AMMA: Adaptive Multimodal Assistants Through Automated State Tracking and User Model-Directed Guidance Planning

Jackie (Junrui) Yang*
Stanford University

Leping Qiu
University of Toronto

Emmanuel Angel Corona-Moreno
Stanford University

Louisa Shi
University of Washington

Hung Bui
Stanford University

Monica S. Lam
Stanford University

James A. Landay
Stanford University

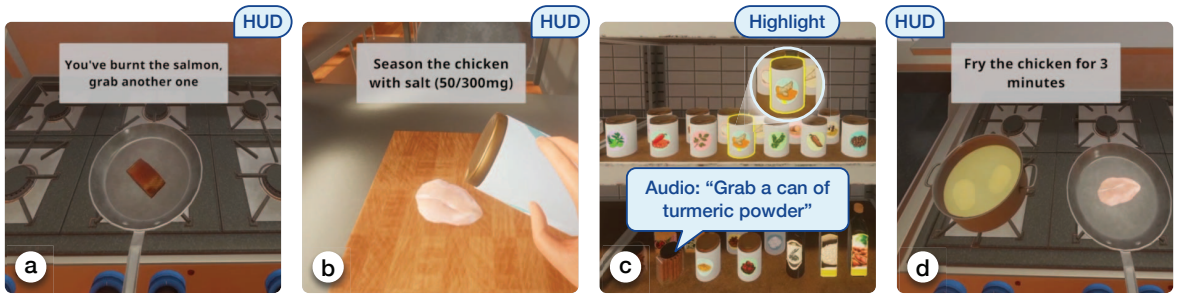

Figure 1: Adaptive Multimodal Assistants (AMMA) is an architecture for building guidance assistants that can adapt to users' progress, preferences, and capabilities. a) **Progress**: The user overcooked the fish fillet, which is no longer edible. Therefore AMMA asks the user to grab another one and continue; b) **Preferences**: The user turned their heads-up display on when seasoning previously, so AMMA uses it again for a similar step; c) **Preferences**: The user gathered ingredients fastest when highlighting and audio were on, therefore AMMA uses it again; d) **Capabilities**: AMMA estimates the duration for the user to perform different steps to optimize planning. In this case, AMMA asks the user to fry the chicken while boiling potatoes to reduce the completion time.

Abstract

Novel technologies such as augmented reality and computer perception lay the foundation for smart assistants that can guide us through real-world tasks, such as cooking or home repair. However, the nature of real-world interaction requires assistants that adapt to users' mistakes, environments, and communication preferences. We propose Adaptive Multimodal Assistants (AMMA), a software architecture for task guidance with generated adaptive interfaces from step-by-step instructions. This is achieved through 1) an automatically generated user action state tracker and 2) a guidance planner that leverages a continuously trained user model. The assistant also adjusts its guidance and communication delivery methods based on observed user performance as well as implicit and explicit user feedback. We demonstrated the viability of AMMA by building an adaptive cooking assistant running in a high-fidelity virtual reality-based simulator. A user study of the cooking assistant showed that AMMA can reduce the task completion time and the number of manual communication methods changes.

Keywords: Augmented reality, interface generation, smart assistant

Index Terms: Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Mixed / augmented reality; Human-centered computing—Human computer interaction (HCI)—Interactive systems and tools—User interface toolkits;

1 Introduction

Intelligent guidance systems can help people learn new tasks and achieve known tasks more efficiently [21]. They can be useful in everyday tasks like furniture assembly and cooking and in industrial settings such as debugging and repairing equipment and physical infrastructure. Recent years have seen advances in computer perception for augmented reality (AR) [11, 42, 40, 14, 2]. Novel interface technologies [10, 41, 22], such as AR or virtual reality (VR) can provide richer multimodal interfaces in situ. These new technologies are moving towards a future where AR systems understand the user's situation/preferences and can adaptively give users instructions.

However, the typical design of current guidance systems only supports rigid, step-by-step, predefined instructions [25, 7], which does not suffice to accommodate real-world situations. In the real world, the user might not always exactly follow the guidance, and the action might have a chance of failure. The best guidance for a user depends on what action the user is currently performing and the situation resulting from the previous actions. The guidance system needs to understand if users have made mistakes and help them recover from these mistakes. For example, a navigation application must adjust the route if the user fails to follow the directions properly.

Furthermore, current adaptive systems use user-agnostic algorithms to adjust their interfaces, and they do not explicitly consider the varied abilities of users (walking speed, reaction time, etc.) and the difference in preference for guidance modalities. For instance, a common approach [21] is to adjust the level of detail of instructions based on users' reaction time. However, such an approach may provide an expert with adaptively detailed information when they respond slowly to instructions because of their slow walking speed.

In this paper, we propose Adaptive Multimodal Assistants (AMMA), a multimodal adaptive guidance architecture. AMMA allows developers to build a guidance assistant that adapts to the user's progress, preferences, and capabilities to generate the most efficient instructions in the most effective communication modality. AMMA achieved this by: 1) a state tracker that uses a given computer perception algorithm to understand users' current task status and proposes possible steps to reach their goals; 2) a guidance planner that learns from a user's behavior while they are performing the task and observes their feedback through modality changes. By taking in a specification of a certain activity domain, AMMA facilitates the development of a truly adaptive guidance assistant for real-world complex tasks such as cooking and furniture assembly. These tasks are complex in that there is more than one path to achieve the task

*e-mail: jackiey@stanford.edu

2642-5254/24/$31.00 ©2024 IEEE
DOI 10.1109/VR58804.2024.00108

and more than one modality to inform the user of the guidance.

AMMA differentiate from previous adaptive systems in its guidance planner, which uses user modeling as an intermediary step to better fit users' diverse requirements. Our guidance planner observes the user's behavior but does not directly map that to guidance based on rules. A user performing a step slower can be caused by having trouble A) understanding the instructions, B) walking to the area, or C) transitioning between the actions. Compared to using a one-size-fits-all solution, AMMA would learn what is causing this step to be slow by modeling the user's past behavior and making appropriate adjustments based on that user model. In the end, the system will perform differently for different users: A) use a different modality for this instruction; B) optimize instructions prioritizing walking distance; C) optimize instructions grouping the same type of actions.

We demonstrated the capacity of AMMA architecture by building an augmented reality cooking assistant in a simulated virtual reality environment (similar to what has been proposed by Lacoche et al. [26]). We conducted an evaluation user study that compares it with a baseline assistant with a pre-defined guidance policy. Our evaluation results show that Adaptive Multimodal Assistants significantly reduces the task completion time and the number of times users need to manually adjust the communication modalities. 10/12 participants in our study preferred our adaptive system over the baseline system.

Our contributions are:

1. AMMA, an architecture for building intelligent guidance assistants that can adapt to users' progress, preferences, and capabilities. It includes a state tracker that tracks users' actions, handles action orders, and helps users correct mistakes; a guidance planner that generates the guidance policy using a parameterized user model trained from user interactions.
2. The implementation of a cooking assistant built with AMMA demonstrates the architecture's capacity.
3. An evaluation of the cooking assistant in a simulated environment that illustrated the adaptive assistant substantially improved task performance, reduced the cost of manual UI adjustments, and was preferred by the users.

In the following sections, we first outline related work in intelligent guidance systems and adaptive interfaces. We then present the architecture and algorithms of Adaptive Multimodal Assistants. After that, we describe the implementation of Adaptive Multimodal Assistants in a cooking assistant and the evaluation of the assistant. Finally, we discuss the limitations of our work and future directions.

2 Related Work

There are three different categories of research related to our adaptive multimodal assistant: adaptive user interfaces, state tracking in guidance systems, and multimodal interfaces for guidance systems.

2.1 Adaptive Interactive Systems

Many adaptive behaviors are implemented for guidance systems. Unlike most apps where the user instructs the machine for information, guidance systems take the initiative most of the time and guide the user through a task by giving the user instructions.

Prior research has studied adaptive guidance systems that can change their behavior according to the user's interactions. Some work makes adaptations based on developer-specified policies. For example, AdapTutAR [21] uses response time to adjust the level of detail (LOD) of tutoring content for guiding equipment usage. Other systems use developer-provided information with optimization or machine learning techniques. For example, Lindlbauer et al. [29] uses cognitive load and developer-provided task load cues to adapt the LOD of relevant notifications from other applications for minimizing distractions. Another example is the work in ScalAR [37], which uses developer-provided training data to adapt AR interfaces for showing AR content in the users' own environments. These systems either require heuristic rules from the developer or may not be able to adapt to the user's varying capabilities and preferences.

