should be written such that it expects the user to perform one action from the predefined action library. The operator can write this instruction using any wording they want, as there is no required structure. From empirical testing, the operator can typically write the instruction as they would describe it to a person. However, sometimes instructions require more detail of the objects, similar to being more specific to a person. One example valid instruction for our system from Figure 3.2 is "turn on the coffee machine", which expects the end user to press a button.

The other input from the operator is pictures taken of the objects for each instruction. On the HoloLens app, the operator can go through each instruction and take these pictures using the front-facing PhotoVideo (PV) camera. Our system uses a voice command "operator image" to take an image of what the user currently sees without having to click any buttons. Using the "turn on the coffee machine" example, the operator would take a picture of the coffee machine, focusing on the button the user is expected to press. From empirical testing, images are typically best if the operator is as close to the relevant object as possible while ensuring it is completely in view and not blurry. It is also helpful to take images straight-on with the relevant object instead of from odd angles. At the instant the image is taken, the

HoloLens app saves a snapshot of the pose and camera properties from the PV camera, which is used later in the Rendering Block described in Section 3.3.4.

3.3.2 Semantic Understanding

GPT-4V System Prompt

"You will be given multiple instructions that a user has to perform. You will be given them one at a time as the user completes them. Your output should be in JSON format. One JSON field should be called 'objects', which will contain a list of objects (strings) that exist in the provided image that the user should use to complete the current instruction. The object within the 'objects' list should have a corresponding action the user should perform on the object to complete the instruction. Another JSON field called 'actions' will contain these actions (strings) in a list where each entry is the action the user should perform on the object in the 'objects' list. The possible actions for the user include: press, twist, pull, pick up, place the picked up object at this location. Output one of these actions exactly as listed. Output only one object and action in each list that would be best to complete the instruction based on the image. Ensure the object you output exists in the current provided image. In the 'objects' list, each object string should be structured as such: '<object position> <object name>' with the following properties: <object name> is the specific object the user should use to complete the instruction. If the object is a component within a larger object, output the most specific component needed for the instruction. Be specific while trying to use the lowest amount of words for the object name. <object position> is the position of the object in the image. Check that the position ensures the object can't be confused with other similar objects, also include the object color if it is unique. Be specific while trying to use the lowest amount of words for the position. Do not include any other JSON fields other than 'objects' and 'actions', which should both be lists of the same length. Make sure the outputs make sense given previous instructions the user has completed."

Figure 3.3: GPT-4V System Prompt

For each instruction, the text instruction written and images taken by the operator are parsed by a VLM that can understand the semantics between the natural language instruction and images. GPT-4V is the VLM used and is instructed with the system prompt shown in Figure 3.3. We use GPT-4V through OpenAI's API called from their Python *openai* package from our server [4]. Due to this prompt, giving GPT-4V an instruction and image of the relevant object results in it outputting an object description and action. This object description output will be used as the object detection prompt in Localization, so ensuring GPT-4V outputs a reliable and accurate object description is crucial. This prompt was developed over many iterations of empirical testing. Having GPT-4V output the object description and action in a list allows the system to be more extensible in the future.

The system prompt includes the possible action from the predefined visuals library as input: "The possible actions for the user include: press, twist, pull, pick up, place the picked up object at this location." The list of possible actions in this system prompt would look different for systems with a different set of predefined action visuals. GPT-4V is instructed to output the object description in a format such that the object position precedes the object name to help the object detection model use positional information. Parts of the prompt such as this are driven by empirical testing alongside the object detection VLM, GroundingDINO. In the example displayed in Figure 3.2, the instruction "turn on the coffee machine" and the image shown are input to GPT-4V, which outputs the object description "front power button" and action "press".

While the prompt shown in Figure 3.3 seems verbose, we found that the object detection model, GroundingDINO, has higher accuracy with specific types of prompts, which include a short position of the object, as well as a short object description. We found this more precise and wordy GPT-4V prompt to output object prompts that produced a higher accuracy for GroundingDINO than earlier iterations, such as the version shown in Figure 3.4.

Earlier Iteration of GPT-4V System Prompt

"You will be given an instruction that a user has to perform. Your output should be in JSON format. One JSON field should be called 'objects', which will contain a list of objects (strings) that exist in the provided image that the user should use to complete the instruction. Each object string should be structured as such: '<object position> <object name>', and should be understandable by a fifth grader. <object position> is information to help a fifth grader find the object in the image without ambiguity. If there are multiple of the same object, give a relative position of this object to the others. <object name> is the object the user should use to complete the instruction, which should be understandable by a fifth grader. Additionally, the user will have to interact with one of the objects to perform the instruction. The possible actions for the user include: press, pull, twist, pick up, place the picked up object at this location. So, another JSON field should be called 'action' which will include only one of the actions that the user should perform. Do not include any other JSON fields other than 'objects' and 'action'"

Figure 3.4: Earlier Iteration of GPT-4V System Prompt

From our iterations of the GPT-4V system prompt we found these changes to positively affect our system's accuracy:

1. **Giving GPT-4V the previous conversation context:** Initially, we did not include any previous conversation to subsequent GPT-4V API calls. However, we found that giving all past instructions and responses (not images) allowed GPT-4V to ensure its next response would make sense given. For example, an instruction "place the lid on top of the coffee machine" was provided to GPT-4V with separate images of the lid and coffee machine. Without context, GPT-4V would not know that the lid was already picked up and would typically output the repeated instruction. With context, GPT-4V could see the lid was picked up and could focus on placing it on the coffee machine.

2. **Shorter and more specific object descriptions:** The final system prompt specifies the object position and object name to "try to use the lowest amount of words". Additionally, system is instructed to output the most specific object necessary, and include the color of the object if it is unique. These portions of the prompt were found based on empirical testing with GroundingDINO, which can get confused with very long text prompts, and is well-suited to find specific colors.

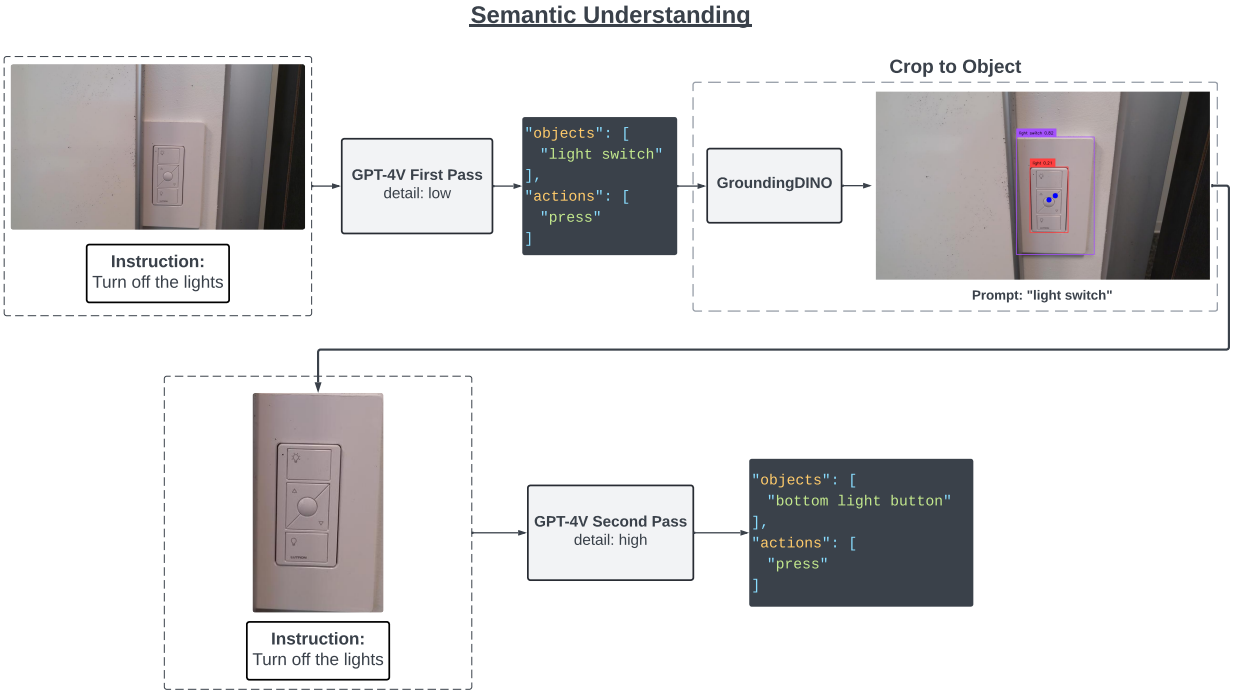


Figure 3.5: Block Diagram for Semantic Understanding

Other considerations that affect the reliability and accuracy of the object description and action output by GPT-4V are the resolution and region of the image given as input. For each image taken by the operator on the HoloLens, our system calls GPT-4V with two passes, as shown in Figure 3.5. The first pass inputs the image with low detail, which uses a 512x512 pixel version of the image, according to the GPT-4V API, to obtain GPT-4V's first output [4]. Then, we use this first output to prompt GroundingDINO for bounding

box detection. If there is only one bounding box, then the first pass is used as the final GPT-4V output. Otherwise, in a step we call "Crop to Object", a cropped version of the original image is obtained by cropping to the smallest region of the image containing all of the output bounding boxes. Thus, the second pass inputs this cropped image with high detail, which ensures the image is resized with a higher resolution, according to the GPT-4V API, to obtain GPT-4V's final output [4].

From empirical testing, we found that GPT-4V would output more precise and expected object descriptions from an image cropped or zoomed into the relevant object, essentially cutting out the background. Hence, the second pass to GPT-4V using the Crop to Object step was included. Further examples of Crop to Object are explained in Section 3.3.3. This step was one part of enabling our system to display guidance visuals at specific parts of objects.

In some cases, GPT-4V will output an invalid response, which either means that its response was not in a valid JSON format, or the action it outputs was not one of the valid actions. In these cases, we allow GPT-4V to run three total attempts, which typically corrects itself as long as there is an object relevant to the instruction present in the image. From our testing, GPT-4V typically does not give a valid response when the image does not include any object that is relevant to the instruction, which is useful since GPT-4V essentially enforces an image-level version of Assumption 1, ensuring that the image taken contains the object required for the instruction.

The output of the Semantic Understanding Block is stored in a JSON file in our system, which stores the final object descriptions and actions for each instruction. The system is expected to have one object description and action for instructions that require "press", "pull", and "twist" actions. Instructions that require pickup/place actions are expected to have two object descriptions and actions. One is a description of the object to pick up with

the action “pick up”, and the other is a description of the location where the picked up object should be placed, along with the action “place the picked up object at this location”, which was developed over multiple empirical tests of GPT-4V’s output. GPT-4V is instructed to output one of these actions as such. For pickup/place instructions, the system expects GPT-4V to output a pickup object and a location to place the object, but sometimes GPT-4V will not include both actions in its output. Thus, the system ensures that pickup/place instructions include both actions with a post-verification check.

3.3.3 Localization Block

The Localization Block’s goal is to take the object descriptions for each instruction output by GPT-4V and precisely locate the 2D coordinates of those objects in the image taken by the operator. GroundingDINO is a VLM that finds bounding boxes for a given text prompt above a certain confidence. Thus, the object descriptions for each instruction stored in the JSON file are used by the Localization Block as the text prompt to GroundingDINO, as shown in Figure 3.2. We run GroundingDINO locally on our server by downloading the weights to the GroundingDINO Swin-B version of their model and using their Python library [3]. We tested using both the Swin-T and Swin-B versions of the model available on their GitHub and empirically found that the Swin-B version outputs more reliable and accurate detections for referring expression tasks that require the model to understand positional information such as finding the “bottom button”.

Figure 3.6 visualizes the block diagram for the Localization Block. To increase the reliability and accuracy of detecting the correct specific object in an image, we use a similar two-pass structure with the same Crop to Object step used in the System Understanding Block. In the top diagram (1), Crop to Object crops the input image to the smallest region containing all

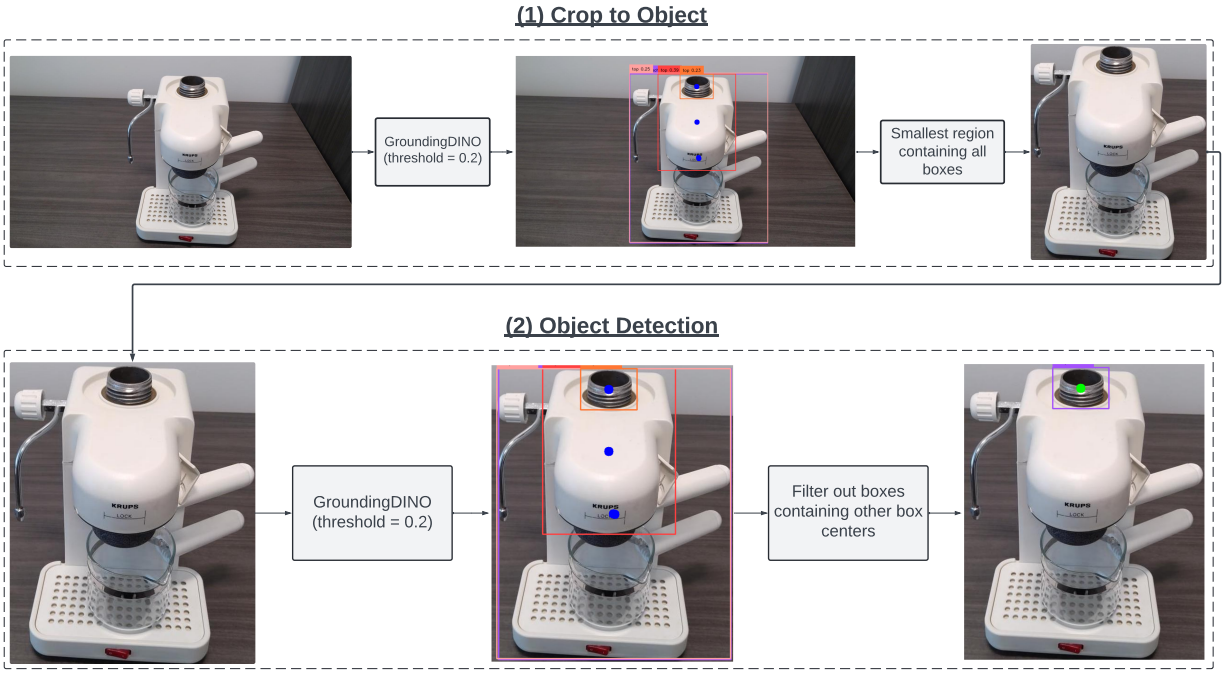


Figure 3.6: Block Diagram for Localization with prompt "top screw hole of coffee machine".

of the bounding boxes. GroundingDINO is prompted in the first pass to find the "top screw hole of coffee machine" and outputs several bounding boxes. Using this first output, the wrong object location would have been selected since the bounding box locating the screw hole has a lower confidence value than other bounding boxes. To increase the accuracy, we crop the image to the smallest region containing all of the bounding boxes, which crops to the coffee machine in this example. Now, the second pass prompts GroundingDINO again for "top screw hole of coffee machine" but using the cropped image. To further increase the accuracy required from object detection, the bounding boxes are filtered by removing boxes containing other box centers, essentially keeping the most specific box found on an object. Then, the resulting box with the highest confidence is output. This is represented by the bounding box with the green dot at the end of Figure 3.6.