Another strategy is to rely on reinforcement learning to train the system [5, 33, 3]. For example, Mu et al. [33] developed a tutoring system that uses a multi-armed bandit algorithm to adapt to individual students. These systems are highly adaptive to users, yet they require many time-consuming training sessions.

Both strategies don't have the user model as a separate component but react to observations of the users based on task-specific policies. The policy requires more work from developers and can't accommodate every user. The reinforcement learning solution requires many users or an extended period of use to achieve adaptation.

In domains other than guidance systems, prior research on user interface (UI) transformation has explored combining user modeling with algorithms to generate interfaces that adapt to a specific user. For example, SUPPLE [16, 17] collects user traces and finds the most efficient interface layout for the user. Li et al. [28] models the user's notification preferences based on how they interact with notifications and uses the model to inform the delivery of future notifications. These systems learn user preferences based on past behavior to adapt UI. In contrast, in a guidance system, the user's past behavior is often not the most efficient way to complete the task we are guiding them. Different guidance tasks also involve different constraints and dependencies, so it is hard to generalize a user-trace-based model for a diverse list of supported tasks. Therefore, AMMA is built upon well-established techniques for user modeling [8, 31]. In this style of adaptation, the developer provides a user model that *applies to all tasks in a high-level activity* (see definition in Section 3.1) and AMMA personalizes the model for the current user. AMMA can then use the personalized model and a simulator based on the state tracker to generate a personalized policy for presenting instructions. Our architecture combines the personalization of a user model and the unique task characteristics embedded in the simulator.

Another trend of research is on affective computing systems. These systems change the software behavior according to the user's affective state. Chao et al. [9] changes a virtual tutor agent's facial expression based on the user's affective state. Yannakakis [43] experiments with changing a game's difficulty level based on the user's physiological state and expressed emotions. Affective Music Player [38] plays different music to help users achieve a target mood. AMMA differs from them because it focuses on the efficiency of guidance rather than improving or changing affective states.

2.2 State Tracking in Guidance Systems

AMMA can track the relevant physical objects (e.g., food products in a cooking task or furniture components in an assembly task) and the corresponding steps where they are used. It can also handle error recovery, ambiguous actions that could be part of similar steps, and accept users' out-of-order steps (see Section 4.2 for more details).

There are existing guidance systems that do not use automatic state tracking. TutoriVR [25] augments VR video tutorials for VR sketching tasks. Cao et al. [7] propose an AR tutoring system using AR avatars to guide users to operate a complex machine. These systems rely on manually reporting the user's current step, which introduces an additional burden to the user that makes it hard to observe and learn from the user's behavior at scale.

Some other guidance systems contain automatic state tracking based on pre-provided state graphs. Zauner et al. [44] describes an AR assembly guidance system using a state diagram from an authoring tool. Miyawaki and Sano [32] propose a cooking agent that uses a recipe as a state model and sensors to keep track of state transitions. Cooking navi [19] uses a recipe database as the state graph. However, all these systems expect users to always follow the steps and can't handle the users' mistakes or assist the user in recovering from them. AMMA, in contrast, have a unique state tracker that can understand the user's mistakes through predefined

actions and can compute the possible steps that the user can take to recover from them (see Section 3.1 and Section 4.2). This design allows AMMA to work with complex tasks, i.e., more than one path to achieve the task and more than one modality to inform the user of the guidance. In comparison, many prior projects mainly target simpler tasks where there is only a specific sequence of actions.

Another style of state tracking relies on machine perception to directly translate perceived signals to states. Hamada et al. [19] and DuploTrack [18] use computer vision techniques to generate the task state. This approach is demonstrated with the assembly process of LEGO bricks for real-time guidance but is unsuitable for complex activities, such as cooking and furniture assembly. The state cannot be easily inferred from a single image in these activities. AMMA uses a more general approach, i.e., infer the user's situation by tracking the sequence of *actions* that the user has taken.

2.3 Multimodal Interfaces for Guidance Systems

Multiple approaches have been proposed to guide users through complex tasks in real-world situations. Prior research has been focused on handling multimodal input [4, 23, 41, 36]. Multimodal input allows users to convey their intentions efficiently and naturally. For guidance systems, in contrast, output and feedback to the user are more important as users are primarily following instructions.

For multimodal output, some researchers compare the performance of different modalities [6, 30, 13] for guiding users through tasks. For example, Liu [30] compared how auditory, visual, and multimodal displays affected drivers' performance in different driving conditions. Cooking Navi [19] provides text, videos, and audio simultaneously to better guide users through cooking procedures. Although these multimodal methods are effective, users have different preferences in different scenarios. Therefore, adapting the output methods is a crucial problem to solve if we are to further enhance the user experience and performance of guidance systems. Unlike prior work, we aim to provide an adaptive assistant that can automatically adjust to an individual's preference and performance.

In our work, we use common multimodal output methods, including a heads-up display (with text instructions or object highlighting), a large monitor (with video or text instructions), and audio instructions. Unlike the prior work, AMMA also observes user behavior and feedback to adapt the modalities to the user's current situation.

3 Overview of AMMA

AMMA comprises three components (see Fig. 2): 1) **Interaction support** observes the user's actions, monitors the user's feedback, and provides guidance; 2) **Tracking and planning** contains the state tracker and guidance planner; 3) **Activity specification** supports a specific activity, including the user model template, the multimodal feedback UIs, and the specifications of tasks, steps, and actions.

3.1 Definitions of Activity, Task, Object, Step, and Action

We define terms formally below. The overall goal of a guidance assistant system is to aid a user in performing a high-level **activity**, such as "cooking dinner" or "assembling a bed." An activity consists of several **tasks**, such as "cooking chicken cacciatore" as an entree for dinner. Like commonly used text-based instructions, we break a task into **steps**. In the real world, steps are usually applied to certain physical **objects**. For example, a component when assembling furniture or an ingredient when cooking. **Actions** are individually observable atomic behaviors that can be combined to accomplish a task. For example, a step such as "transfer the cooked salmon from the pan to the dish" can be broken down into actions, such as "remove the salmon from the pan" and "add the salmon to a dish".

3.2 Interaction Support

The interaction support component needs to observe users' actions and feedback, and provide multimodal guidance to the user.

Assistants built with AMMA start by observing users' actions ❶ and taking in users' feedback ❷. AMMA assistant observes actions, including the user's position, their current action, and the object that they are working with. It also uses explicit user feedback, such as the adjustment of the modality settings. In addition, AMMA assistant combines the data from multiple user actions to improve its modality choices. For example, it obtains implicit user feedback, such as a user's reaction time to a type of guidance modality. Both types of feedback help the system understand whether the used guidance modality is the best fit for the user and the current step.

The AMMA-based system generates guidance and a choice of modality ❸. The set of different modalities/UIs is used to render instructions to guide the user in completing the task.

3.3 Tracking and Planning

The core of AMMA contains the state tracker and the guidance planner: A state tracker is a sub-system that tracks the current state from the user's actions and produces the valid actions and steps at the current state. A guidance planner is a sub-system that accepts the user's feedback and selects the most appropriate step and modality to guide the user. We will dive into their technical details in Section 4.

The observed actions are fed into a state tracker ❹. The state tracker monitors all the objects related to these actions. It also tracks the possible steps of the object and the status of these steps (the actions that have been completed and need to be completed).

The guidance planner ❺ uses the objects and possible steps from the state tracker ❹ to select the best step and the best modality to provide guidance. It contains three modules: the user model ❻, the simulator ❼, and the policy algorithm ❽. The assistant gathers feedback from the user to train the user model ❻. The user model summarizes our understanding of the user behavior. It can, based on the user: 1) predict how long a certain step will take, 2) offer the best guidance modality of a step. It takes in the user's current position in the task and the type of step they are on to make a prediction. The simulator ❼ is similar to the state tracker ❹: Both the simulator and the state tracker keep track of a list of objects and their current step. The main difference is that the simulator operates on a list of virtual objects with no physical representations. The simulator also accepts available steps for the objects it keeps track of. Another difference is that the simulator only simulates at the step level instead of at the action level, as the guidance given to the users is at the step level. This allows AMMA's policy algorithm ❽ to simulate more quickly while reducing the workload on the policy algorithm.

3.4 Activity Specification

AMMA has an activity specification component to account for the domain-specific information related to the activity. It includes: 1) **Action specification**: action types (e.g., make one cut with a knife or screw in a screw) and how to observe them (e.g., computer vision or embedded sensors); 2) **Step specification**: step types (e.g., cut food into n pieces or fix wooden legs to a tabletop) and the actions they are composed of (e.g., $n-1$ single cut or four screws screwed into the legs); 3) **Task specification**: a set of tasks (e.g., make a salad or assemble a dining desk) and their instructions described in steps; 4) **Multimodal feedback UIs**: a set of guidance interfaces that can convey a step's instructions and the task's current status. 5) **User Model Template**: a differentiable model of the user's action time, modality preference, and default parameter values.

For *action specification*, AMMA needs a list of actions and a computer perception implementation to recognize the actions. For example, for furniture assembly, the actions might be: gather components, screw in a screw, insert one component into another, etc. We can imagine that in a cooking task, a smart scale can be used for accurately measuring how much seasoning has been added, and in a map navigation scenario, a GPS can be used to determine if the user has made a turn/made a lane change successfully. AMMA also needs

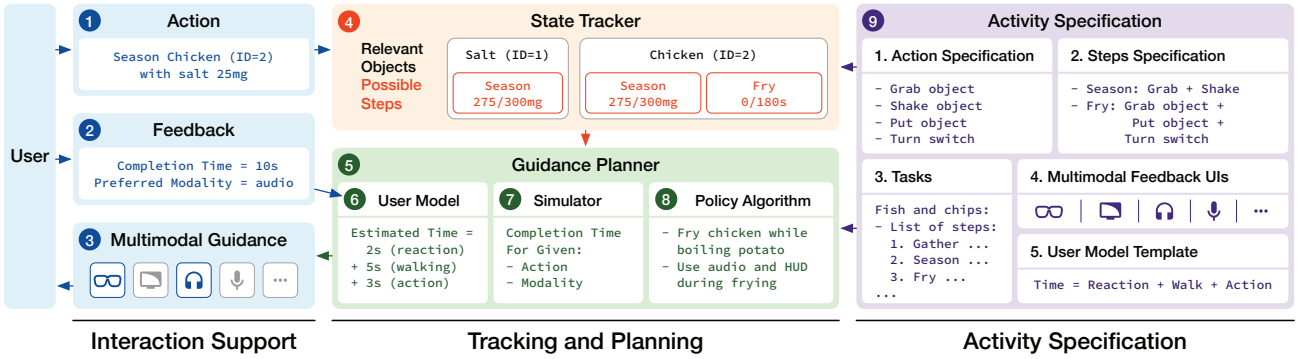

Figure 2: AMMA Overview: AMMA's interaction support renders the feedback interface and gives instructions to the user while observing the user's actions and feedback. AMMA's state tracker keeps track of all the relevant objects in the scene and their steps according to the task. It also accepts the user's actions and updates the corresponding object's step status. AMMA's guidance planner processes the current state from the state tracker through its policy algorithm, simulator, and user model. Then, the policy algorithm selects the best step and modality to present the associated guidance. Both AMMA's state tracker and guidance planner are generated directly from activity specifications provided by the developer, including tasks, steps, and action specifications.

to know whether the action is **harmless**, **reversible** or **irreversible**. The full definition and rationale will be given in Section 4.2. For example, touching an object is considered *harmless*, adding a solid object to a dry container is *reversible* (you can simply remove it), but adding a liquid to a container that already holds liquid is not, as you cannot pull one component out of a mixture. In the case where an action involves adding another object, we distinguish between whether it remains a separate *component* or it undergoes full *integration* into the resulting object (see scenario 6 in Section 4.2).

For *step specification*, AMMA expects step definitions by the actions they are composed of. For example, for frying shrimp for 60 seconds, we expect the user to add shrimp to a pan, heat the pan for 1 second 60 times, and then remove the shrimp from the pan. Similarly, for attaching legs to a tabletop, the actions can be gathering a leg of the table and nailing the leg onto the bottom of the tabletop. Note that sometimes an action in a certain step may require not only a certain type of object but also the state that the object is at. For example, when preparing for surgery, the surgeon needs to grab a scalpel, and the scalpel needs to be at the step "cleaned".

For *task specification*, AMMA needs a list of *tasks* comprising a sequence of steps. For example, it is a list of recipes in the case of cooking and a list of instructions in the case of furniture assembly.

An application built with AMMA also needs to specify a set of *multimodal feedback UIs*. These UIs can render the step provided by the guidance planner as an instruction to the user. For each modality (headphones, monitors, smart watches, or HUDs), the application can provide multiple choices of UIs, and the guidance planner 5 will select the appropriate one for the user.

Finally, the application needs to specify a *user model template*, which contains a differentiable model (a formula) to produce the predicted task completion time, error rate, and the user's preferred modality from the next step that the system plans to instruct.

Both the simulator 7 and the state tracker 4 are automatically generated from activity specifications 9.

4 State Tracker and Guidance Planner Design

We detail the design of the state tracker and the guidance planner.

4.1 Design Goals

We want AMMA to capture the user's current task state and provide the most appropriate guidance according to their preferences and prior behavior. Our design goals are:

Adaptability to Progress: AMMA should track users' progress, detect users' mistakes, and generate the possible next steps.

Adaptability to Preferences: AMMA should learn the user's communication preferences and guide users in the preferred modality.

Adaptability to Capability: AMMA should adapt to the user's capabilities based on prior observed behavior and offer instructions that will maximize the user's performance.

4.2 State Tracker Design

Our proposed state tracker addresses **Adaptability to Progress**. It tracks all the objects relevant to the tasks, the steps they are at, and the status of these steps. The state is formally defined as the list of objects and their steps. Therefore, state tracking consists of the following tasks: 1) building step tracking graph; 2) tracking objects through steps; 3) tracking state as a set of object and their steps.

4.2.1 Step Tracking Graph

Our state tracker needs a step-tracking graph to understand what steps have been taken to an object and the possible steps in the future. For example, a task like cooking a fried chicken dinner with a list of steps is shown in Fig. 3. An AMMA-based system will build the step-tracking graph based on the provided steps.

The step-tracking graph is a directed graph, where the steps are the nodes, and the directed edges connect step A to B if B depends on A; e.g., step 4 (S4) depends on S3 because it's working on the same potato. To reduce repeated work when mistakes are made, we define three types of dependencies: *continuation dependency*, *component dependency*, and *integration dependency*. **Continuation dependency** means that the current step is a continuation of another step working on the same object, e.g., S2 is a continuation of S1 because they are both working on the same object. **Component dependency** means that the current step requires an object from another step as a *component*. We refer to the object that the component belongs to as its **parent object**. **Component objects** are still being tracked as a normal object, but they can no longer be used as a component dependency of another step. However, if its *parent object* is destroyed, the *component* object will be **released** and can still be used to complete another step with *component dependency*. For example, S5 is a component dependency of S14 because the potato wedges can still be reused if the chef puts ketchup instead of mayonnaise on the plate. **Integration dependency** means that the current step requires an object from another step to be permanently bound to the current object. For example, S1 is an integration dependency of S5 because the salt is permanently bound to the potato.