Figure 3.7 shows another example of Localization for the prompt "white coffee maker lid".

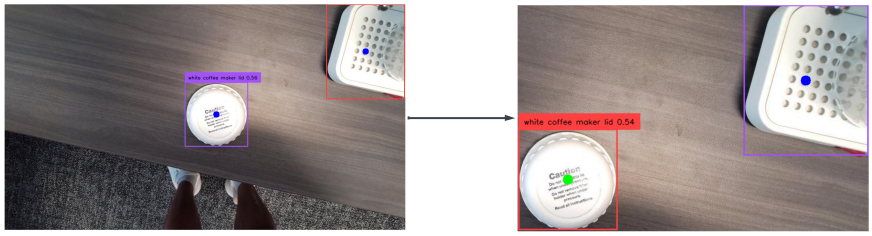


Figure 3.7: Another example of outputs from the first and second pass of GroundingDINO in Localization Block with prompt "white coffee maker lid".

In this example, the first-pass bounding boxes are found in the center and top-right of the input image. After cropping to the smallest region containing all of the bounding boxes, the image is zoomed in to the relevant region of the image. Then, the second pass of GroundingDINO results in the correct object having the highest confidence. As mentioned, GroundingDINO requires a threshold for the minimum confidence of bounding boxes to output. The threshold we used for all GroundingDINO inferences was 0.2. This means that all boxes with a detection confidence higher than 20% are output, which we empirically found to increase the output's accuracy by allowing us to filter through several boxes through the steps in Figure 3.6, rather than narrowing down possibilities with a higher threshold.

After both passes of object detection shown in Figure 3.6 are complete, a final bounding box will be found. The output of the Localization Block is the 2D image coordinate corresponding to the center of the resulting bounding box, which is sent from the server to the AR application upon completion.

3.3.4 Rendering Block

Using the 2D coordinate of the object from the Localization Block and the HoloLens 2 PV camera's pose saved from the instant the operator took the picture, the Rendering Block determines where to place the visual in 3D space, as shown in Figure 3.2. To determine the 3D coordinate from 2D images, we use the spatial mapping capabilities of the HoloLens 2 to maintain a 3D map of the environment around the headset using the Unity AR Foundation package [1].

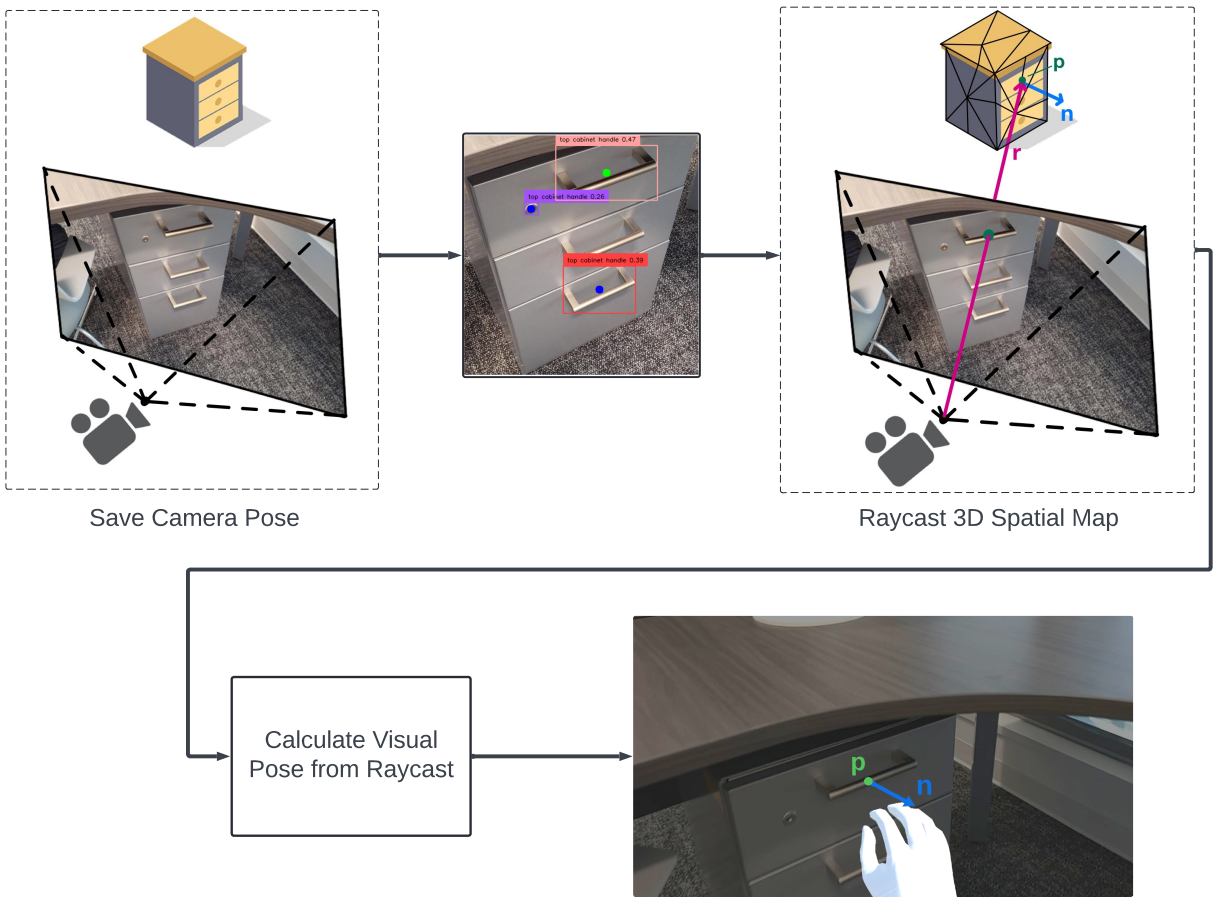


Figure 3.8: Block Diagram for Rendering

The components of the Rendering Block are visualized in Figure 3.8. First, when the operator

takes a picture of an object for an instruction, the system saves the pose of the HoloLens 2 PV camera (the external, front-facing camera that takes the picture) at the instant the image is captured. Then, after the Localization Block completes and sends the 2D coordinate of the object, the system creates a raycast from the PV camera origin through the location corresponding to the 2D coordinate on the image plane. In the example shown in Figure 3.8, the green dot is the output 2D coordinate shown in the bounding box detection image. This raycast then intersects with some 3D location on the 3D spatial map, which is the 3D location of the object detected in the 2D image from the Localization Block. This results in an intersection location (point $\vec{p}$, which is a vector in world coordinates), and a normal vector of the intersected surface (vector $\vec{n}$) shown in Figure 3.8.

From the raycast intersection location and normal, the hand visual is placed based on the type of action. The action visual to display for each instruction is obtained from the JSON file output by the Semantic Understanding Block, which was determined by GPT-4V. For the press, pull, and twist actions, the visual is placed in front of the surface at a certain offset based on the action, which is calculated along its normal by the point $\vec{p} + \mathit{offset} * \vec{n}$ using the notation from the raycast step in Figure 3.8. The hand rotation for these visuals is such that the visual is facing the surface, so its forward vector is aligned with the vector $-\vec{n}$. Each of these visuals has a looping animation to mimic its action. The press and pull animations move along the normal vector, while the twist animation maintains its position but uses the normal vector as its rotation axis.

For the pickup/place action, the 3D locations of both the object to pick up and the location to place it need to be obtained. Then, the pickup/place animation starts at the raycast intersection location of the object to pick up and ends at the raycast intersection location of where to place the object. The pickup/place animation also loops and is calculated using spherical interpolation, such that the animation is always moving in an upward arc from the

start to end location. Images of these visuals are shown in Figure 3.2.

3.4 End User Phase

3.4.1 Scanning Objects

The end user is the individual who is performing the tasks that our system is providing guidance for. The design of the Scanning Step in the End User Phase, shown in Figure 3.9, is largely shared with the bottom diagram (2) of the Operator Phase in Figure 3.2. The main difference is that in the End User Phase, the end user is shown each image taken by the operator in the Operator Phase, and they are instructed to take as close of a matching picture as possible of the object.

This design was chosen because the goal of the End User Phase is to display guidance for the tasks set up by the operator. Because the end user needs guidance for their tasks, they cannot be expected to figure out which objects are required for each instruction. Therefore, the end user simply has to take matching pictures of the same objects displayed to them on a hand menu (shown in Figure 3.10a), without requiring any knowledge of the tasks. The End User Phase has a voice command, "scan image", which will take a picture and is set up for this scanning step. This allows the end user to take pictures hands-free once their body and head have positioned the camera. Our system allows the end user to retake the picture until they confirm it using the dialog box shown in Figure 3.10b.

The purpose of the Operator Phase is to set up the system so the End User Phase can generate guidance for the end user's tasks and environment. This means that setup data from the Operator Phase needs to persist over time whenever the end user runs the application. The two main components that achieve this are (1) the storage of each instruction's object

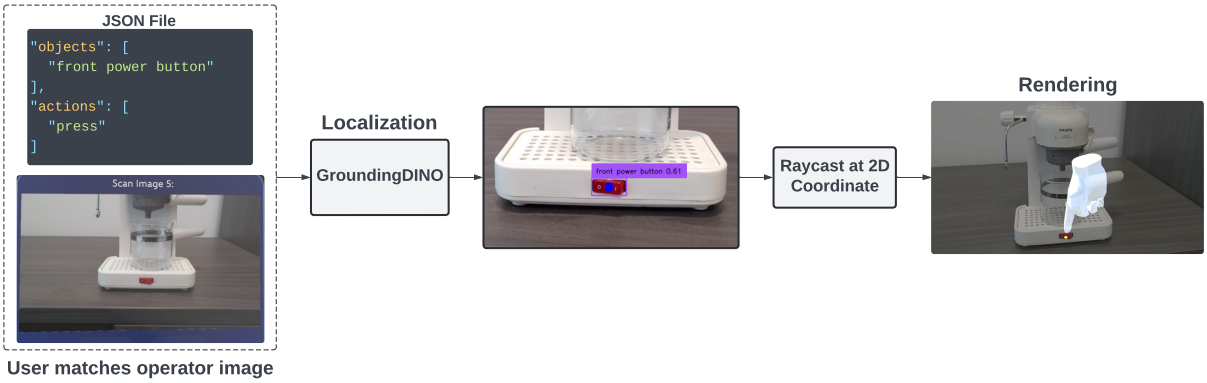


Figure 3.9: System Design for Scanning Step of End User Phase

(a) Hand menu displaying image to scan

(b) Dialog box to confirm picture taken

Figure 3.10: Menus for End User During Scanning Step

descriptions and actions in a persistent JSON file on the server, and (2) the storage of all the images taken by the operator on the HoloLens 2. This allows the End User Phase to retrieve these outputs from the Operator Phase so that the operator does not need to reconfigure the system unless the tasks change or relevant objects in the environment are replaced. The End User Phase does not affect or change the operator setup, such as the JSON file or operator images.

The design decision to have the end user scan (take a picture of) each image even after the

operator has already done so in the Operator Phase is because our system is designed such that it can be used by the end user any time after the operator sets it up for these tasks with the objects in the current environment. Therefore, it is likely in many use cases that object locations in the environment have changed since the Operator Phase. It is also possible that the user is not in the same environment as the operator. For example, an operator may set up guidance on how to assemble a computer from a company facility, which can be used by an end user assembling the same computer from their house. Thus, having the end user scan each image upon startup of our application ensures that the guidance is generated at the correct 3D location based on the changed object locations or environment.

For each matching picture taken by the end user, the Scanning Step begins by inputting the image and object description generated by the Operator Phase stored in the JSON file to the same Localization Block detailed in Section 3.3.3, running on the server. In the example shown in Figure 3.9, the object description "front power button" is retrieved from the JSON file configured in the Operator Phase. Then, the output of the Localization Block is sent back to the AR application for the Rendering Block, which is explained in Section 3.3.4. The type of action visual to display is chosen by the action output by the Operator Phase stored in the JSON file. The Rendering Block determines the placement of the action animation once all of the images for an instruction have been taken and localized.

3.4.2 Using Generated Task Guidance

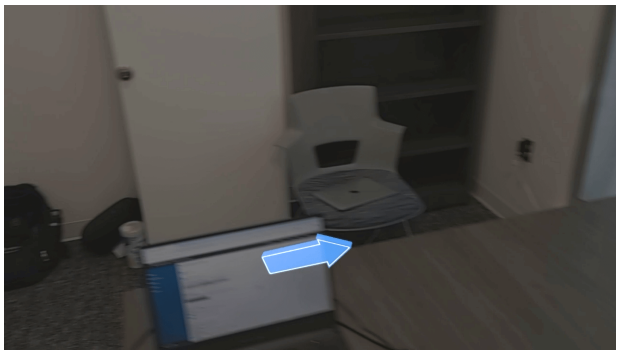
Once all images have been scanned by the end user and all visuals have been rendered from the Scanning Step in Figure 3.9, the system has generated all guidance visuals for the end user. Now, the user can advance through each instruction while receiving guidance from the generated action visual. A dynamic 3D arrow is positioned in front of the user that points

(a) Guidance for “tighten the top white cap”

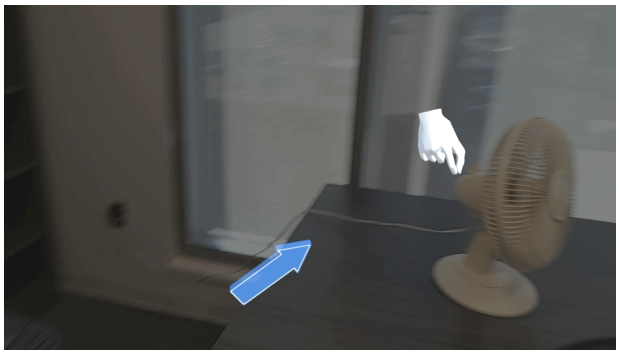
(b) “turn off the lights with bottom button”

(c) Hand menu to view current instruction text

(d) Guidance for “open the top cabinet”



(e) 3D Arrow guiding user to visual to the right



(f) 3D Arrow guiding user to visual within view

Figure 3.11: Using AR Task Guidance Generated by Our System

to the location of the generated hand visual in space to guide the user to the visual when they cannot find it, as shown in Figure 3.11 images (e) and (f). This 3D arrow is hidden while the user is within a one-meter distance of the hand visual, and the visual is within their field of view. Additionally, the end user has a hand menu with the current instruction text displayed that they can reference at any point, as shown in Image (c) in the figure. The

user can advance to the next instruction by using the voice instruction "next instruction", or by clicking a button labeled with each instruction number located on the hand menu. Examples of visuals generated by our system are shown in images (a), (b), and (d) in Figure 3.11.