To handle ambiguity, nodes representing identical steps that depend on the same nodes are merged as one; the merged nodes inherit all the outgoing dependencies; the merge is performed recursively to

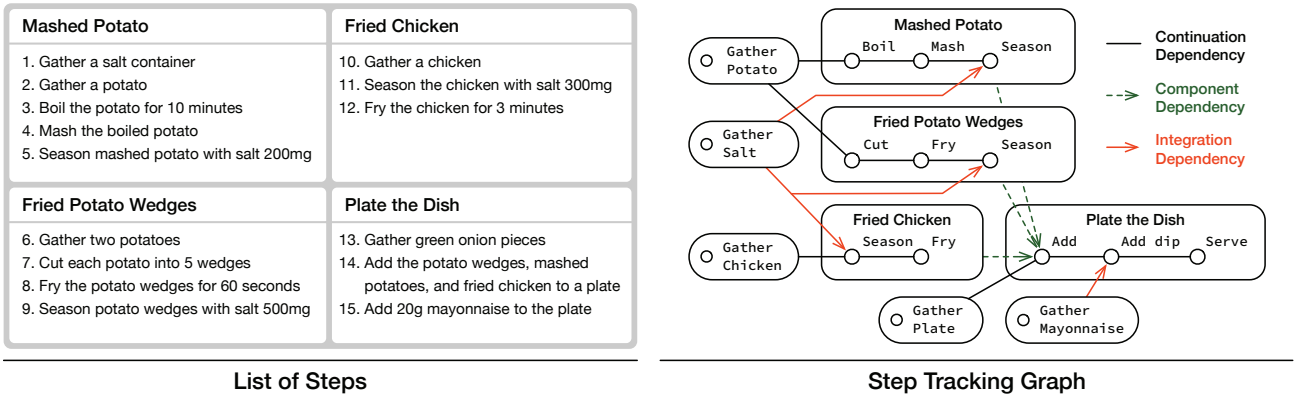

Figure 3: State tracking of AMMA converts a list of steps (left) to a step tracking graph (right) by 1) merging similar steps, such as step 2 and step 6, together, and 2) identifying continuation, component, and integration dependencies. This allows AMMA's state tracker to track the user's normal, irrelevant, and out-of-order actions. It can also show the user how to recover from mistakes.

find all ambiguous steps. In the case of the above recipe, we can see that S2 and S6 are both grabbing a potato and depend on no other steps. We merge them and make step "boil the potato" and step "cut the potato into five wedges each" depending on the merged node.

It should be noted that the step-tracking graph is naturally acyclic. Any topological sort of graph is a legal execution. We say a step is *valid* if all the steps they depend on have been performed.

4.2.2 Tracking Object Through Steps

Perfectly, the assistant only needs to read out the step-by-step instructions to the user. But in reality, the user may fail to follow the directions precisely. A guidance assistant must handle six scenarios:

Scenario 1 the user follows the instructions correctly;
Scenario 2 the user performs an irrelevant action (e.g., picking up an ingredient that is not needed for a recipe);
Scenario 3 the user performs an action out of order (e.g., for a fish and chips recipe with a cooking order of fish, chips, and then sauce, and the user cooks the sauce first);
Scenario 4 the user performs an ambiguous action (e.g., the user grabbed a potato, but it can be used in two different steps – one is to deep fry the potato into fries, and the other is to make mashed potatoes);
Scenario 5 the user makes a recoverable mistake (e.g., the user accidentally takes the potato chips off the plate and needs to add them back);
Scenario 6 the user makes an unrecoverable mistake (e.g., the user burns the fish, and it is no longer edible).

Scenario 1 is obvious: if the user performs an action corresponding to the instructed step, then the user follows the step correctly. For *scenario 2*, because we only observe actions relevant to the activity, if the user does something completely irrelevant such as using their phone while cooking a dish, it will not change anything in the state tracker. The difficulty lies in the user performing a relevant action that does not change the status of the object being acted upon. For example, we track if the user grabs an object in the ingredient-gathering step, and if the user simply touches a food item while seasoning it, it shouldn't change its state. Therefore, we introduce a property for actions called "**harmless**", which means if the state tracker observed this action and it is not expected as part of the current step, we can safely ignore it. For *scenario 3*, any valid step that the user performs, which is not the one instructed, is considered out-of-order and recorded as such. As usual, we will proceed the object to its next step if all the current step's actions have been performed. To distinguish between *scenarios 5 and 6*, we introduce a property for actions called **reversible**. The developer needs to identify reversible actions and their corresponding reverse actions to recover from errors. For example, adding an extra item or removing a required item is reversible by just removing or adding the item, respectively, but overcooking the fish is not reversible.

We use our step-tracking graph to minimize the extra work necessary when an irreversible mistake is made. For example, if the user drops the fried potato on the dish on the ground (see Fig. 3), we will only ask the user to redo the "Fried Potato Wedges" part of the tasks instead of asking the user to redo the entire dish.

4.2.3 State Tracking Through Step Tracking

The **current state** of a system built by AMMA is defined as a set of *objects*, their *current step*, and the *actions* for these steps.

At the start, the valid steps are those without previous steps and dependencies (start actions). In the example of furniture assembly, these are usually gathering steps where we gather all the components needed for the task. Once an action that is part of the start action is completed, the state tracker keeps track of all the steps dependent on the resulting object of the action. Note that to deal with ambiguity in actions, an object can be at different steps, and the tracker must anticipate multiple sets of possible actions at the same time. If all the actions required for the step are completed, the state tracker removes the current step from the object and adds the next step to the object. If an unexpected action is performed on an object, the state tracker will act accordingly depending on whether the action is *harmless* or *reversible*. If the action is irreversible, the state tracker will remove that step from the potential steps. If all possible steps are removed from an object, it probably means that something is wrong (e.g., the fish got overcooked) with the object, and the algorithm will **destroy** that object. When an object gets **destroyed**, the state tracker removes the object from tracking and asks the user to throw it away.

Normally, if an object is *destroyed*, we simply remove it from the state tracker. The guidance generation component of AMMA will guide the user to repeat the steps to re-create another object of the same type. Objects involving components are treated differently.

1. If a parent object is destroyed, all its components are *released* so that they can be used in another step.
2. If a component object is destroyed, a special action will be added to its parent object that requires a same-type object to be added before the current step of the parent can be completed.
3. Note that objects used in integration dependencies can be destroyed without incurring any extra work. For example, if the bottle of salt is dropped into a garbage bin, a potato that has been seasoned needs not be redone.

4.3 Guidance Planner Design

The guidance planner design satisfies the goal of providing **Adaptability to Preference** and **Adaptability to Capability**.

For **Adaptability to Capability**, we leverage the user model to predict user performance in completing a task [8]. For guidance, the policy algorithms have to adapt to the specific task, the progress made according to the state tracker, as well as the user model.

For **Adaptability to Preference**, AMMA should choose the best modality for the user while requiring as little feedback as possible. We adopted two methods to learn the user's preference: 1) observe a user's explicit feedback, e.g., the user says they want audio guidance at a certain step; 2) observe a user's implicit feedback, e.g., the user performs the step faster when using audio guidance.

The guidance planner needs to consider both the user's behavior through their feedback and the characteristics of the task to select the best step and best modality to guide the user. The planner understands the user's behavior through a user model trained with the user's feedback. It can also know the dependencies within the task through the generated simulator. Finally, it runs a policy algorithm combining both insights to generate guidance.

Note that user modeling is not new in guidance planner; it has been used for software heuristic evaluation [8, 31] or adapt interfaces to user ability at run time [17]. The way AMMA uses user modeling is novel as it is for guidance systems that have a long expanding task list (different dishes in a cooking app or different furniture in a furniture assembly app) rather than a specific task targeted by a regular app (writing a document for a word processing app).

First, the planner needs to train a personalized user model. The user model is initialized from the *user model template* provided in the *activity specification*. The guidance planner can then use implicit and explicit feedback to train the user model to adapt to the behaviors of the current user (see Section 3.3). The model can also output some observable intermediary results to reduce training time. This is so that AMMA can use a replay buffer and stochastic gradient descent to check these intermediary results and the final time estimation against real-world observations to train the user model faster. These observations include where the action may happen, how long it takes to take the *first* action in a step, etc. Under AMMA architecture, the application will use batched stochastic gradient descent [24] to train the user model to produce the best modality for the user. In practice, because implicit feedback relies on an accurate user model, we only use this style after the user model has been trained for a while. In the case of the cooking assistant app, we disabled the implicit feedback for the first cooking session (personalization session, see Section 6.1).

In the simulator, each step has a *failure rate* and a *time cost* as predicted by the user model. The failure rate is the probability that the user will fail to complete the step. If the failure happens, the simulator removes the object from its state. The time cost is the time it takes to complete the step. The simulator will also keep track of the total time. If there are time-sensitive steps, it will also check if the time is up and remove the object from the state tracker.