3.5 General-Purpose Task Guidance System Design

In this section, we discuss to what extent our system implements the components that should be present for a General-Purpose AR Task Guidance system as listed in the Problem Statement (Section 1.3). Our approach and proof-of-concept system were developed with these components in mind to answer our first research question RQ1. The advantages of such a system are that it addresses and improves upon Limitations 1-3 of current AR task guidance systems.

Components 1 and 2 - General Tasks and Environments

To generate guidance visuals from our system, an operator determines the end user's tasks and writes valid text instructions for each action necessary to complete those tasks. These tasks are not known to the system beforehand and can be written using free-form natural language. Thus, Component 1 is present in our system. For our system, the tasks must require actions from the set of actions in the predefined visual library, which could extend to as many actions as necessary for the possible tasks. Moreover, our system uses pictures taken by the operator and end user, as well as a 3D spatial map of the environment, which does not require any special preparation. Thus, Component 2 is present in our system.

Components 3 and 4 - Automatically Determining Where to Place Visuals without Complex Technical Input

Once a system expert creates the reusable library of predefined action visuals, our system does not require CAD models of physical objects in the environment, lengthy data collection and training of a custom ML model, or any other complex technical input. Thus, Component 3 is present in our system. Through our Operator Phase and authoring approach of utilizing VLMs that can understand the semantics of images and text (GPT-4V), can localize objects (GroundingDINO), and an approach that can convert from a 2D image coordinate to a 3D world coordinate, our system can determine which visuals to place from the predefined visual library and where to place them (Component 4).

Components 5 and 6 - Clear Task Guidance Robust to Object Location Changes

Our End User Phase is designed such that upon running the application, they take matching pictures of the objects in each operator image, which allows the system to localize the objects even if they have moved locations in the environment. Thus, our system is robust to object location changes since the end user can still receive guidance for their tasks, as Component 5 states. Lastly, the goal of an AR task guidance system is to provide clear guidance to an end user inexperienced with their tasks, as stated by Component 6. While our selected guidance visuals were not justified using a research method, our proof-of-concept system was developed with clear guidance in mind by providing helpful hand visuals and a 3D arrow. The presence of this final component is explored by the results from our user study in Chapter 4.1.

Chapter 4

User Study Design, Results, and Discussion

4.1 Study Design

We conducted a user study where participants used our proof-of-concept AR task guidance system to test its accuracy as a general-purpose system, as well as its usability by non-experts. One goal of this user study was to validate that our proof-of-concept system follows our general-purpose AR task guidance approach, outlined by Components 1-6 in Section 1.3. Additionally, the study was conducted to determine the accuracy of such a system in generating clear guidance output visuals during the operator phase, and the accuracy of end users completing the expected tasks during the end-user phase.

We also recruited participants with no prior experience with the system, as another goal is to validate that our proposed approach is usable by non-experts. Conducting a user study with these goals allows us to collect data relevant to answering our research questions RQ1 and RQ2 in Section 1.4.

4.1.1 Apparatus

The study was conducted in a typical office room using various objects either placed in or part of the room: a fan, alarm clock, coffee maker, drawer cabinet with three vertically stacked drawers, and light switch. The office room is shown in Figure 4.1. Each object used for the study is shown in Figure 4.2.

Figure 4.1: Picture of the office used for the user study.

Participants used our proof-of-concept system developed on the Microsoft HoloLens 2, using hand tracking and voice commands for system input. A laptop (with NVIDIA GeForce GTX 1660 Ti GPU) was used as the server running the system's backend software, including the ML models.

(a) Top of alarm clock

(b) Drawer cabinet

(c) Top cap for coffee machine

(d) Coffee machine

(e) Back panel of fan

(f) Light switch panel

Figure 4.2: Pictures of objects used for user study

4.1.2 Tasks

Each participant conducted one session, using the system as both the end user and the operator in two separate phases. The goal of the end user was to use a system previously set up by an operator, and attempt to complete certain tasks purely based on the generated guidance visuals. When using the system as the operator, the goal was to set up the system

to generate guidance for tasks that would achieve a certain end state of the room from the initial state.

Each participant began in the end-user phase of the study. Since using the system as an end user requires operator setup, they used the configured system from the previous participant for this phase (P1 data was not recorded for this phase). Between each participant, the study conductor changed the location of each object (except for the light switch mounted to the wall). To begin, the participant used the system as the end user, which prompted them to take a matching picture from the HoloLens of each object shown in the previous operator's images.

Once all matching pictures were taken and the guidance set up, the participant was instructed to go through each instruction, which was unknown to them, using the "next instruction" voice command and perform the real-world task that they think the 3D visuals are guiding them to do. For this phase, they could not look at any of the text instructions, so their action was purely based on the generated guidance shown to them. We decided to not let them use the text instruction because users could have simply used the text instruction to complete the tasks in this study without using the guidance visuals. After following all of the instruction visuals, this phase of the study was concluded.

In the next phase of the study, the participant set up the system as the operator by writing text instructions and taking pictures to generate guidance. Between phases, the study conductor changed the location of each object again. The participant was given guidelines for writing instructions and taking pictures:

Instruction Guidelines:

1. Write the instruction as you would describe it to another person.

2. If the output is incorrect, it can help to include details of the object, as you would with a person.

Picture Guidelines:

1. Get as close to the relevant object as possible while ensuring it is fully in view and not blurry.
2. Try to take pictures from straight-on.

They were told that their goal was to get the room into a specified end state from the room's initial state:

Room Initial State:

- Fan off
- alarm clock making noise
- disassembled and turned-off coffee maker
- drawer cabinet with all three drawers closed
- room lights on

Expected Room End State:

- Fan on
- alarm clock noise off
- coffee made
- one of the drawers open (each participant chose top, middle, or bottom)
- room lights off

To achieve this end state, either one or multiple actions had to be performed to accomplish each task using the objects shown in Figure 4.2. Each object was spread around the room, often requiring the user to physically walk around the room to complete the tasks. First, to

turn the fan on, a speed knob on the back of the fan had to be turned clockwise to any of the numbered settings. To turn the alarm clock noise off, a snooze button on top of the alarm clock had to be pressed. To make coffee, the coffee maker lid had to be placed onto the top of the coffee maker and screwed in completely. Then, the power switch of the coffee maker had to be pressed. To open one of the drawers, the user chose either the top/middle/bottom drawer to pull open. To turn off the room lights, the bottom button of the light switch needed to be pressed. By this phase, the user had already followed guidance to do most of these tasks as the end user without knowledge at the time that these were their tasks.

Specifying the end state of the room does not bias the participant on certain language to use when writing their instruction. For example, instructing them to "snooze the alarm clock" may have biased them to use that phrase as their written instruction, compared to the state of "alarm clock noise off" which does not contain an action. Additionally, to instruct them on how to write a valid instruction that includes an action from one of the predefined action visuals for our system, each participant had already seen the 3D visuals during the end-user phase but was also shown a short video of each animation as a reminder. They were told that each of their written instructions should be expected to output only one of the displayed visuals. Thus, these videos did not bias participants towards using specific action terminology when writing their instructions, such as press, pull, twist, and pickup/place. Lastly, the study conductor changed the order to achieve the end states between participants but kept lights off as the final state for all participants.

Using the provided information, the participant then wrote their instructions in a text file given the Instruction Guidelines, where each instruction was entered on a new line in the file. Then, the participant used the HoloLens application to take pictures given the Picture Guidelines. Once complete, they tested the guidance that the system generated. For the guidance output for each instruction, the participant stated whether the output was as they

expected and would be clear for a new person to perform correctly.

For outputs they deemed insufficient, we asked the participant whether the incorrect output was due to the output's location, selected action visual, or both. For the pickup/place action, the order of the visual from start to finish was included in the location category. For incorrect outputs, the study was designed such that the participant could update their instruction wording and/or picture in subsequent trials until they stated that 100% of their instructions had clear outputs. Thus, each participant's operator phase lasted an indefinite number of trials until this condition was achieved.

4.1.3 Measures

During the end-user (first) phase of the study, as each participant followed our system's guidance to perform tasks unknown to them, we manually logged whether they completed the correct action for each instruction before moving on to the next one. Thus, each action was only logged as accurate if they completed the expected action before saying "next instruction". During this phase, the system also logged object detection output images for all pictures taken by the participant during scanning.

During the operator (second) phase of the study, each participant stated whether each instruction output visual would be clear to a new person for them to complete the correct action. For outputs that were not clear, they updated their inputs in a subsequent trial. During each trial, we manually logged their response for each instruction output. Thus, each output was only logged as accurate if the participant stated that the output visual was clear. During each trial, the system also logged all written instructions by the participant, the output JSON file (described in Section 3.3), and object detection output images.

After both phases, participants had a post-study interview, asking them subjective qualita-

tive and quantitative questions, shown below:

- **Q1:** On a scale of 1-7, did the proposed system provide guidance visuals that you expected for your input text instructions? 1 (outputs not expected) - 7 (outputs were expected)
  - If not, how did they differ from your expectations?
- **Q2:** Did the proposed system make it easy to get the output you expected?
  - **Q2a:** On a scale of 1-7, Given the guidelines, how easy was it to write instructions to get an expected output? 1 (very easy) - 7 (very hard)
  - **Q2b:** Given the guidelines, how easy was it to take good pictures to get an expected output? 1 (very easy) - 7 (very hard)
- **Q3a:** On a scale of 1-7, when needing to go back and revise instructions and/or pictures after an output was incorrect, was it easy to know what to change? 1 (very easy) - 7 (very hard)
  - **Q3b:** On a scale of 1-7, did those changes make the output better? 1 (output was not better) - 7 (output was better)
- **Q4:** How could the proposed system and approach be improved?
- **Q5:** What are some example real-world applications for a proposed system such as the one you tried?

We wrote down participant responses to each question during the post-study interview. After each session, we went through each participants' written instructions, JSON files, and object detection output images for each trial from the operator phase to determine the cause of any incorrect outputs. We did the same for the object detection output images from the

end-user phase.

4.1.4 Participants

We had 12 participants (75% male, 25% female) between ages 18 and 23, all being undergraduate or graduate students (referred to as P1-P12). None of these participants had prior experience with our system, and only two of them had used AR more than three times.

4.1.5 Procedure

First, we obtained Institutional Review Board (IRB) approval to conduct a user study. During participant recruiting, we sent emails to prospective participants containing a consent form to inform them about the study and their rights before consenting. Once a participant arrived for their study session, we began with a pre-study questionnaire conducted on the QuestionPro software with background questions, such as their age, gender, and experience with AR. We also ensured to wipe down the HoloLens 2 before participant use.

Then, the participant began the end-user phase, followed by the operator phase. During these phases, we instructed the participants and they conducted their tasks as outlined in Section 4.1.2. Once participants were done using the system during these phases, we conducted a post-study interview, verbally asking them questions shown in Section 4.1.3.

The email used for recruitment, consent form, pre-study questionnaire questions, and the post-study questionnaire document are all shown in Appendix A. Each participant's session lasted between 60-90 minutes.

4.2 Results

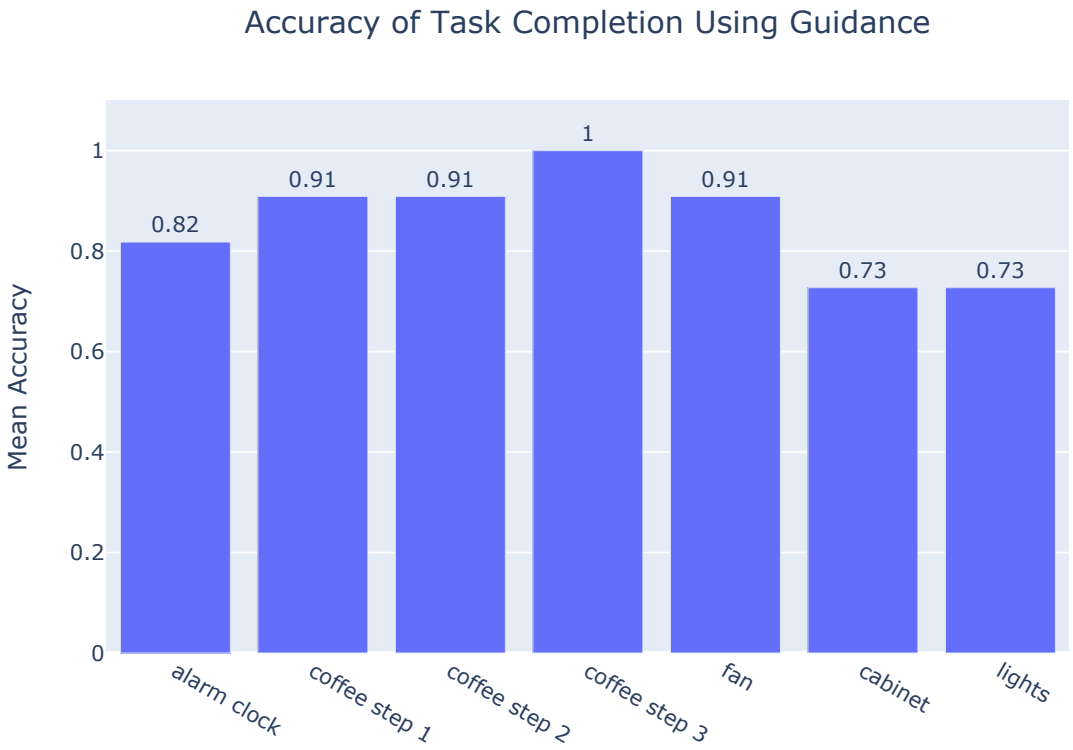


Figure 4.3: Mean accuracy of participants in the end-user phase completing the correct action from the previous operator's instructions

Beginning with the end user phase of the study, which each participant completed first, the accuracy that each participant completed the correct real-world action purely from our system's generated guidance is shown in Figure 4.3. Each bar represents the accuracy averaged over all participants of completing the correct action for the object shown on the x-axis. Coffee steps 1, 2, and 3 refer to placing the lid on top of the coffee machine, screwing the lid in, and pressing the coffee machine "on" button, respectively. This graph shows that the accuracy of correct action completion varied based on the instruction.

Based on Figure 4.3, out of the 11 participants (data for P1 not collected for this phase), two participants did not complete the expected action of snoozing the alarm clock noise (82% accuracy) based on the system's guidance. One participant did not complete the expected action of placing the coffee lid on the coffee machine (coffee step 1), which was the same case for screwing in the lid (coffee step 2) and turning on the fan (all having 91% accuracy). All participants completed the expected instruction of turning on the coffee maker (coffee step 3) (100% accuracy). Lastly, three participants did not complete the expected action of opening the previous operator's specified cabinet drawer and turning off the lights (each having 73% accuracy).

During the operator phase of the study, which was the second phase for each participant, the number of trials it took to obtain a clear and accurate output for each instruction is shown in Figure 4.4. Each box plot shows the distribution of the number of trials over all the participants for the instruction related to the object on the x-axis. Each box portion, if visible, shows the interquartile range of the number of trials, with a horizontal bar displaying the median number of trials. The top whisker extends to the maximum number of trials the instruction took for any of the participants, while the bottom whisker (if visible) shows the minimum. Lastly, the black cross marker shows the mean number of trials for each instruction over all of the participants.

From left to right in Figure 4.4, obtaining a clear and accurate guidance output for the alarm clock instruction from their written instruction and image took participants 1.5 trials on average, represented by the black cross marker. The first coffee step, placing the lid on the coffee machine, took 2.25 trials on average. The second coffee step, screwing in the lid, took 1.17 trials on average. The third coffee step, turning on the coffee machine, also took 1.17 trials on average. The fan instruction took 2.17 trials on average. Obtaining an accurate output for opening their selected cabinet drawer took 1.25 trials on average.

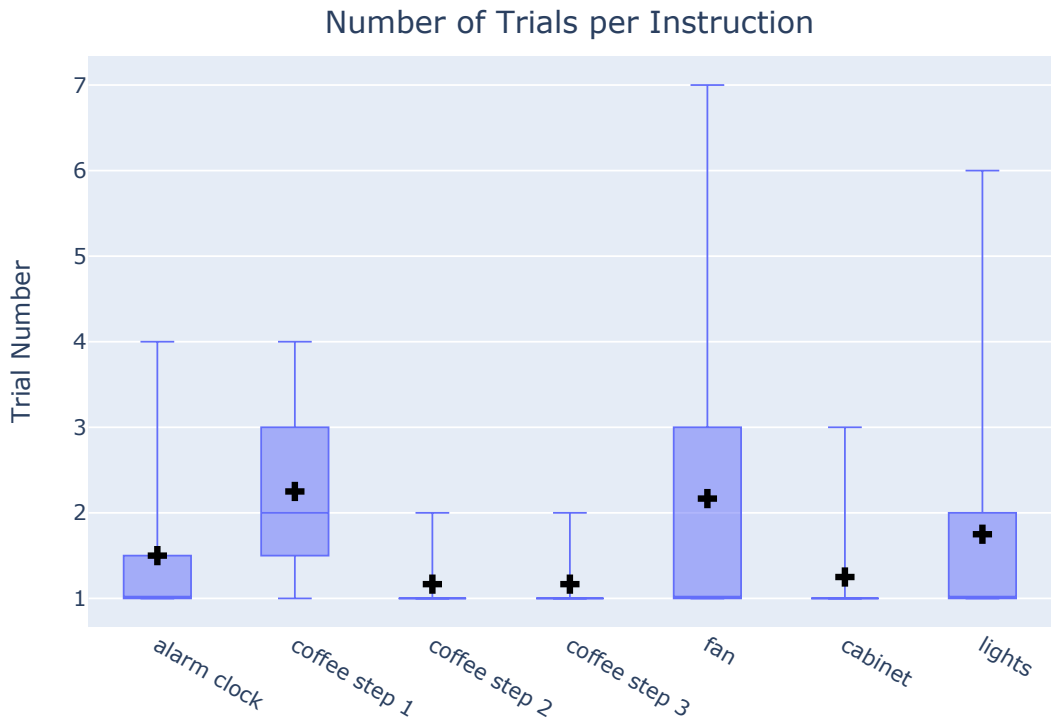
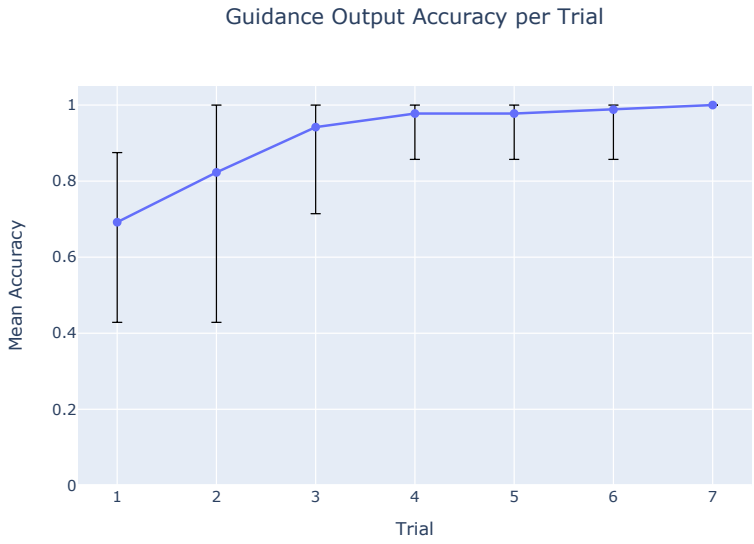


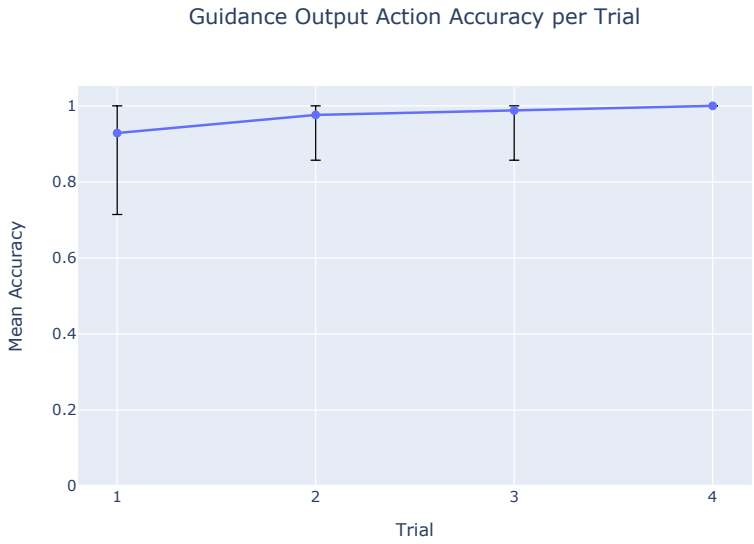
Figure 4.4: Distribution of the number of trials it took participants to obtain an accurate output for each instruction during the operator phase. The whiskers on each box plot show the minimum and maximum number of trials for the instruction related to the object on the x-axis, and the black cross marker shows the mean number of trials for the instruction.

Lastly, their lights instruction took 1.75 trials on average. Thus, the mean number of trials for all instructions was less than 3, and the median was 1 for all instructions, except for the first coffee step, which had a median of 2 trials. Despite every average number of trials being lower than three, it is apparent from the plots that some participants had more than three trials for some instructions. For example, the maximum number of trials for the fan instruction was seven, and it was six for the lights instruction.

In addition to the number of trials per instruction, Figure 4.5a visualizes the mean accuracy of all of the guidance outputs over each trial number for all participants during the operator



(a)



(b)

Figure 4.5: Mean accuracy of all guidance outputs' visual and location (a; on top) or just action visual (b; on bottom) per trial number for all participants during the operator phase. Black error bars for each trial show the maximum and minimum output accuracy values from all of the participants during that trial.

phase. The error bar above the mean data point for each trial shows the maximum accuracy value from all of the participants in that trial number. This is similar for the error bar below the mean data point but for the minimum accuracy value during that trial. The x-axis stops at seven trials because all participants had 100% accuracy for all outputs by trial seven, as displayed.

Figure 4.5a shows that the accuracy of the guidance outputs increases over most subsequent trials, increasing more drastically during the earlier trials. In ascending order of trials, the guidance outputs from all participants in trial one had a mean accuracy of 69%. Trial two had a mean accuracy of 82%. Trial three had a mean accuracy of 94%. Trials four and five both had a mean accuracy of 98%, which are the only trials that don't have an increase in the guidance output accuracy. Trial six had a mean accuracy of 99%, and all outputs from trial seven had 100% accuracy. The error bars show that some participants had either higher or lower accuracy values than the average for each trial, due to several reasons explored in Section 4.3.

Figure 4.5b shows similar accuracy metrics, but only for the selected action visual. Thus, location inaccuracies are not considered in this Figure. As shown, the accuracy of the action visual is much higher than the overall accuracy and achieves 100% accuracy by trial four. The action visual mean accuracy is greater than 90% for all trials shown in Figure 4.5b. Therefore, the overall accuracies shown in Figure 4.5a are typically influenced more heavily by the output location precision, as opposed to the action visual selected.

Lastly, responses to the quantitative questions from the post-study interview are shown in Figure 4.6. The average rating for Q1 was 6.17, meaning that participants largely believed the system provided guidance visuals that they expected for their input instructions. The average rating for Q2a was 2.17, meaning that participants mainly felt it was easy to write instructions to get an expected output. The average rating for Q2b was 2.42, meaning that

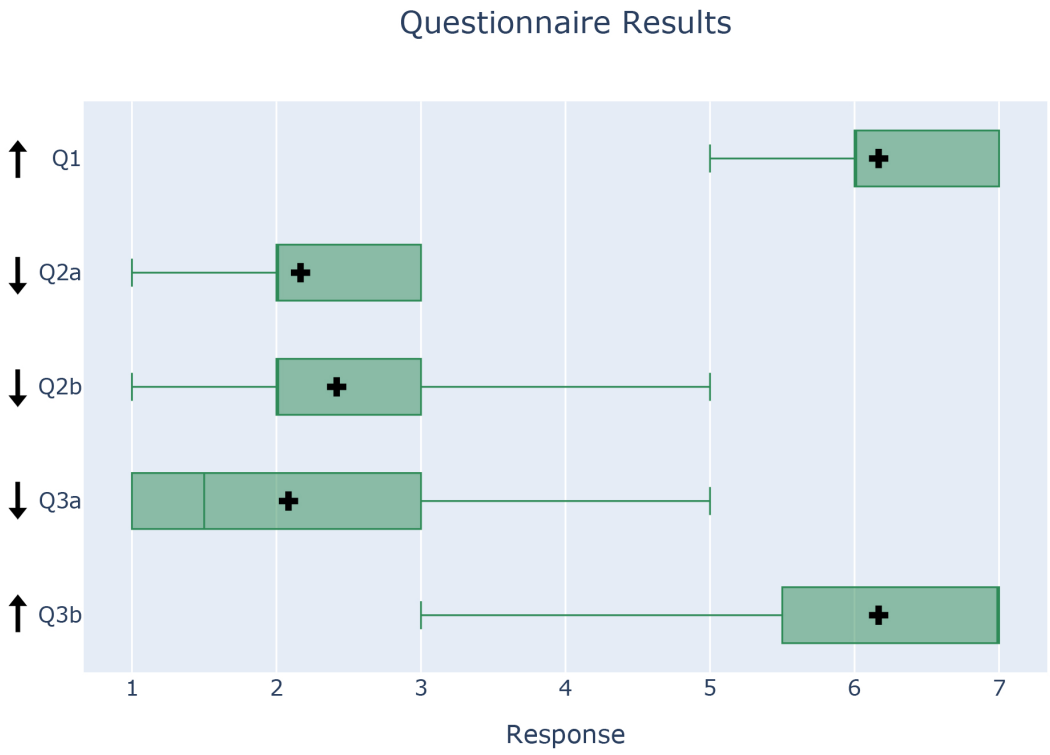


Figure 4.6: Post-study questionnaire quantitative responses. The black cross marker for each question indicates the mean response value over all participants. An up arrow next to the question number signifies that a higher number is more favorable, while a down arrow means a lower number is more favorable.

Q1: "Did the proposed system provide guidance visuals that you expected for your input text instructions?"

Q2a: "Given the guidelines, how easy was it to write instructions to get an expected output?"

Q2b: "Given the guidelines, how easy was it to take good pictures to get an expected output?"

Q3a: "When needing to go back and revise instructions and/or pictures after an output was incorrect, was it easy to know what to change?"

Q3b: "Did those changes make the output better?"

participants mostly felt it was easy to take good pictures to get an expected output.

The average rating for Q3a was 2.08, meaning that participants mainly thought it was easy to know what to change when an output was incorrect. Lastly, the average rating for Q3b was 6.17, meaning that participants largely believed their changes made the output better.

4.3 Discussion

4.3.1 General-Purpose System Validation

Our first research question, RQ1, asks how a General-Purpose AR Task Guidance System can be designed according to Components 1-6. The results from our user study validate that these components are present in our proof-of-concept system.

Components 1 and 2 - General Tasks and Environments

Our user study tested our system with tasks and an environment for which the system had no a priori knowledge. Additionally, the environment was not specially prepared for the system, as it was a typical office space containing normal objects that changed location between each participant and phase. The tasks chosen for our study required actions in our predefined visual library of "press", "pull", "twist", and "pickup/place". According to Figure 4.4, participants stated that 100% of outputs generated by the system were clear to a new person in less than three trials on average. Since our study's tasks and environment were selected as samples from possible general tasks and environments, our proof-of-concept system showed that it is capable of generating guidance for these general tasks, validating **Component 1** in our system, and for this general environment, validating **Component 2**

in our system.

To work with generic tasks, our system used free-form text instructions written by participants, which varied based on the person. Thus, our system had to handle many cases of text formats, object names, action terminologies, and varying levels of detail, among other differences, as shown below.