The policy algorithm uses the trained user model and a simulator to generate the user guidance. For AMMA, the policy algorithm can be implemented using reinforcement learning or a search algorithm. We used Q learning and a multi-layer perceptron (two 64-neuron hidden layers) Q function trained with the user model and the simulator for the reinforcement learning implementation. Usually, the initial training takes a long time (around 2 hours on an M1 Macbook Pro CPU for a recipe in the cooking assistant as defined as Section 5). Therefore, we pre-train an initial neural network for each task before its first usage. While the user is using the assistant, our guidance planner keeps training the network with the new user model. For updating the weights of the Q-network, AMMA keeps the replay buffer across different versions of the model so that AMMA has more data to train with. We prioritized sampling of data generated with the newer user models. Our search algorithm is a Monte-Carlo tree search [39] that uses a baseline policy to simulate the completion time at the maximum depth. The baseline policy is implemented by choosing the first step in the task that has not yet been completed. Although both algorithms can be useful depending on the application and the implementation, we found that the search algorithm can more quickly adapt to the constantly updating user model in our cooking assistant implementation. Therefore, our user study used the search algorithm for the guidance planner.

5 Example of AMMA-Supported Assistant: A Cooking Assistant

AMMA can support building guidance assistant systems for activities that require adaptability to preference, capability, and progress. Below, we provide an in-depth introduction to how to implement a cooking assistant using AMMA. Further discussions about how AMMA may support other activities are in Section 7.2. As AMMA is focused on state tracking and policy generation, we used a simulated environment to avoid the complication and inconsistency of computer vision action recognition algorithms for implementing the cooking assistant. We described our activity specification below.

5.1 Tasks, Steps, and Actions in Cooking

For cooking, we defined nine observable actions in the game: grab, add a product to a container, remove a product from a container, pour a product to a container, pour a product from a container, season condiment onto a product, cut product, start to cook product (including boil, fry, bake, microwave, deep-fry, and grill), stop cooking product, and blend products together in a container. In the case of these nine actions, only "grab" is *harmless*. "Add a product to a container" and "remove a product from a container" are reversible and they are the inverse of each other. "Pour a product into a container" is only reversible when there is no other liquid in the container, and it can be reversed with "pour a product from a container". From the recipes defined in the game, we have seven types of steps: gather ingredients, transfer product to the container, pour product into the container, season a product with condiments, cut a product into several pieces, cook a product in a particular fashion for several seconds, and blend products together in a container. For the tasks, we build a converter that can convert most of the 80 recipes supplied with the game into a list of steps in the format that AMMA supports.

5.2 User Modeling for Cooking Tasks

For user modeling, we observed that the amount of time a step required is usually composed of the following components (similarly to Keystroke-level model [8]): $T = M*P+W+I+A*N$. T is the total time spent on a step. M is the mental preparation time. P measures the impact on preparation time based on the user's preference for the modality. For example, if the instruction is confusing, it may take longer for the user to start the first action. The user's modality preference depends on the current step type and the presented modalities. W is the walking time from the user's location to where the first action can happen. I is the idle time when the user waits for the action to become available. In our case, it is only available for the *stop cooking product* action, accounting for the time for the user to wait until the product is cooked for the specified amount of seconds. A is a variable of the execution time of a single action in the step. This changes according to what step type is being executed. N is the number of actions needed. For "transfer", "season", and "pour", it is the number of ingredients. For "cut", it is the number of pieces minus one. For all other actions, $N = 1$.

For the walking time W, we use Fitts' law [15] to model this behavior [12]: $W = U\log_2(D/2W+1)$. U is a variable for the user's average walking speed. D is for the distance between the user's last position and where the user needs to be to execute the current step. We estimate the position of each step according to the

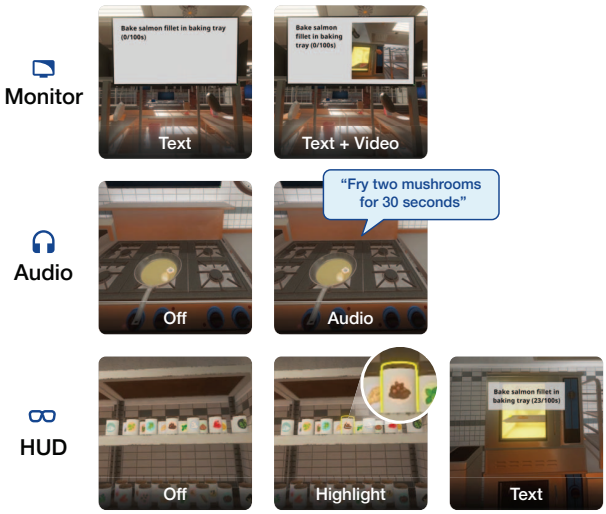

Figure 4: Guidance modalities implemented in our cooking assistant: monitor (text instructions only or text + video instructions), audio (on or off), HUD (highlight, text instructions, or off).

average positions of where that step has occurred before. W is the width of the target, and we empirically assigned it to be 1(m).

We separately model the success rate as follows: $S = S_b * S_a^N$. S is the success rate of a step. S_b is the base success rate of a step. S_a is the success rate of action in a step. N is the number of actions needed, similar to the definition above.

We assigned heuristic values to all the variables as initial values. We modeled all of these variables as "parameters" in the PyTorch library and used batched stochastic gradient descent [24] to fit the model at run time to estimate users' behavior.

5.3 Multimodal Guidance UIs

We provide multimodal guidance in three modalities: a heads-up display (HUD), a large monitor, and audio guidance. For the monitor, we can show only the text instructions or the next instruction and a video demonstrating the action. For audio guidance, we can speak the next instruction. For the HUD, we can either highlight the current object or display the next instruction. The audio guidance and the HUD can be disabled to reduce distraction. All interfaces are locally loaded as an asset bundle in the Unity plugin. The Python server communicates with the plugin to toggle the modalities, update the instructions, and play audio guidance.

5.4 Implementation

We used an existing VR game called *Cooking Simulator VR* for the simulated environment. [1] Our assistant implements the tracking, planning, and most activity specifications in a Python program *Cooking Assistant Server*. It also implements the interaction support and the multimodal guidance UI in a Unity plugin for Cooking Simulator VR *Cooking Assistant Plugin*. The two parts communicate through a TCP socket. The plugin receives user actions like grabbing and then passes onto the message queue. The server then responds according to its policy, which the plugin passes to the user through the modality and/or with the information instructed by the model. [2]

6 Evaluation

In a user study, we tested the adaptive multimodal guidance features of AMMA to examine if they help reduce task completion time and manual switching among modalities.

For any guidance system, a state tracker is required to decipher which step the user is at and which steps the user can take next. Therefore, we built a baseline condition with the same state tracker for comparison. The baseline condition uses a manual modality selection where the user can use the same commands to adjust the output modality used in AMMA. Similar to most software preference settings, it will not automatically change according to the user's current action. The baseline also uses a baseline guidance algorithm, where we choose the first available action in the recipe.

6.1 Procedure

The experimenter first demonstrates the system's functionality with a simple training recipe T (i.e., baked tuna). During the training, we explained how to grab, season, cut, and pour products in the game and change the guidance modality using voice commands. We then asked the user to try the same recipe T in VR. During this part of the training, we verbally guided the user on interacting with the system. After the first training recipe, we asked the user to cook another recipe P (i.e., baked tenderloin with fried carrot) to personalize our user model. We did not give the user verbal guidance for this recipe, and the user relied on the embedded guidance system. At this point, we asked the user to fill in a NASA-TLX questionnaire [20] to reduce any carry-over effects of the likely higher training + personalization task loads.

After the training + personalization phase, we asked the user to try two structured recipes A_1, B_1 with either condition (AMMA and baseline). A_1 and B_1 both have four similar general steps: seasoning, cooking, cutting, and plating The two recipes have 11 and 12 actions involving different cooking components (ingredients and cookware) to reduce the learning effect. The recipes and conditions were counterbalanced to further address the learning effects and the impact of differences in the recipes. We measured the task completion time and the number of times the user adjusted the modality setting. After each recipe/condition, we assessed the task load of each version of the system using the NASA-TLX. Finally, we asked the participants for their overall preferences regarding the two conditions and thoughts on using different guidance modalities.

The procedure took around 60 minutes, and we compensated each user with a 30 USD gift card. Our university's Institutional Review Board approved this study. To closely simulate a real-world cooking experience, we also find a room large enough to fit in most of the main scenes of the cooking simulator so that the user is really walking to navigate the virtual environment.

6.2 Participants

We recruited 13 participants (six male and seven female, aged 19-40 $M = 27, \mu = 27$). We discarded P8's data because the simulation crashed many times, so P8 could not finish the task. Most of our participants had used VR before, with one using VR daily, two using VR weekly, two using VR monthly, six using VR a few times a year, and two using VR for the first time. Most of our participants are frequent cooks, with four cooking daily, four cooking weekly, four cooking monthly, and one cooking a few times a year.