Examples of instructions from participants' final trial written for achieving the end state of "fan on":

- "Behind the fan, turn the knob that is located at the top to the right"
- "Pinch the right knob on the fan and rotate clockwise from 0 to 1."
- "Twist the right knob, with labled numbers at the back of the fan to turn it on"
- "Locate the fan, find the back of the fan and look for a knob to turn. Turn the knob clockwise."
- "turn the knob on the fan"

Our system generated clear guidance to the operator for all of these instructions above. Other interesting cases of written instructions that our system still generated clear output visuals for include the following:

1. **Typos (underlined) and lengthy instructions:** "Find the button on the coffee maker towards the bottom of the base. Click the lever buttone so that it is in the on setting."
2. **Varying object descriptions (cup instead of cap):** "pick up white cup and place on top of coffee machine"
3. **Vague instructions:** "grab top most cabinet"

These cases were able to be handled by our system, allowing clear guidance outputs to still be output for instructions such as those above.

Components 3 and 4 - Automatically Determining Where to Place Visuals without Complex Technical Input

Participants used our system by writing text instructions and taking pictures without the need to create any complex technical input. As shown in Figure 4.6, results from our interview question Q2a show that users believe this input was not complex, as they rated that it was easy to write instructions. This is similar for taking pictures, as participants rated that it was easy to take good pictures in Q2b. Since the text instructions and pictures are the main inputs from the user, the responses from Q2a and Q2b validate **Component 3** in our system. Additionally, our system automatically determined the location and predefined action visual of the generated guidance without requiring a system expert, validating **Component 4** in our system.

Components 5 and 6 - Clear Task Guidance Robust to Object Location Changes

While the participants in the operator phase believed that the guidance outputs from their setup of the system were clear to a new person, we tested this by having the next participant follow the generated guidance visuals without any knowledge of the tasks. Based on Figure 4.3, the guidance helped participants complete the expected actions for each task, as the average correct action completion over all the instructions was 85.7% (average of all accuracies in Figure 4.3). For seven instructions, this means that an average of between one and two instructions were not completed correctly by participants. Thus, our proof-of-concept system does generate a majority of guidance that is understandable by a naive user, as stated by **Component 6**, but can improve in future systems. Additionally, participants achieved this mean accuracy of 85.7% based solely on the visuals, as they were not allowed to see

the text instructions, and with the objects moving locations between participants, meaning that the system mainly outputs understandable guidance despite object location changes, as stated by **Component 5**. Their expected task completion accuracy would most likely increase if they were able to see the text instruction alongside the guidance visuals.

4.3.2 System Usability by Non-Experts

Our second research question, RQ2, asks how a general-purpose AR task guidance system can be designed to be usable by non-expert users, including both operators and end users. Results from the questionnaire shown in Figure 4.6 show participants' ratings on the usability of the approach used by our system. One aspect of the usability of the approach is how easy it is for non-expert users to write instructions that generate an expected output. Participant responses to question Q2a show that they mainly thought it was easy to write instructions for this purpose. Part of using the system also included updating instructions over multiple trials to fix incorrect outputs. In response to Q3a, participants rated that they primarily felt it was easy to know what to change when an output was incorrect.

In interview question Q3a responses, four participants elaborated by answering that they thought knowing what to update was intuitive based on seeing the incorrect visual placement. Furthermore, four participants said that they added specificity to their instruction, which they felt made outputs better. This is related to responses in Q2a where 50% of the participants answered that they felt the instructions needed to be more specific than they first thought. Three of these six also said that was only the case for certain instructions that took multiple trials to correct. The instructions these three participants were referencing were different for each of them. One stated the fan instruction needed to be specific, another stated the light switch, and the last referenced the alarm clock instruction.

Below is an example of the participant updating the fan instruction with more specificity:

- **Trial 1:** "Twist the knob at the back of the fan to turn it on"
- **Trial 2:** "Twist the right knob at the back of the fan to turn it on"
- **Trial 3 (final):** "Twist the right knob, with labled numbers at the back of the fan to turn it on"

Two participants also said in their response to Q2a that they felt it was like instructing a person.

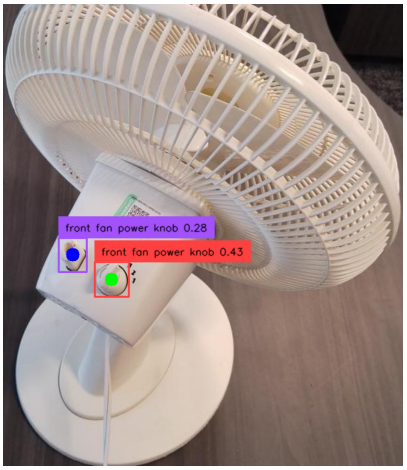
The instructions written on Trial 1 by one of these participants are shown below:

1. "Move the cap on top of the coffee maker"
2. "Twist the cap"
3. "Press the switch on the coffee maker"
4. "Pull open the middle cabinet"
5. "Twist the fan knob"
6. "Press the snooze button on the alarm clock"
7. "Press the off button on the light switch"

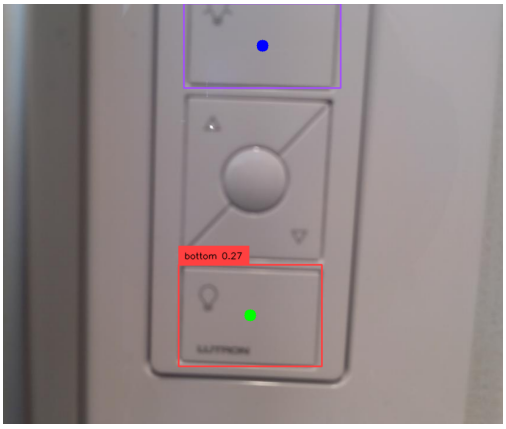
The only output that this participant needed to update was the fan instruction, which was updated to "Twist the fan knob with numbers on it", which corrected the output. Another consideration was that one participant responded in Q2a that as a non-native English speaker, it was challenging to think of words to describe specific terminology, such as "twist" and "knob". This participant was still able to generate accurate output visuals, only needing to update one instruction in a second trial.

The ease of taking good pictures that generate an expected output is also important to the usability of the approach. Responses to question Q2b show that participants mainly thought that it was easy to take good pictures for this reason. When explaining their response for Q2b, seven of the participants answered that it took them time to adjust to the HoloLens camera position, which is a bit above eye level, but their ratings show that they were able to adjust enough to feel that taking pictures was easy. Two participants also answered Q2b elaborating that they thought they had to get close to some objects, and that this could be hard for somebody who cannot move their body easily.

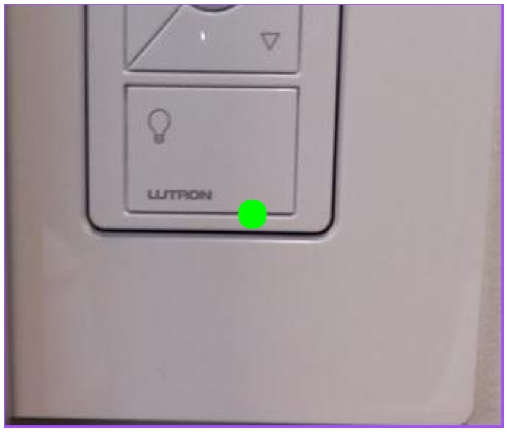
(a) Fan image on trial 1



(b) Fan image on final trial



(c) Light switch image on trial 1



(d) Light switch image on final trial

Figure 4.7: Pictures of fan and light switch compared between first and final trials for one participant. The green dot shows the output from object detection during Localization.

One participant also stated in Q2b that they believed problems with pictures were less intuitive than instruction wording. This is related to two participants' responses to Q3a, saying that they were sometimes unsure whether to change the instruction or picture when updating incorrect outputs. An example of the images taken on Trial 1 and the Final Trial by the participant who stated that picture issues were less intuitive is shown in Figure 4.7.

As shown in Figure 4.7, object detection output the correct knob, as shown by the green dot in each image. However, due to other reasons, such as tracking or 3D spatial map errors, the output from Trial 1 was not accurate for this participant. The pictures from the final trial show that the participant decided to take the picture from a different angle for the fan, and closer to the bottom button for the light switch. The final trial shown was Trial 3, which output accurate guidance visuals.

4.3.3 Further System Analysis

Analysis of Operator Phase Incorrect Outputs

While the system did achieve 100% clear outputs for each participant during the operator phase by their final trial, incorrect outputs during previous trials had several reasons for not providing clear visuals. A breakdown of the causes for these incorrect outputs is shown in Figure 4.8. We conducted this breakdown through observation and subjective analysis of data collected during the study. From our analysis, the primary causes for incorrect outputs are object detection (GroundingDINO) and semantic output (GPT-4V), which are at fault 36.9% and 30.8% of the time, respectively. The next highest cause was tracking/system errors at 15.4%. Then, 9.23% of issues were due to the 3D spatial map from the HoloLens. Lastly, the user's picture and text instruction were attributed as the cause of incorrect outputs only 6.15% and 1.54% of the time. These issues are what caused participants to need multiple

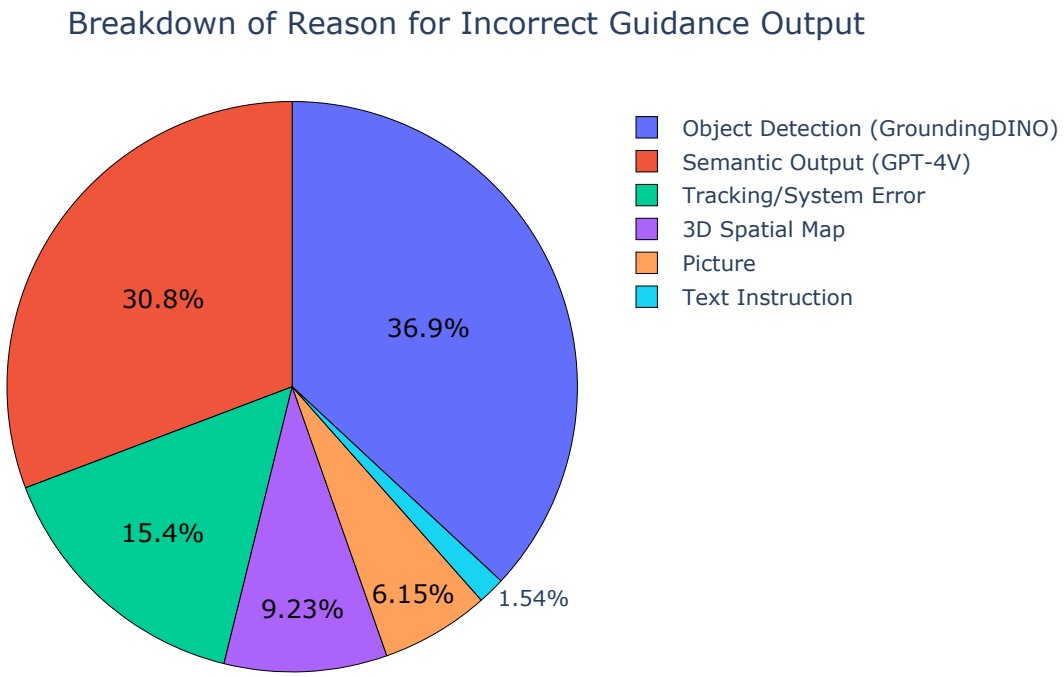


Figure 4.8: Breakdown of causes of incorrect outputs for participants during the operator phase.

trials of updates, shown in Figures 4.4 and 4.5a.

Every case is different due to the nature of current ML models, but below are examples of why object detection and semantic understanding faults occurred, as shown in Figure 4.8:

Examples of major reasons for object detection (GroundingDINO) faults:

1. **Reading Words in Pictures:** The object detection model used, GroundingDINO, was found to not be capable of reading words in pictures. Many participants specified words on objects, which did not increase the accuracy of the output as they expected. For example, the alarm clock snooze button contained the text “SNOOZE/SLEEP

OFF", but specifying this text would not increase the accuracy of finding the correct button since GroundingDINO cannot read the text in the image.

2. **Lack of 3D Spatial Understanding:** The object detection model is only trained on 2D images. Thus, it does not have spatial understanding like GPT-4V seems to have, which several participants used in their instructions. For example, specifying the "knob on top of the fan", would lead the model to prefer outputting objects at the top of the 2D image, despite the fan knob being in the center of the image.

3. **Imperfect 2D Location Reference Comprehension:** Specifying the "top cabinet drawer", "bottom button", or "right knob" also occasionally output the wrong object, despite it being very clear to a human.

4. **Multiple Similar Items in Image:** The object detection model was found to occasionally confuse objects in an image. For example, a prompt for "white coffee machine cap" would typically find the expected object, but if the image contained any other circular-looking white objects, it may confuse it for a cap.

Examples of major reasons for semantic understanding (GPT-4V) faults:

1. **Using 3D Spatial Information:** Because the semantic understanding model used, GPT-4V, seems to have some spatial understanding of the world, the object description it outputs sometimes includes 3D spatial descriptions, as opposed to describing the position just in the 2D image. For example, an instruction including "button on top of the alarm clock" may output the object description as such, which is not understandable by the object detection model, as described above.

2. **Pickup/Place Ordering:** The semantic understanding of the pickup/place action required an extra step of not only locating the objects but also determining which is

the object to pick up and which is the location to place the object. The model was found to have issues determining which object should have the action "pick up" and which should have the action "place the picked up object at this location". An example of this case is shown in Figure 4.9. This example shows that the initial incorrect output repeats the "white caution lid" object twice, while the correct output determines that the location to place the lid is the "top of the coffee maker".

3. **Ambiguous or Incorrect Object Descriptions:** The semantic understanding model is not 100% accurate for vague instructions. For example, the instruction "press the light switch off button" sometimes may output the correct "bottom button", but sometimes may output another button, since the model does not perfectly understand the room's specific light switch.

Tracking/system errors were also caused due to several reasons in different cases. Reasons for tracking/system errors shown in Figure 4.8 are explained below.

Examples of major reasons for tracking/system faults:

1. **Getting Too Close to Objects:** Sometimes, participants got very close to objects to take pictures of them, which would disorient the HoloLens optical trackers. When this happened, typically the instruction the user was on during this error would have an output in a completely different location than the scanned object, often confusing the participant.

2. **Errors in System Software:** Issues in the system software itself occasionally occurred, which typically caused the system to output no visual for an instruction. These errors include not handling when users moved onto the next instruction without confirming any picture, users taking pictures while on the wrong current instruction due to voice commands not registering, or not handling errors from the GPT-4V API.

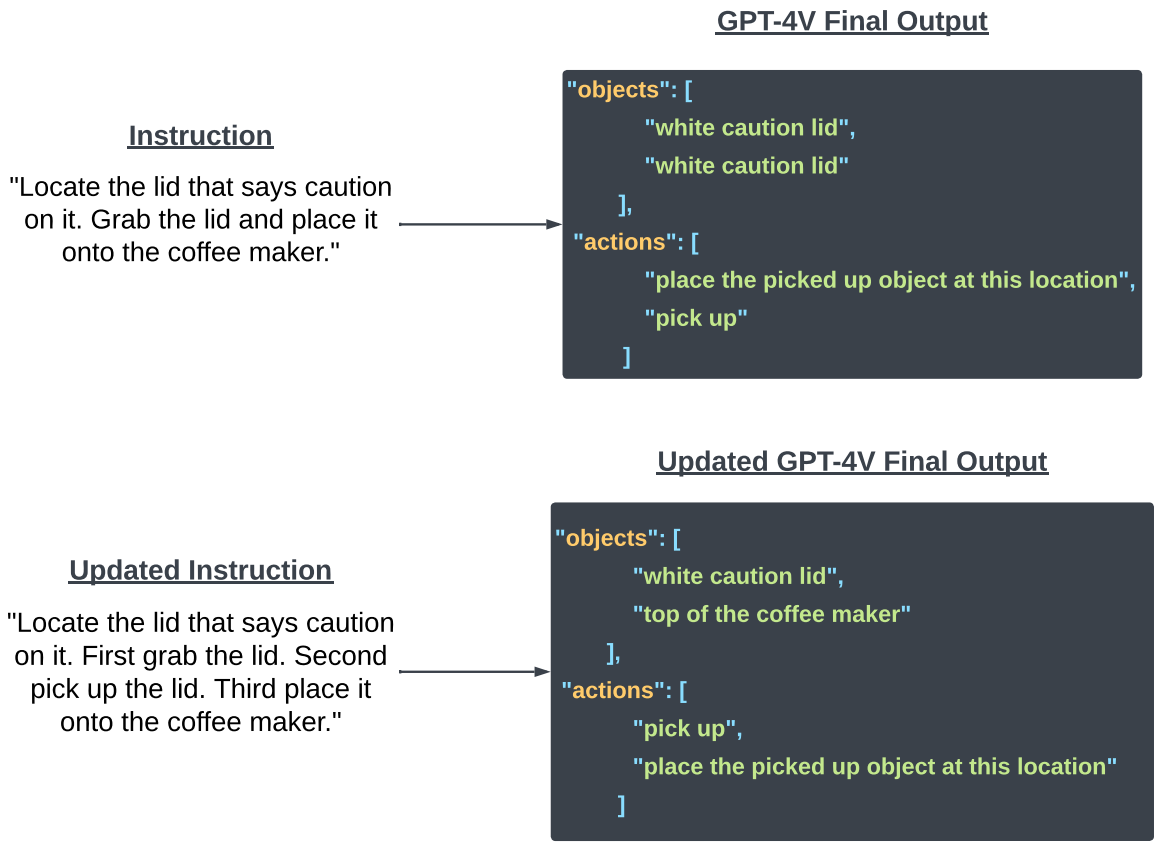


Figure 4.9: Initial operator instruction leading to incorrect GPT-4V output (top), compared to updated operator instruction leading to correct GPT-4V output (bottom)

Errors due to an imprecise 3D spatial map, as shown in Figure 4.8, are mainly due to drawbacks of current spatial mapping technology on the HoloLens 2. The main instruction that encountered problems with the spatial map was turning on the fan. Part of the issue was due to the unusual geometry of the desk fan used, shown in the left of Figure 4.1. Additionally, the HoloLens 2's spatial mapping is dynamic and not consistently accurate every frame, and has notable trouble tracking shiny or dark objects. We found that walking around and allowing the HoloLens to see all of the objects immediately after app startup reduced the prevalence of spatial map errors, which were the main reasons for a fan instruction requiring

seven trials, as shown in Figure 4.4.

The last reasons for incorrect outputs were attributed to pictures that did not match the provided Picture Guidelines, as well as instructions that were overly vague even for a person to understand, which only happened once. Thus, the low percentage of these cases shows that the majority of incorrect outputs from our system are not due to user instruction and picture inputs, but instead due to system and technical flaws, according to the analysis in Figure 4.8. Thus, our approach seems to be viable and usable by users, while technical issues need to be addressed in the future, as discussed in Chapter 5.

Analysis of End User Phase Incorrect Task Completion

To understand the reasons for participants not completing the expected actions during the end-user phase, a subjective analysis was conducted from observations and data collected during the study, which is displayed in Figure 4.10. The leading cause of incorrect actions during the end-user phase was due to the system placing visuals in incorrect locations, despite all visuals being stated as clear during the operator phase of the previous participant, which occurred 45.5% of the time. The next highest reason for participants not completing the expected action was due to tracking/system errors at 27.3%. Then, participants not matching the picture displayed to them happened twice, or 18.2% of the time. Lastly, one time the actual guidance visual was not understood, or 9.09% of the time. These issues are the primary causes of the imperfect accuracies shown in Figure 4.3.

To generate the guidance visuals for the end user, the system uses object detection again on the matched pictures taken by the end user, as well as the object descriptions and actions generated during the operator phase. Thus, although guidance outputs are clear during the operator phase, Figure 4.10 shows that there is still a possibility for guidance outputs to be

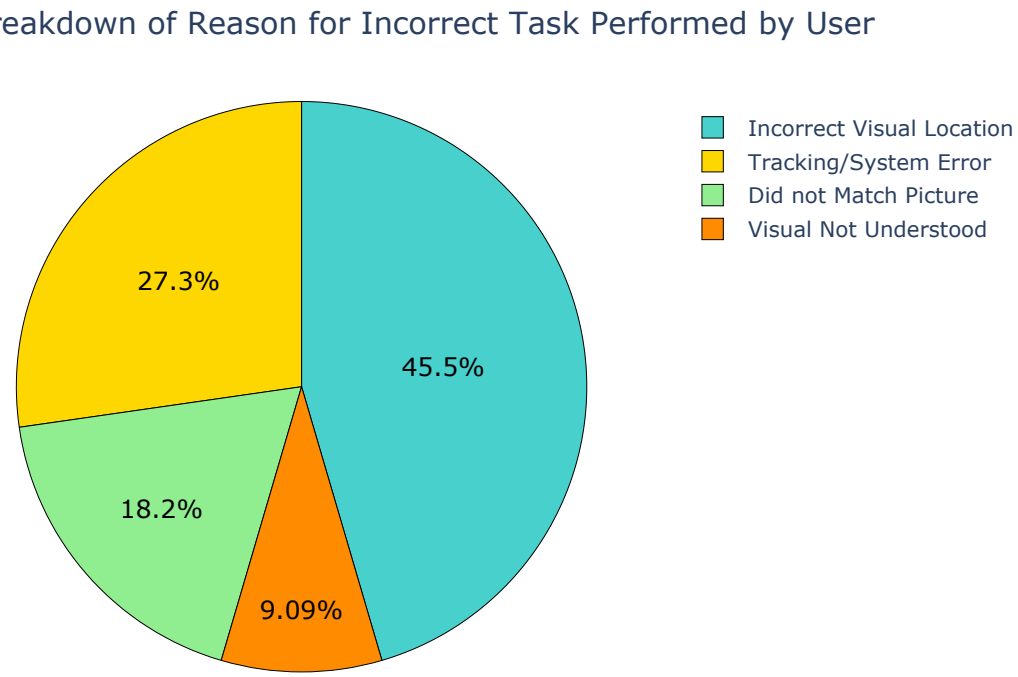
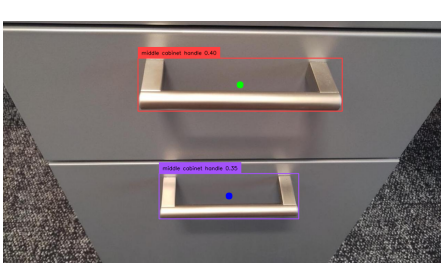


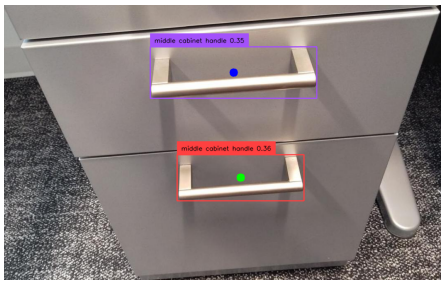
Figure 4.10: Breakdown of causes of participants' performing incorrect tasks based on guidance visuals during the end-user phase.

placed at an unclear location during the end-user phase. This primarily occurred for objects with multiple similar subparts, such as three cabinets, or multiple light switch buttons, as shown by the lower accuracy for those instructions in Figure 4.3. Examples of these cases are shown in Figure 4.11.

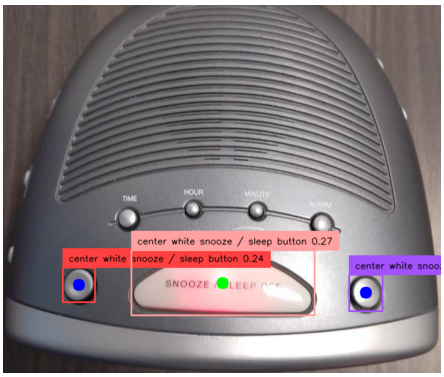
As shown in Figure 4.11, images that were mostly matched by the end user still resulted in different objects being output by object detection. Image (a) shows that the operator image resulted in the middle cabinet, as they expected, while the end user image (b) resulted in the bottom cabinet. Image (c) shows that the operator image resulted in the correct snooze button, while image (d) resulted in the wrong button on the left. This is because current



(a) Operator image of cabinet



(b) End user image of cabinet



(c) Operator image of alarm clock

(d) End user image of cabinet

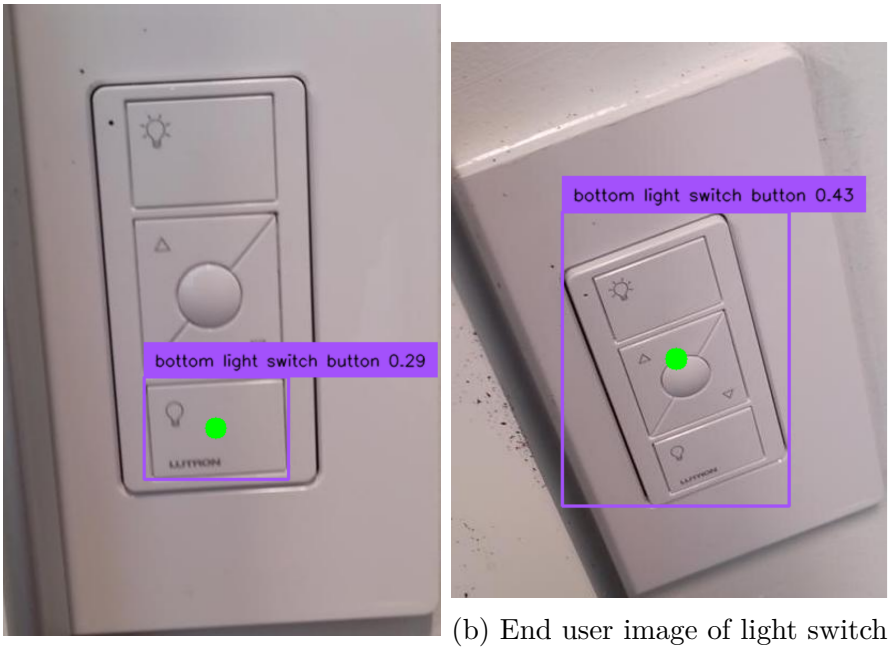
Figure 4.11: Comparison of operator (a and c) and end-user images (b and d) that resulted in different object detection locations. The green dot in each image represents the output from object detection.

object detection models, such as GroundingDINO, can change their output even due to very slight changes, such as small differences in the angle of the image or distance to the object, which can affect outputs with multiple detections of similar confidence values.

The second highest fault case was a tracking/system error. The reasons for these tracking/system errors are the same as those that occurred in the operator phase, including getting too close to objects or errors in the system software. These issues errors typically caused the system to output no visual for the instruction the error occurred during.

Participants not matching the picture displayed to them was the next most common issue. This means that the guidance output by the system was not placed correctly because the

participant during the end-user phase changed the angle or another aspect of the picture enough to cause differences in the output of the detection model. The end user is expected to match the picture as much as possible to decrease errors such as these, but the system does not deter users from doing so, which can cause unclear guidance visuals, as shown by the example in Figure 4.12.



(a) Operator image of light switch

(b) End user image of light switch

Figure 4.12: Picture of an operator image of the light switch compared to an end-user image of the light switch, which should try to be matched. The green dot in each image represents the output from object detection.

The last issue, which occurred once, or 9.09% of the time, was the end user not understanding a visual, despite it having the same location and action that the operator stated was clear to a new person. This occurred due to the twist hand visual not including a direction arrow, so the direction of the twist was ambiguous and led to the participant not performing the expected action. Thus, it is important for all visuals to be very clear to all people, which could have improved our proof-of-concept system.

Comparison of Operator and End User Phase Results

Both the operator and end user phases are important to our approach, as the goal is for the operator to be able to easily generate guidance visuals to guide an end user through a set of tasks, and the end user should be able to easily receive and use the generated guidance. However, since the operator sets up the system once for a set of tasks and has the ability to update instructions and pictures over multiple iterations, we believe it is more important for such a system to optimize the end user accuracy. This is also because the system can be used by multiple end users for several uses with the same objects in possibly various environments, and the end user requires clear guidance to complete their tasks correctly. Thus, our proof-of-concept system's average correct task completion accuracy of 85.7% during the end-user phase (shown in Figure 4.3) is a good start for a general-purpose task guidance approach but should be a focus of optimizing in future work, which will largely be a byproduct of more accurate future vision-language ML models.

Comparing the mean accuracy of the end user task completion per instruction (Figure 4.3) with the mean accuracy of the operator number of trials per instruction (Figure 4.4), there are apparent differences in which instructions caused inaccurate results. In the operator phase, the two instructions with the highest mean number of trials were turning on the fan and placing the lid on the coffee machine. However, in the end-user phase, the two instructions with the lowest completion accuracy were opening the cabinet drawer and turning off the lights.