6.3 Results

In the user study, we found that users performed the cooking tasks faster and needed to switch between modalities less with AMMA as compared to the baseline. We also observed that our user model can better predict users' action time after a personalization session.

6.3.1 Guidance Planner Performance

The average task completion time for AMMA and the baseline was 380.2 and 487.6 seconds, respectively (see Fig. 5a). We used a paired t-test to test the difference between the two conditions, and the result is statistically significant ($t = 5.031, p < .001$). Our

[1] https://store.steampowered.com/app/1358140/

[2] Source code of both parts can be found at https://jya.ng/amma

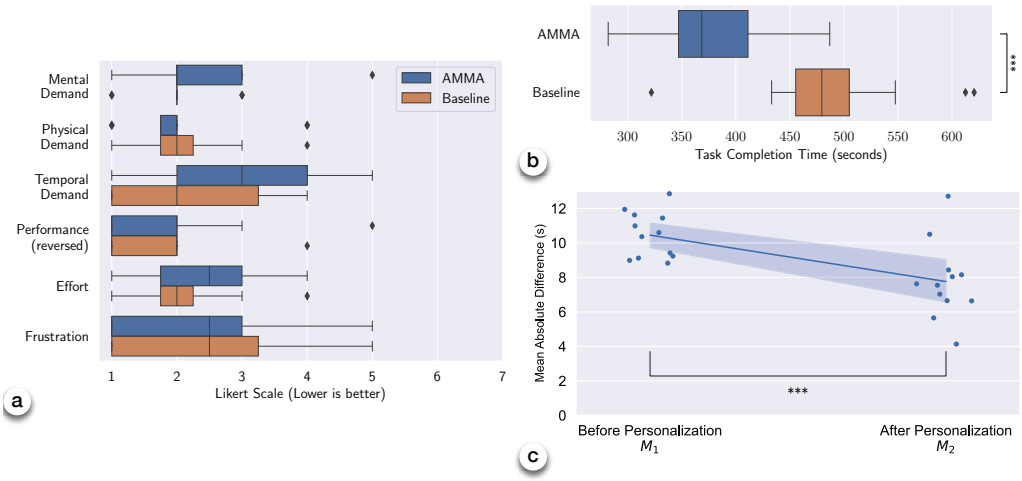

Figure 5: a) The average **task completion time** for AMMA and the baseline was 380.2s and 487.6s. Users spent significantly less amount of time cooking with AMMA ($t = 5.031, p < .001$). ***: $p < 0.001$ b) The medians of the **NASA-TLX** for AMMA and the baseline are 13.5 and 11.5, respectively ($U = 20.0, p = .243$, not significant). This showed the effect on task load is minimal. c) We compared the **user model estimation error** of the user's steps in the AMMA condition using the user model saved before personalization (M_1) and after personalization (M_2). We found that the user model can predict the step time significantly better after personalization than before ($t = 4.94, p < 0.001$). ***: $p < 0.001$ The blue region is a 95% confidence interval. The blue dots denote the original data points.

participants changed modalities 0.75 times and 1.5 times on average while using AMMA and the baseline system, respectively. A paired t-test shows this difference is statistically significant ($t = 2.287, p < .05$). Therefore, AMMA significantly reduced the user's completion time and the need to change between modalities.

6.3.2 State Tracker Performance

During our user study, we found that AMMA helps users to recover from 10 mistakes during the personalization and two experimental sessions (note that even the baseline uses AMMA's state tracker). Five participants poured too much oil into the pan, and the system successfully asked them to pour out the excess amount. Three participants shattered the plate, and the guidance system asked them to grab another plate. Two participants seasoned food products with the wrong ingredients, and the system correctly asked them to grab another food product to do again. Our system can help users recover from mistakes and complete tasks.

6.3.3 Cognitive Load

The medians of the NASA-TLX for AMMA and baseline are 13.5 and 11.5 (see Fig. 5b). While a slight disadvantage exists for AMMA, the result is not statistically significant ($U = 20.0, p = .243$). One notable difference was the lower temporal demand for the baseline. This is likely due to the increased density of instructions offered by AMMA. In practice, AMMA is only going to plan actions that the user model estimates the user should be able to finish in time, so we did not observe a significant impact on the task load.

6.3.4 User Model Performance

To confirm that our user model was adapting to users' behavior, we compared the user's models saved before the personalization session (a generic user model provided by us) M_1, and after the personalization session M_2, (see Fig. 5c). We evaluated the model accuracy using the user's step time captured in the AMMA condition. We compared M_1 and M_2's performance for each participant and found a statistically significant difference between the models (10.45 and 7.76 on average, respectively, $t = 4.94, p < 0.001$). This shows that our user modeling technique is working, and from observing user behavior, our user model is getting 26% better at estimating step time even after a single cooking session.

6.3.5 Qualitative Feedback

Overall, 10/12 participants considered AMMA to be more supportive of their preferences and capabilities. In the qualitative feedback on different guidance modalities, many participants reported that they liked a specific modality a lot, including the highlighting on the HUD (P1, P2, P4, P6, P10, P13), the text instructions on the HUD (P13), the audio guidance (P1, P6, P7, P12), and the video instruction on the monitor (P4). However, some people also said they did not like a specific modality, such as the instructions on the HUD (P2, P4, P11) or the video on the monitor (P2, P10, P13). The most common complaints were that the instructions on the HUD were too close to their eyes; and the video on the monitor is not useful as the experimenters had already explained how to do the task. Participants also reported task-related modality preferences (e.g., P4 thought the highlighting in the HUD was useful for picking up ingredients, and P6/P12 thought the text instructions on the HUD were helpful for baking and frying.) This shows a person-to-person and task-to-task variation in modality preference, which confirms our hypothesis in designing the guidance planner to adapt to the user's preference.

As for comments on the two implementations of the system (AMMA and the baseline), P7 pointed out that the highlighting on the HUD is turned on automatically in the AMMA condition for gathering ingredients, which is very useful. P10 pointed out that the AMMA condition is "more stressful but a lot of fun" as there are more parallel cooking steps, and the baseline guidance is more boring. P6 and P13 wished the guidance display could simultaneously show multiple steps' status. Other comments about the system and the simulation environments include: P10 noted that in a virtual environment, it is more fun to parallelize more steps for speed. However, he may prefer to trade speed for fewer errors in the real world. P6 wished she could close the oven door with her elbow, which is impossible in the simulated kitchen.

7 Discussion

We discuss other applications and limitations of AMMA and opportunities for future work, as well as how our techniques for guidance assistants might be used in general multimodal apps.

7.1 State Tracking and Guidance Planner Enhances Guidance Assistants

Guidance assistants to date have had limited use cases. Perhaps the most common use case is in mapping applications. The step-by-step driving guidance provides one example of a state tracker in today's guidance assistants. AMMA further generalizes the concept of a state tracker to other applications. Our evaluation showed that our state tracker can recognize and help users recover from their mistakes. In the future, more assistants can be built to utilize AMMA and make other tasks as easy as following turn-by-turn navigations.

Even in today's mapping applications, we have observed sub-optimal behaviors, such as when the app keeps asking users to make quick lane changes on crowded streets. We have shown that AMMA's user model can adapt to the user's abilities and aid the user in completing the cooking task faster. It would be useful if future guidance assistants consider the user's characteristics and give the most optimal guidance for the current user. Furthermore, AMMA suggests a better way of handling output modalities in guidance assistants over the conventional methods of manual customization. By providing an adaptive interface according to different users and tasks, AMMA allows users to make fewer manual adjustments, helping the user focus on the task rather than the system settings.

7.2 AMMA in Other Applications

Cooking was selected as a challenging case for guidance. The user needs to handle solid and liquid objects, manage tasks in parallel, perform varied actions, and deal with failure cases. These attributes cover a wide range of physical tasks, e.g., furniture assembly involves handling solid objects while not requiring parallelism. We built our cooking assistant based on prior psychology research, similar to the Keystroke-Level Model proposed by Stuart Card [8]. Domain experts can build similar models for other applications by looking at the actions users need to perform and the relevant psychological and physiological models of those actions.

AMMA can be used in more applications where adaptive guidance is useful. Here we describe two other use cases.