This is most likely because the operator phase also requires participants to determine and update instruction wording to output accurate visuals, while the end user phase is only affected by the picture since the instruction is already set up by the operator. Thus, issues with semantic understanding and GPT-4V, as shown in Figure 4.8, are not encountered

during the end-user phase. Because of this, instructions that are more susceptible to semantic understanding errors of the written instruction, such as placing the lid on the coffee machine and turning on the fan, are less prone to errors just from the picture during the end-user phase. In contrast, some instructions that required fewer operator iterations in Figure 4.4, such as opening the cabinet drawer, could be more prone to issues due to differences in the picture during the end-user phase, as shown in the example in Figure 4.11.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

We created a General-Purpose AR Task Guidance approach and proof-of-concept system capable of generating guidance for tasks unknown to the system a priori and in arbitrary environments with no special preparation. Our approach uses a predefined library of reusable action visuals, which contains visuals for all actions that can be output by the system as task guidance. Once this library of visuals is created, our approach can generate guidance for tasks requiring any of these actions. Then, the approach can be used without any complex technical input by an operator, who knows how to perform the tasks, and an end user, who is receiving guidance on how to use the objects in the environment to complete their tasks.

The operator generates guidance from only natural language instructions and pictures they take so that the system can determine which visuals to place and where to place them. Then, an end user takes matching pictures, which determine where to place guidance visuals, even if the location of objects move, or the objects are located in completely different environments. At this point, the naive end user can follow the 3D guidance visuals while reading their current instruction.

Our system can accept natural language instructions written by an operator with a semantic understanding approach that utilizes a vision-language ML model, GPT-4V, with a special-

ized system prompt to output relevant object descriptions needed to complete each instruction, as well as the action required. Our system then implements a localization approach that uses the GroundingDINO object detection vision-language ML model to determine the 2D location of the relevant objects in the images taken by the operator. Using these 2D locations, our system has a rendering approach that determines the 3D location and surface normal of each object using a tracked 3D spatial map and places the correct action visual using this 3D information, which is one of the implemented hand animations showing the action for the user to perform.

The semantic understanding approach with GPT-4V and localization approach with GroundingDINO both utilize a two-pass approach, where the first "Crop to Object" pass is meant to crop images to the region containing the relevant objects. Then, the cropped image is used to obtain the final output from the second pass. Additionally, the localization approach obtains the most specific box by filtering out detected boxes containing other box centers.

Our approach was validated in a user study, which showed that our system was usable by participants with no prior knowledge of the system by only using written instructions and pictures as input. Participants' responses to a post-study interview show that they felt the system made it easy to write instructions and take pictures to generate an expected guidance output. When outputs were not correct in the first trial, participants also felt it was easy to know what to change to increase the accuracy of the outputs.

Additionally, all participants as operators in our user study were able to generate guidance visuals that they stated would be clear to a new person by their final trial. Participants as end users followed guidance from the system's 3D visuals that were generated by the previous participant's operator phase, and they completed the expected action defined by the operator with high accuracy, especially given that end users were not allowed to read the current instruction text. We also break down reasons for our proof-of-concept system's

incorrect outputs during the user study, providing analysis to help future systems make accelerated design decisions.

5.2 Future Work

5.2.1 Future Evaluations

Formal Comparison with Another Task Guidance Approach

To determine how our proposed General-Purpose AR Task Guidance Approach compares to alternative task guidance approaches that are not general-purpose, a formal comparison study should be conducted in the future. Our proof-of-concept system could be compared to a system with similar features, but the operator has to semi-manually place the guidance visuals. For example, an operator could select the correct action visual for each instruction and determine a placement using a similar raycast method on the 3D spatial map. Both approaches should also ensure to be tested with objects that move locations in the environment, as tested in our user study.

Formal Comparison of Reusable Guidance Visuals

Our approach uses visuals in a predefined action library that can be reused for general tasks requiring actions from this library, and our choice to use 3D hand visuals that mimic human actions was not backed using research methods. Thus, a formal comparison study should be conducted in the future to determine what visuals are both reusable and most beneficial in task guidance systems. For example, such a study could compare our approach using hand visuals, object highlighting, text and arrow annotations, and combinations of these as

guidance visuals.

Formal Evaluation of General-Purpose Approach Accuracy

While our user study validated our approach's viability and usability for one set of tasks previously unknown to the system and for one environment with no special preparation, a formal evaluation method should be determined and conducted on a larger set of general tasks and environments. This evaluation could utilize numerous images of objects and tasks using actions from a set of predefined actions, and determine the accuracy of the correct 2D object location and selected action for instructions needed to complete the tasks. Additionally, to simulate differences in pictures taken by end users, location changes of objects, or even different object environment backgrounds, transformations could be applied to the object images to measure the accuracy under these circumstances. Thus, it may even be possible to curate a dataset for future similar task guidance approach evaluations.

5.2.2 Future System Enhancements

Immediate System Enhancements

Participants in our user study interview were asked about possible improvements to our proof-of-concept system. Multiple participants stated that they wanted to be able to see a live preview of the HoloLens camera feed, similar to phone cameras. Additionally, several participants answered that having more guidance visuals could be nice, such as highlighting objects, clockwise and counter-clockwise twists, and a press-and-hold animation. Thus, implementing these improvements is an immediate system enhancement directly from user feedback.

Task Completion Recognition and Feedback

One limitation of our approach is that it does not determine when the end user has completed their task, as well as whether they are performing the correct action or not. Our system should be extended in the future to provide end users with this feedback to correct errors during task execution, and automatically detect and transition between instructions when users are complete. All actions in the predefined action library would need to be recognizable from video sequences of users performing an action in general environments. Thus, it could be possible to use current vision transformer models that are capable of detecting specific actions from a video [15].

Using Extracted Object Models as Visuals

Currently, a limitation of our approach is that it requires a predefined visual library that needs to be created by a system expert to encompass all possible actions required by the tasks. These visuals need to be reusable, so they are generic and are not as detailed as using a 3D model of a specific object. Current research has shown that using detailed object models is more clear to users as task guidance than generic visuals such as arrows [21]. Thus, extending our system to be capable of extracting 3D models from the tracked 3D spatial map of relevant objects and using these models as detailed guidance visuals could provide clearer guidance, similar to the approach used by Park et al. [20].

Enable Tracking of Objects in Real Time

Another limitation of our current system is that its visuals are rendered based on the location of the object during the localization step. If the objects are moved from the location they were

detected during localization, the guidance visuals will not update with these moving objects. Thus, to provide more robust and clear guidance visuals, our localization and rendering approaches should be extended to be capable of tracking objects moving over time. Thus, computer vision algorithms or other object detection models would need to be used that work in real time.

Remove Need to Take Pictures

The largest setup time requirement when using our system is having to take pictures of each object. The operator needs to take pictures to generate guidance, and the end user needs to match these pictures each time they want to receive task guidance. Thus, two possible enhancements exist: (1) Remove the end user picture phase, or (2) Remove the need for all pictures in both the operator and end user phases. The first change would require the system to compare objects in the environment to the known operator images during the end-user phase and know which physical objects correspond to objects in the images. The second and more ambitious possibility would also require the system to detect objects relevant to the current instruction in real time as the operator is surveying the environment, removing the need to take pictures altogether.

Combine Semantic Understanding and Localization

As a large part of the current system relies on vision-language ML models such as GPT-4V and GroundingDINO, increasing the reliability and accuracy of these models would drastically increase the system's output accuracy. Specifically, a large limitation of our system is that it uses two separate models, one for semantic understanding and another for object detection. This led to several issues with the two models not having the same capabilities

and understanding. Thus, merging these two models into one single model capable of both text and image semantic understanding and localization could potentially streamline our approach. Models with both of these capabilities seem to be advancing rapidly, such as the CogVLM architecture [29].

Generate Instructions through Operator Demonstration

Our approach currently requires instructions to be input using natural language. This is one possible approach to generate guidance visuals in a usable manner, but another one that could possibly be simpler could be for the system to watch the operator perform each task, and generate instructions and visuals simply from these demonstrations. Thus, this enhancement is similar to the enhancement of removing the need to take pictures, but replacing an instruction input with an operator recording phase. This is also similar to InstruMentAR's recording phase, but for general tasks and environments without finger-attached hardware, as opposed to limited tasks related to machine instruments panels [17].

Improve Incorrect Guidance Automatically or Manually

As shown in Section 4.3.3, a large source of faults in the system are due to object detection and semantic understanding errors. One improvement that could be made to reduce these errors is to have the system allow the operator to manually adjust the outputs from object detection (GroundingDINO) or GPT-4V. For example, the system could show the user the final object detection image, and if the 2D coordinate is located on the wrong object, the operator could manually move it. Another approach to do this automatically is to use prompt learning, which is a method in machine learning that improves a model's ability to update text prompts to obtain an expected output. For instance, if an operator notices an output

is correct and performs the correct action, the system could feed the incorrect output 2D location and updated 2D location to the model to increase its future accuracy.

Bibliography

[1] AR Foundation | AR Foundation | 5.1.3, 2024. URL https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.1/manual/index.html.

[2] ChatGPT, 2024. URL https://openai.com/chatgpt.

[3] IDEA-Research/GroundingDINO, April 2024. URL https://github.com/IDEA-Research/GroundingDINO.

[4] OpenAI Platform, 2024. URL https://platform.openai.com.

[5] Y. Bengio. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009. ISSN 1935-8237, 1935-8245. doi: 10.1561/2200000006. URL http://www.nowpublishers.com/article/Details/MAL-006.

[6] Yuanzhi Cao, Xun Qian, Tianyi Wang, Rachel Lee, Ke Huo, and Karthik Ramani. An Exploratory Study of Augmented Reality Presence for Tutoring Machine Tasks. In *CHI Conference on Human Factors in Computing Systems*, pages 1–13, Honolulu HI USA, April 2020. ACM. ISBN 978-1-4503-6708-0. doi: 10.1145/3313831.3376688. URL https://dl.acm.org/doi/10.1145/3313831.3376688.

[7] Francesca De Crescenzio, Massimiliano Fantini, Franco Persiani, Luigi Di Stefano, Pietro Azzari, and Samuele Salti. Augmented Reality for Aircraft Maintenance Training and Operations Support. *IEEE Computer Graphics and Applications*, 31(1): 96–101, January 2011. ISSN 0272-1716. doi: 10.1109/MCG.2011.4. URL http://ieeexplore.ieee.org/document/5675633/.

[8] G. Dini and M. Dalle Mura. Application of Augmented Reality Techniques in Through-life Engineering Services. *Procedia CIRP*, 38:14–23, 2015. ISSN 22128271. doi: 10.1016/j.procir.2015.07.044. URL https://linkinghub.elsevier.com/retrieve/pii/S2212827115008033.

[9] John Ahmet Erkoyuncu, Iñigo Fernández Del Amo, Michela Dalle Mura, Rajkumar Roy, and Gino Dini. Improving Efficiency of Industrial Maintenance with Context Aware Adaptive Authoring in Augmented Reality. *CIRP Annals*, 66(1):465–468, 2017. ISSN 00078506. doi: 10.1016/j.cirp.2017.04.006. URL https://linkinghub.elsevier.com/retrieve/pii/S0007850617300069.

[10] Nirit Gavish, Teresa Gutiérrez, Sabine Webel, Jorge Rodríguez, Matteo Peveri, Uli Bockholt, and Franco Tecchia. Evaluating virtual reality and augmented reality training for industrial maintenance and assembly tasks. *Interactive Learning Environments*, 23(6):778–798, November 2015. ISSN 1049-4820, 1744-5191. doi: 10.1080/10494820.2013.815221. URL http://www.tandfonline.com/doi/full/10.1080/10494820.2013.815221.

[11] Jens Grubert, Tobias Langlotz, Stefanie Zollmann, and Holger Regenbrecht. Towards Pervasive Augmented Reality: Context-Awareness in Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics*, 23(6):1706–1724, June 2017. ISSN 1077-2626. doi: 10.1109/TVCG.2016.2543720. URL http://ieeexplore.ieee.org/document/7435333/.

[12] Tanmay Gupta, Amita Kamath, Aniruddha Kembhavi, and Derek Hoiem. Towards General Purpose Vision Systems: An End-to-End Task-Agnostic Vision-Language Architecture. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16399–16409, June 2022. URL https://openaccess.thecvf.com/

content/CVPR2022/html/Gupta_Towards_General_Purpose_Vision_Systems_An_End-to-End_Task-Agnostic_Vision-Language_Architecture_CVPR_2022_paper.

[13] Steven J. Henderson and Steven K. Feiner. Augmented reality in the psychomotor phase of a procedural task. In *10th IEEE International Symposium on Mixed and Augmented Reality*, pages 191–200. IEEE, October 2011. ISBN 978-1-4577-2185-4 978-1-4577-2183-0 978-1-4577-2184-7. doi: 10.1109/ISMAR.2011.6092386. URL http://ieeexplore.ieee.org/document/6162888/.

[14] Gaoping Huang, Xun Qian, Tianyi Wang, Fagun Patel, Maitreya Sreeram, Yuanzhi Cao, Karthik Ramani, and Alexander J. Quinn. AdapTutAR: An Adaptive Tutoring System for Machine Tasks in Augmented Reality. In *CHI Conference on Human Factors in Computing Systems*, pages 1–15, Yokohama Japan, May 2021. ACM. ISBN 978-1-4503-8096-6. doi: 10.1145/3411764.3445283. URL https://dl.acm.org/doi/10.1145/3411764.3445283.

[15] Xiangyu Li, Yonghong Hou, Pichao Wang, Zhimin Gao, Mingliang Xu, and Wanqing Li. Trear: Transformer-Based RGB-D Egocentric Action Recognition. *IEEE Transactions on Cognitive and Developmental Systems*, 14(1):246–252, March 2022. ISSN 2379-8920, 2379-8939. doi: 10.1109/TCDS.2020.3048883. URL https://ieeexplore.ieee.org/document/9312201/.

[16] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection. 2023. doi: 10.48550/ARXIV.2303.05499. URL https://arxiv.org/abs/2303.05499. Publisher: arXiv Version Number: 4.

[17] Ziyi Liu, Zhengzhe Zhu, Enze Jiang, Feichi Huang, Ana M Villanueva, Xun Qian, Tianyi

Wang, and Karthik Ramani. InstruMentAR: Auto-Generation of Augmented Reality Tutorials for Operating Digital Instruments Through Recording Embodied Demonstration. In *CHI Conference on Human Factors in Computing Systems*, pages 1–17, Hamburg Germany, April 2023. ACM. ISBN 978-1-4503-9421-5. doi: 10.1145/3544548.3581442. URL https://dl.acm.org/doi/10.1145/3544548.3581442.

[18] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider,

Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben

Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. GPT-4 Technical Report. 2023. doi: 10.48550/ARXIV.2303.08774. URL https://arxiv.org/abs/2303.08774. Publisher: arXiv Version Number: 4.

[19] Riccardo Palmarini, John Ahmet Erkoyuncu, Rajkumar Roy, and Hosein Torabmostaedi. A systematic review of augmented reality applications in maintenance. *Robotics and Computer-Integrated Manufacturing*, 49:215–228, February 2018. ISSN 07365845. doi: 10.1016/j.rcim.2017.06.002. URL https://linkinghub.elsevier.com/retrieve/pii/S0736584517300686.

[20] Kyeong-Beom Park, Minseok Kim, Sung Ho Choi, and Jae Yeol Lee. Deep learning-based smart task assistance in wearable augmented reality. *Robotics and Computer-Integrated Manufacturing*, 63:101887, June 2020. ISSN 07365845. doi: 10.1016/j.rcim.2019.101887. URL https://linkinghub.elsevier.com/retrieve/pii/S0736584518306240.

[21] Sara Romano, Enricoandrea Laviola, Michele Gattullo, Michele Fiorentino, and Antonio Emmanuele Uva. More Arrows in the Quiver: Investigating the Use of Auxiliary Models to Localize In-View Components with Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics*, 29(11):4483–4493, November 2023. ISSN 1077-2626, 1941-0506, 2160-9306. doi: 10.1109/TVCG.2023.3320229. URL https://ieeexplore.ieee.org/document/10268999/.

[22] Dhaval Sahija. Critical review of machine learning integration with augmented reality

for discrete manufacturing. *International Journal for Innovative Research in Multidisciplinary Field*, 7:118–126, December 2021. doi: 10.2015/IJIRMF.2455.0620/202112017.

[23] Chandan K. Sahu, Crystal Young, and Rahul Rai. Artificial intelligence (AI) in augmented reality (AR)-assisted manufacturing applications: a review. *International Journal of Production Research*, 59(16):4903–4959, August 2021. ISSN 0020-7543, 1366-588X. doi: 10.1080/00207543.2020.1859636. URL https://www.tandfonline.com/doi/full/10.1080/00207543.2020.1859636.

[24] Aditya Sharma, Luke Yoffe, and Tobias Höllerer. OCTO+: A Suite for Automatic Open-Vocabulary Object Placement in Mixed Reality. In *IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*, pages 157–165, Los Angeles, CA, USA, January 2024. IEEE. ISBN 9798350372021. doi: 10.1109/AIxVR59861.2024.00028. URL https://ieeexplore.ieee.org/document/10445543/.

[25] Ana Stanescu, Peter Mohr, Dieter Schmalstieg, and Denis Kalkofen. Model-Free Authoring by Demonstration of Assembly Instructions in Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3821–3831, November 2022. ISSN 1077-2626, 1941-0506, 2160-9306. doi: 10.1109/TVCG.2022.3203104. URL https://ieeexplore.ieee.org/document/9874259/.

[26] Wenjin Tao, Ze-Hao Lai, Ming C. Leu, Zhaozheng Yin, and Ruwen Qin. A self-aware and active-guiding training & assistant system for worker-centered intelligent manufacturing. *Manufacturing Letters*, 21:45–49, August 2019. ISSN 22138463. doi: 10.1016/j.mfglet.2019.08.003. URL https://linkinghub.elsevier.com/retrieve/pii/S2213846319300112.

[27] Dorin Ungureanu, Federica Bogo, Silvano Galliani, Pooja Sama, Xin Duan, Casey

Meekhof, Jan Stühmer, Thomas J. Cashman, Bugra Tekin, Johannes L. Schönberger, Pawel Olszta, and Marc Pollefeys. HoloLens 2 Research Mode as a Tool for Computer Vision Research. 2020. doi: 10.48550/ARXIV.2008.11239. URL https://arxiv.org/abs/2008.11239.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[29] Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, Jiazheng Xu, Bin Xu, Juanzi Li, Yuxiao Dong, Ming Ding, and Jie Tang. CogVLM: Visual Expert for Pretrained Language Models. 2023. doi: 10.48550/ARXIV.2311.03079. URL https://arxiv.org/abs/2311.03079.

[30] Giles Westerfield, Antonija Mitrovic, and Mark Billinghurst. Intelligent Augmented Reality Training for Motherboard Assembly. *International Journal of Artificial Intelligence in Education*, 25(1):157–172, March 2015. ISSN 1560-4292, 1560-4306. doi: 10.1007/s40593-014-0032-x. URL http://link.springer.com/10.1007/s40593-014-0032-x.

[31] Haotian Zhang, Pengchuan Zhang, Xiaowei Hu, Yen-Chun Chen, Liunian Li, Xiyang Dai, Lijuan Wang, Lu Yuan, Jenq-Neng Hwang, and Jianfeng Gao. GLIPv2: Unifying Localization and Vision-Language Understanding. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 36067–36080. Curran Asso-

ciates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ea370419760b421ce12e3082eb2ae1a8-Paper-Conference.pdf.

[32] Vasilios Zogopoulos, Eva Geurts, Dorothy Gors, and Steven Kauffmann. Authoring Tool for Automatic Generation of Augmented Reality Instruction Sequence for Manual Operations. *Procedia CIRP*, 106:84–89, 2022. ISSN 22128271. doi: 10.1016/j.procir.2022.02.159. URL https://linkinghub.elsevier.com/retrieve/pii/S2212827122001603.

Code and Attributions

Code on GitHub: https://github.com/DstoverVT/General-Purpose-AR-Task-Guidance

Figure Icon Attributions:

1. Computer Vectors by Vecteezy (in Figure 1.1)
2. Coffee icons created by srip - Flaticon (in Figure 1.1)
3. Augmented reality icons created by Futuer - Flaticon (in Figure 1.1)
4. Finger icons created by Smashicons - Flaticon (in Figure 1.1)
5. Desk icons created by Payungkead - Flaticon (in Figure 1.1)
6. Photo by Karsten Winegeart on Unsplash (in Figure 2.1)
7. Illustration Vectors by Vecteezy (in Figure 3.8)
8. Video Icon Vectors by Vecteezy (in Figure 3.8)

Appendices

Appendix A

User Study Materials

A.1 Consent Form

VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY

Consent to Take Part in a Research Study

Title of research study: ***24-288 Usable Task Guidance in Augmented Reality using Vision-Language Models***

Principal Investigator: *Dr. Doug A. Bowman – Contact: (540) 231-2058* ***or*** *dbowman@vt.edu*
Other study contact(s): *Daniel Stover - Contact: (703) 431-8190* ***or*** *danielstover@vt.edu*

Key Information:

The following is a short summary of this study to help you decide whether or not to be a part of this study. More detailed information is listed later on in this form.

We invite you to take part in a research study about Augmented/Virtual Reality (AR/VR) user interfaces. The research will take around 60-90 minutes to complete. The participation is entirely voluntary.

What should I know about being in a research study?

- Someone will explain this research study to you
- Whether or not you take part is up to you
- You can agree to take part now and later change your mind
- Your decision will not be held against you
- You can ask all the questions you want before you decide.

Consent to Take Part in a Research Study

What should I know about this research study?

This research project is intended to evaluate a proposed system that provides guidance throughout typical daily tasks. This research will help us understand the usability of our proposed system for task guidance. You will be asked to wear a Microsoft HoloLens 2 AR headset and interact with virtual content displayed in the headset while performing real-world tasks. Afterwards, you will be asked to fill out questionnaires about your experience (More detailed information can be found under **"What are the experimental procedures?"**). The research will take around 60-90 minutes to complete. There are risks that you might feel dizziness or nausea (More detailed information can be found under **"Is there any way being in this study could be bad for me? (Detailed Risks)"**). There are no significant benefits that individual subjects might experience from participating in this research. We cannot promise any benefits to others from your taking part in this research. However, the subjects will get to experience novel Augmented/Virtual Reality technology. Potential benefits to others also include the AR/VR industry gaining a better understanding of usability issues of AR interfaces to support everyday uses. If you are a student at Virginia Tech, the decision whether to participate or not will have no effect on your academic performance or relationship with Virginia Tech.

Detailed Information:

The following is more detailed information about this study in addition to the information listed above.

Inclusion Criteria: The study population for this study includes people who: 1) are at least 18 years old, 2) are English speakers, 3) have perfect or corrected normal vision (contact lenses and glasses are supported).

Who can I talk to?

If you have questions, concerns, or complaints, or think the research has hurt you, please send an email to Daniel Stover at *danielstover@vt.edu* or talk to the research team directly at *Room 3215, Gilbert Place, 220 Gilbert St, Blacksburg, VA 24060*

This research has been reviewed and is approved by the Virginia Tech Institutional Review Board (IRB). You may communicate with them at 540-231-3732 or irb@vt.edu if:

- You have questions about your rights as a research subject
- Your questions, concerns, or complaints are not being answered by the research team
- You cannot reach the research team
- You want to talk to someone besides the research team to provide feedback about this research

How many people will be studied?

We plan to include about 40 people in this research study.

What happens if I say yes, I want to be in this research?

You will become a participant in this research and will sign up for a time slot. You will be invited to try out the AR system we developed, and give us your feedback about a proposed AR task guidance system. The study will take place in Gilbert Place. The meeting location is in Gilbert 3215.

Consent to Take Part in a Research Study

What are the experimental procedures?

The experiment can be divided into five steps. In the first steps, participants will be welcomed upon arrival. We will introduce briefly what our study is about, the motivations behind it to the participants, and explain to participants any question they might have. Second, participants will be asked to read and sign the consent form. Third, they will be asked to fill out a pre-study questionnaire to collect demographic information and prior experience with Augmented/Virtual Reality. Fourth, participants will be asked to configure and evaluate guidance for real-world tasks using the AR system. During the study session, the AR system will record data about your tasks, including text and external pictures. Fifth, participants will be verbally asked to answer post-study questionnaires, which will be written down. The whole experiment will take roughly 60 to 90 minutes.

What is the total duration of this study?

The duration of an individual subject's participation will be 60-90 minutes.

What happens if I say yes, but I change my mind later?

You can leave the research at any time, for any reason, and it will not be held against you.
All the data collected to the point of withdrawal will be properly discarded and destroyed immediately.

Is there any way being in this study could be bad for me? (Detailed Risks)

Using AR technology can produce symptoms of sickness or discomfort in some users. These symptoms are usually mild, and may include dizziness, nausea, eye strain, headache, or disorientation. During tasks involving physical movement, there is also some risk that you could collide with obstacles in the physical environment. Each participant will be accompanied by an experimenter alongside as a safety guarantee.

What happens to the information collected for the research?

We will make every effort to limit the use and disclosure of your personal information, including survey responses, diaries, and interview notes, only to people who have a need to review this information. We cannot promise complete confidentiality. Organizations that may inspect and copy your information include the IRB, Human Research Protection Program, and other authorized representatives of Virginia Tech.

If identifiers are removed from your private information or samples that are collected during this research, that information or those samples could be used for future research studies or distributed to another investigator for future research studies without your additional informed consent.

The results of this research study may be presented in summary form at conferences, in presentations, reports to the sponsor, academic papers, and as part of a thesis/dissertation.

What else do I need to know?

Your information and samples (both identifiable and de-identified) might be used to create products or to deliver services, including some that may be sold and/or make money for others. If this happens, there are no plans to tell you, or to pay you, or to give any compensation to you or your family.

Consent to Take Part in a Research Study

There are no significant benefits that individual subjects might experience from participating in this research. However, the subjects will get to experience novel Augmented/Virtual Reality technology.

Signature Block for Capable Adult

Your signature documents that you have read and understand the information outlined in this document and your permission to take part in this research. Please keep a digital signed copy of this form for your records.

Signature of subject	Date
Printed name of subject	
Signature of person obtaining consent	Date
Printed name of person obtaining consent	

A.2 Pre-Study Questionnaire

Gender

- ◯ Male
- ◯ Female
- ◯ Non-binary / Another gender
- ◯ Prefer not to say

Age

Occupation

Major / Area of Specialization (if student)

Please rate your current fatigue level:

- ◯ 1 (not tired at all)
- ◯ 2

- ◯ 3
- ◯ 4
- ◯ 5 (very tired)

Please rate your experience with Augmented Reality:

- ◯ 1 (not experienced at all)
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5 (very experienced)

Please rate your experience with Virtual Reality:

- ◯ 1 (not experienced at all)
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5 (very experienced)

How often have you tried Augmented Reality:

- ◯ Never
- ◯ 1-3 times
- ◯ 3-10 times
- ◯ More than 10 times

A.3 Post-Study Interview Questions

AR Task Guidance Post-Study Interview

Question 1: On a scale of 1-7, Did the proposed system provide guidance visuals that you expected for your input text instructions?

- **Quantify 1:** 1 (outputs not expected) – 7 (outputs were expected) =

If not, how did they differ from your expectations?
Response:

-

Question 2: Did the proposed system make it easy to get the output you expected?
(a). On a scale of 1-7, Given the system guidelines, how easy was it to write instructions to get an expected output?

- **Quantify 2.1:** 1 (very easy) – 7 (very hard) **=**

Response:

-

(b). On a scale of 1-7, Given the system guidelines, how easy was it to take good pictures to get an expected output?

- **Quantify 2.2:** 1 (very easy) – 7 (very hard) =

Response:

-

Question 3: On a scale of 1-7, when needing to go back and revise instructions and/or pictures after its output was incorrect, was it easy to know what to change?

- **Quantify 3.1:** 1 (very easy) – 7 (very hard) =

Response:

-

On a scale of 1-7, Did those changes make the output better?

- **Quantify 3.2:** 1 (output was not better) – 7 (output was better) =

Response:

-

Question 4: How could the proposed system and approach be improved?
Response:

-

Question 5: What are some example real-world applications for a proposed system such as the one you tried?
Response:

-

A.4 Participant Recruitment Email

Usable Augmented Reality Task Guidance using Vision-Language Models

Email Subject: Participants Needed for an Augmented/Virtual Reality Research User Study

The 3D Interaction (3DI) group is inviting you to participate in a research study with virtual reality (VR) and augmented reality (AR) technologies. Our team is investigating the best ways to interact with AR and VR, and this study compares various methods of interaction with virtual data and information. The study will make use of state-of-the-art AR/VR technologies.

Participants in the study will come to Gilbert Place 3215, 220 Gilbert St, Blacksburg, VA 24060. Participants will try out an Augmented Reality app that guides them through typical office tasks. The entire experimental session will take about 60-90 minutes. The participation is entirely voluntary.

Participants need to be English speakers, at least 18 years old, have normal vision (contacts or glasses are allowed). Participants should not have any symptom or known recent exposure to COVID-19.

Participation will be on first come first serve basis. Participants' involvements in the study will not impact their grades or academic performance in any way.

For more information contact Daniel Stover at danielstover@vt.edu to schedule a session. The project is supervised by Dr. Doug A. Bowman in the Computer Science department.

This experiment is approved, as required, by the Virginia Tech Institutional Review Board under protocol #24-288.