7.2.1 Furniture Assembly

Furniture assembly is one example that can benefit from the architecture of AMMA. For the furniture assembly activity, we define:
Tasks "assemble a dining table", "assemble a bed", etc.
Steps "unwrapping component", "slot together components", etc.
Actions "screw in a screw", "insert an component into a slot", etc.
UIs a HUD showing animation/text instruction, audio guide, etc.
We can also provide a user model where the time is related to the user's hand dexterity, the object's weight, and the user's spatial reasoning speed. One tricky part about furniture assembly [1] is spatial conflicts between the components, so the system must check for conflicts to determine the possible steps. With AMMA, the assistant can learn which modality (text instructions, videos, or figures) is the best for each user for each task.

7.2.2 Turn-by-Turn Navigation

Even though turn-by-turn navigations today already have the concept of a state tracker, AMMA can provide a better guidance planning algorithm. AMMA can organize multimodal UIs, including:
HUD AR overlay arrows indicating the next turn/showing a mini-map to the destination with a list of upcoming turns/off
Audio giving a short or long audio guidance/off
Display show only the current turn/show the current turn and the next few turns/show the full route
We can use feedback from the user, such as the stress level detected in the user's steering motion [35] or explicit modality switching, to determine the best UI for the user. We can also model the user's driving behavior (e.g., relative driving speed, tendency to make mistakes for different instructions, etc.) to plan a personalized route.

7.3 Limitations and Future Work

Currently, the AMMA-based cooking assistant in the cooking simulator has a high accuracy of user action estimation. This emulates a scenario where the actions are not sensed by pervasive IoT sensors attached to cookware. In the real world, we think computer state tracking and the six scenarios we set out to handle (Fig. 3) will still apply in more realistic settings. To work with other common computer vision-based inputs, future systems can allow users to cancel the effect of an action to avoid major errors in state tracking.

As a proof of concept, the cooking assistant only supports cooking actions in the Cooking Simulator game. The cooking assistant can be further expanded with actions and steps to accommodate more diverse cooking actions (e.g., mashing potatoes or steam cooking). Another limitation of the simulated evaluation is that the actions and time may differ from cooking in the real world. We believe that in real life, as the cooking times are longer, the increased perceived task load will decrease more, and the benefit of accurately estimating the user's cooking time and planning accordingly should persist.

A limitation we discovered during the user study is the transparency of the user model. The game broke many times during P8's user study session, so the user model estimated a very low success rate. Therefore, the system asked the user to grab more ingredients than instructed in the recipe to ensure the user could successfully cook the dish. In a real-world deployment, perception errors can also lead to similar results. In that case, it would be useful to allow users to help the user model recover from it.

7.4 From Adaptive Guidance Assistants to Mixed-initiative Multimodal Apps

In our work, AMMA is solely focused on guidance assistants. In other mixed-initiative applications [34], the idea of state tracking and user modeling-based planning can also be beneficial when the machine is taking more of the lead. For example, when an assistant receives a user request missing some parameters, the assistant usually needs to conduct slot filling, which means the assistant guides the user to provide all the necessary information for the request.

Another interesting direction is to investigate using user models and guidance planners for goals other than completion time. We may want to reach the goal within a reasonable time but minimize our carbon footprint or maximize the fun of doing a mundane task by asking users to try different paths drawing on affective computing. The architecture of the guidance planner (user model, simulator, and policy algorithm) can be adjusted to better achieve a secondary goal.

Choosing the most appropriate feedback for the user and the scenario is also crucial for the future of VR/AR [27]. VR/AR elevates computer-to-human information transfer beyond 2D screens. Information can appear anytime, anywhere, in any form, like the guidance modalities in the cooking assistant. This opportunity brings possibilities but also concerns. Implemented inappropriately, this can be a disruption rather than a pleasant experience. Future applications outside the guidance domain can consider using a similar approach to decide the best feedback for a user in the scenario to achieve the best experience between computer systems and human users.

8 Conclusion

AMMA shows a promising future where computer systems can learn the user's current status, preferences, and capabilities and give guidance accordingly. Through our study, we demonstrated an assistant built with AMMA could allow users to complete the task faster, reduce the number of manual adjustments of the modality, and learn users' behavior through observations. By expanding the capability of computers to understand human behaviors, future guidance assistants will be able to help people better achieve their tasks.

ACKNOWLEDGMENTS

We thank the reviewers and the user study participants for their feedback. We also thank Tianshi Li for her feedback on writing and the system. During the writing of the paper, Harry Xu also gave valuable advice for the RL policy algorithm. Some icons are from Material Design Icons. This work is partly supported by the National Science Foundation Grant No. 1900638 and Meta Platforms, Inc.

REFERENCES

[1] M. Agrawala, D. Phan, J. Heiser, J. Haymaker, J. Klingner, P. Hanrahan, and B. Tversky, "Designing effective step-by-step assembly instructions," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 828–837, jul 2003. [Online]. Available: https://doi.org/10.1145/882262.882352 9

[2] R. Arakawa, H. Yakura, V. Mollyn, S. Nie, E. Russell, D. P. DeMeo, H. A. Reddy, A. K. Maytin, B. T. Carroll, J. F. Lehman, and M. Goel, "Prism-tracker," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 6, no. 4, p. 1–27, Dec 2022. [Online]. Available: http://dx.doi.org/10.1145/3569504 1

[3] J. Bassen, B. Balaji, M. Schaarschmidt, C. Thille, J. Painter, D. Zimmaro, A. Games, E. Fast, and J. C. Mitchell, "Reinforcement learning for the adaptive scheduling of educational activities," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, apr 2020. [Online]. Available: https://doi.org/10.1145/3313831.3376518 2

[4] R. A. Bolt, ""put-that-there"," in *Proceedings of the 7th annual conference on Computer graphics and interactive techniques - SIGGRAPH '80*. ACM Press, 1980. [Online]. Available: https://doi.org/10.1145/800250.807503 3

[5] M. J. S. Brinkhuis, A. O. Savi, A. D. Hofman, F. Coomans, H. L. J. V. der Maas, and G. Maris, "Learning as it happens: A decade of analyzing and shaping a large-scale online learning system," *Journal of Learning Analytics*, vol. 5, no. 2, aug 2018. [Online]. Available: https://doi.org/10.18608/jla.2018.52.3 2

[6] J. L. Burke, M. S. Prewett, A. A. Gray, L. Yang, F. R. B. Stilson, M. D. Coovert, L. R. Elliot, and E. Redden, "Comparing the effects of visual-auditory and visual-tactile feedback on user performance," in *Proceedings of the 8th international conference on Multimodal interfaces - ICMI '06*. ACM Press, 2006. [Online]. Available: https://doi.org/10.1145/1180995.1181017 3

[7] Y. Cao, X. Qian, T. Wang, R. Lee, K. Huo, and K. Ramani, "An exploratory study of augmented reality presence for tutoring machine tasks," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, apr 2020. [Online]. Available: https://doi.org/10.1145/3313831.3376688 1, 2

[8] S. K. Card, T. P. Moran, and A. Newell, "The keystroke-level model for user performance time with interactive systems," *Communications of the ACM*, vol. 23, no. 7, pp. 396–410, jul 1980. [Online]. Available: https://doi.org/10.1145/358886.358895 2, 6, 9

[9] C.-J. Chao, H.-C. K. Lin, J.-W. Lin, and Y.-C. Tseng, "An affective learning interface with an interactive animated agent," in *2012 IEEE Fourth International Conference On Digital Game And Intelligent Toy Enhanced Learning*. IEEE, Mar. 2012. [Online]. Available: http://dx.doi.org/10.1109/DIGITEL.2012.60 2

[10] S. Cofer, T. N. Chen, J. Yang, and S. Follmer, "Detecting touch and grasp gestures using a wrist-worn optical and inertial sensing network," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 842–10 849, oct 2022. [Online]. Available: https://doi.org/10.1109/lra.2022.3191173 1

[11] D. Damen, H. Doughty, G. M. Farinella, A. Furnari, E. Kazakos, J. Ma, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "Rescaling egocentric vision: Collection, pipeline and challenges for EPIC-KITCHENS-100," *International Journal of Computer Vision*, vol. 130, no. 1, pp. 33–55, oct 2021. [Online]. Available: https://doi.org/10.1007/s11263-021-01531-2 1

[12] C. G. DRURY and S. M. WOOLLEY, "Visually-controlled leg movements embedded in a walking task," *Ergonomics*, vol. 38, no. 4, pp. 714–722, apr 1995. [Online]. Available: https://doi.org/10.1080/00140139508925143 6

[13] J. B. V. Erp and H. A. V. Veen, "Vibrotactile in-vehicle navigation system," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 7, no. 4-5, pp. 247–256, jul 2004. [Online]. Available: https://doi.org/10.1016/j.trf.2004.09.003 3

[14] A. Fathi and J. M. Rehg, "Modeling actions through state changes," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2013. [Online]. Available: https://doi.org/10.1109/cvpr.2013.333 1

[15] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement." *Journal of Experimental Psychology*, vol. 47, no. 6, pp. 381–391, 1954. [Online]. Available: https://doi.org/10.1037/h0055392 6

[16] K. Gajos and D. S. Weld, "SUPPLE," in *Proceedings of the 9th international conference on Intelligent user interface - IUI '04*. ACM Press, 2004. [Online]. Available: https://doi.org/10.1145/964442.964461 2

[17] K. Z. Gajos, J. O. Wobbrock, and D. S. Weld, "Automatically generating user interfaces adapted to users' motor and vision capabilities," in *Proceedings of the 20th annual ACM symposium on User interface software and technology - UIST '07*. ACM Press, 2007. [Online]. Available: https://doi.org/10.1145/1294211.1294253 2, 6

[18] A. Gupta, D. Fox, B. Curless, and M. Cohen, "DuploTrack," in *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*. ACM Press, 2012. [Online]. Available: https://doi.org/10.1145/2380116.2380167 3

[19] R. Hamada, J. Okabe, I. Ide, S. Satoh, S. Sakai, and H. Tanaka, "Cooking navi," in *Proceedings of the 13th annual ACM international conference on Multimedia - MULTIMEDIA '05*. ACM Press, 2005. [Online]. Available: https://doi.org/10.1145/1101149.1101228 2, 3

[20] S. G. Hart, "Nasa-task load index (NASA-TLX) 20 years later," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 50, no. 9, pp. 904–908, oct 2006. [Online]. Available: https://doi.org/10.1177/154193120605000909 7

[21] G. Huang, X. Qian, T. Wang, F. Patel, M. Sreeram, Y. Cao, K. Ramani, and A. J. Quinn, "AdapTutAR: An adaptive tutoring system for machine tasks in augmented reality," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, may 2021. [Online]. Available: https://doi.org/10.1145/3411764.3445283 1, 2

[22] X. Jin, X. Hu, X. Wei, and M. Fan, "Synapse," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 6, no. 3, p. 1–24, Sep 2022. [Online]. Available: http://dx.doi.org/10.1145/3550321 1

[23] M. Johnston, J. Chen, P. Ehlen, H. Jung, J. Lieske, A. Reddy, E. Selfridge, S. Stoyanchev, B. Vasilieff, and J. Wilpon, "MVA: The multimodal virtual assistant," in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. Association for Computational Linguistics, 2014. [Online]. Available: https://doi.org/10.3115/v1/w14-4335 3

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980 6, 7

[25] B. T. Kumaravel, C. Nguyen, S. DiVerdi, and B. Hartmann, "TutoriVR," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, may 2019. [Online]. Available: https://doi.org/10.1145/3290605.3300514 1, 2

[26] J. Lacoche, E. Villain, and A. Foulonneau, "Evaluating usability and user experience of AR applications in VR simulation," *Frontiers in Virtual Reality*, vol. 3, jul 2022. [Online]. Available: https://doi.org/10.3389/frvir.2022.881318 2

[27] M. Lee, M. Billinghurst, W. Baek, R. Green, and W. Woo, "A usability study of multimodal input in an augmented reality environment," *Virtual Reality*, vol. 17, no. 4, p. 293–305, Sep. 2013. [Online]. Available: http://dx.doi.org/10.1007/s10055-013-0230-0 9

[28] T. Li, J. K. Haines, M. F. R. de Eguino, J. I. Hong, and J. Nichols, "Alert now or never: Understanding and predicting notification preferences of smartphone users," *ACM Transactions on Computer-Human Interaction*, feb 2022. [Online]. Available: https://doi.org/10.1145/3478868 2

[29] D. Lindlbauer, A. M. Feit, and O. Hilliges, "Context-aware online adaptation of mixed reality interfaces," in *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. ACM, oct 2019. [Online]. Available: https://doi.org/10.1145/3332165.3347945 2

[30] Y.-C. Liu, "Comparative study of the effects of auditory, visual and multimodality displays on drivers' performance in advanced traveller information systems," *Ergonomics*, vol. 44, no. 4, pp. 425–442, mar 2001. [Online]. Available: https://doi.org/10.1080/00140130010011369 3

[31] C. Martinie and P. Palanque, "Task models based engineering of interactive systems," in *Companion Proceedings of the 12th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, ser. EICS '20. ACM, Jun. 2020. [Online]. Available: http://dx.doi.org/10.1145/3393672.3398647 2, 6

[32] K. Miyawaki and M. Sano, "A virtual agent for a cooking navigation system using augmented reality," in *Intelligent Virtual Agents*. Springer Berlin Heidelberg, pp. 97–103. [Online]. Available: https://doi.org/10.1007/978-3-540-85483-8_10 2

[33] T. Mu, S. Wang, E. Andersen, and E. Brunskill, "Automatic adaptive sequencing in a webgame," in *Intelligent Tutoring Systems*. Springer International Publishing, 2021, pp. 430–438. [Online]. Available: https://doi.org/10.1007/978-3-030-80421-3_47 2

[34] A. Paranjape, A. See, K. Kenealy, H. Li, A. Hardy, P. Qi, K. R. Sadagopan, N. M. Phu, D. Soylu, and C. D. Manning, "Neural generation meets real people: Towards emotionally engaging mixed-initiative conversations," 2020. [Online]. Available: https://arxiv.org/abs/2008.12348 9

[35] P. E. Paredes, F. Ordonez, W. Ju, and J. A. Landay, "Fast &amp: Furious," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Apr. 2018. [Online]. Available: https://doi.org/10.1145/3173574.3174239 9

[36] C. M. Park, T. Ki, A. J. B. Ali, N. S. Pawar, K. Dantu, S. Y. Ko, and L. Ziarek, "Gesto," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. EICS, pp. 1–22, jun 2019. [Online]. Available: https://doi.org/10.1145/3300964 3

[37] X. Qian, F. He, X. Hu, T. Wang, A. Ipsita, and K. Ramani, "ScalAR: Authoring semantically adaptive augmented reality experiences in virtual reality," in *CHI Conference on Human Factors in Computing Systems*. ACM, apr 2022. [Online]. Available: https://doi.org/10.1145/3491102.3517665 2

[38] A. Schroeder, C. Kroiß, and T. Mair, *Context Acquisition and Acting in Pervasive Physiological Applications*. Springer Berlin Heidelberg, 2012, p. 393–400. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29154-8_48 2

[39] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016. [Online]. Available: https://doi.org/10.1038/nature16961 6

[40] J. J. Yang, G. Banerjee, V. Gupta, M. S. Lam, and J. A. Landay, "Soundr: Head position and orientation prediction using a microphone array," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, apr 2020. [Online]. Available: https://doi.org/10.1145/3313831.3376427 1

[41] J. J. Yang, M. S. Lam, and J. A. Landay, "DoThisHere," in *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. ACM, oct 2020. [Online]. Available: https://doi.org/10.1145/3379337.3415841 1, 3

[42] J. J. Yang, T. Chen, F. Qin, M. S. Lam, and J. A. Landay, "HybridTrak: Adding full-body tracking to VR using an off-the-shelf webcam," in *CHI Conference on Human Factors in Computing Systems*. ACM, apr 2022. [Online]. Available: https://doi.org/10.1145/3491102.3502045 1

[43] G. N. Yannakakis, "Game adaptivity impact on affective physical interaction," in *2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*. IEEE, Sep. 2009. [Online]. Available: http://dx.doi.org/10.1109/ACII.2009.5349384 2

[44] J. Zauner, M. Haller, A. Brandl, and W. Hartman, "Authoring of a mixed reality assembly instructor for hierarchical structures," in *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.* IEEE Comput. Soc, 2003. [Online]. Available: https://doi.org/10.1109/ismar.2003.1240707 2