Predictive Task Guidance with Artificial Intelligence in Augmented Reality

Benjamin Rheault[*] University of Florida
Shivvrat Arya[†] University of Texas at Dallas
Akshay Vyas[‡] University of Texas at Dallas
Jikai Wang[§] University of Texas at Dallas
Rohith Peddi[¶] University of Texas at Dallas
Brett Benda[‖] University of Florida
Vibhav Gogate[**] University of Texas at Dallas
Nicholas Ruozzi[††] University of Texas at Dallas
Yu Xiang[‡‡] University of Texas at Dallas
Eric D. Ragan University of Florida

Abstract

We put forward an augmented reality (AR) system using artificial intelligence (AI) to guide users through real-world procedures. The system uses the cameras of the headset to recognize objects and their positions in real time. It constructs a model to predict: (i) the task being completed, (ii) the step of the task they are on, and (iii) whether they have made any errors. By updating this model in real-time as the user completes the task, the system automatically updates the instructions and cues provided to the user. This system represents a step towards ubiquitous task guidance via AR.

Index Terms: Human-centered computing—Virtual Reality—Mixed / augmented reality

1 Introduction

Augmented reality (AR) systems leveraging artificial intelligence (AI) can provide powerful tools for guiding people through a workflow by providing instructions and superimposing visual cues into the world. By performing object detection and real-time error detection while dynamically updating instructions, users can complete procedures faster and with fewer mistakes. We developed a model that is able to track the state of a task and provide this information to a user through an AR headset. This system makes use of the sensors commonly equipped on such headsets to provide video and spatial data to the model, allowing it to maintain this state.

While the system has broad potential applications to a wide range of procedures, we demonstrate the system within the context of cooking recipes. Our system is able to use sensor data to identify the procedure being completed from a pre-existing set of trained recipes. Multiple recipes are also able to be tracked simultaneously, and can be automatically switched between to support concurrent progress. Having been trained on these recipes, the model is able to detect when an error has occurred in the completion of a recipe and provide an alert to the user detailing the type of error, on what step it was made, and how to address it.

[*]e-mail: brheault@ufl.edu
[†]e-mail: shivvrat.arya@utdallas.edu
[‡]e-mail: akshay.vyas@utdallas.edu
[§]e-mail: jikai.wang@utdallas.edu
[¶]e-mail: rohith.peddi@utdallas.edu
[‖]e-mail: brett.benda@ufl.edu
[**]e-mail: vibhav.gogate@utdallas.edu
[††]e-mail: nicholas.ruozzi@utdallas.edu
[‡‡]e-mail: yu.xiang@utdallas.edu
e-mail: eragan@ufl.edu

Figure 1: A user interacting with the UI. Included elements are the *main menu* (top), *instruction panel* (bottom), and *task list view* (right).

2 User Interface

The AR system is run and displayed with the Microsoft HoloLens 2 headset. The visual interface was designed in Unity and has three main components: the *instruction panel*, the *task list view*, and the *main menu* (see Figure 1). All of these components can be grabbed, moved, and scaled to the user's preferences. The state of the recipe—and therefore the contents of the interface—is determined by our *object detection* and *step prediction* models. These models are not always accurate, so the ability of the user to correct them is central to our interaction design philosophy.

All buttons in the interface can be pressed with a hand motion or by the user fixating their gaze on them. Because tasks often involve manipulating objects, allowing selection with gaze fixation bypasses the need to let go of objects or stop working on the task.

The *instruction panel* displays the text of the current step, as detected by our *step prediction* model (see Section 4). When the model detects that the current step is complete, the step is incremented. If the detected step is inaccurate, the user can correct the model by using the right and left arrows to navigate to the correct step.

The *task list view* displays the completion status for all the steps in the recipe. Five steps are displayed on each page and the user is able to move through pages to review them all. A checkmark or X button next to each step allows the user to correct the model if a step is incorrectly labeled as complete/incomplete.

The *main menu* provides the user with personalization and manual controls. Additional recipes can be added, removed and switched between by opening the View Recipes submenu. This can be useful if the model fails to recognize a recipe or if the user wishes to begin a recipe for which the materials are not present to be recognized. The instructions panel and object locators can also be toggled on and off. Object locators are red dots that appear in the view over objects that are needed to complete the current step (see figure 2).

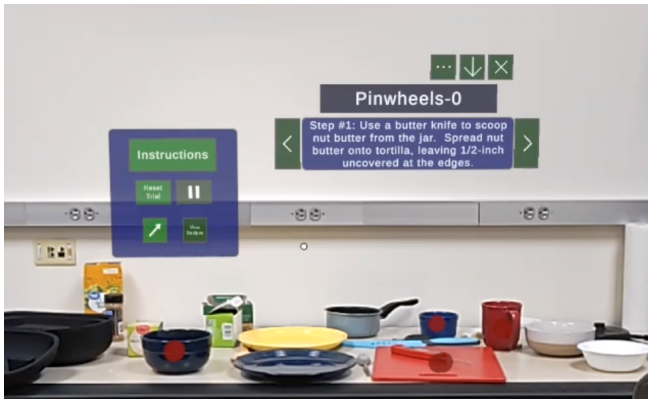

Figure 2: A mixed reality capture of a recipe being completed. Objects needed for the recipe are indicated in the view with Object Locators (red spheres rendered in AR).

3 Object Detection

The object detection phase consists of several steps. The first one is powered by the *Grounding DINO* [4] open-set object detector, which adeptly identifies objects, yielding bounding boxes as initial output. These bounding boxes are further analyzed using the *CLIP* model for image feature extraction, which are fed into the *PROTO CLIP* [5] model. The latter refines these bounding boxes, attaching them with class names from a pre-determined set of 37 object classes.

For accurate real-time prediction of object masks for each detected object, we employed MobileSAM [7], a variant of SAM [3] optimized for efficiency. The integration of these models guarantees precise object detection and segmentation, enabling us to pinpoint exact object locations.

The last step is spatial mapping which generates precise 3D coordinates for each detected object. This is achieved by aligning depth frames with RGB frames, thus creating depth masks for each object. These masks are essential in accurately mapping each object's position in 3D space. We use camera poses, intrinsic parameters, and depth masks to transform the centers of detected 2D segmented object masks into a global coordinate system. This enables the accurate projection of corresponding 3D points onto the headset's view as shown in figure 2.

The above pipeline results in the detailed annotation of the input image with bounding boxes, class labels, and the 3D coordinates of each object. Furthermore, the pipeline generates comprehensive masks and center points for every object, with all critical data, including the 3D positions, communicated efficiently as ROS messages. This methodology enhances object localization accuracy in AR environments and broadens the scope for innovative applications in spatially complex AR settings.

4 Step Prediction

In *step prediction*, we begin by annotating video segments with corresponding recipe steps. This annotated data becomes the training dataset for a deep neural network, which uses Omnivore, a transformer-based feature extractor [2], along with a linear classifier emulating a noisy sensor for the dynamic probabilistic graphical model (dynamic PGM). The dynamic PGM also incorporates information about object and hand locations for user interaction analysis.

Additionally, the dynamic PGM integrates prior recipe knowledge using task graphs to represent step order. As these task graphs are directed and acyclic, any topological sort produces a valid step sequence. This input aids the dynamic PGM in assessing step correctness and correcting feature extractor errors.

At each time step, the query variable—a Boolean indicator denoting the completion of a step—relies on the continuous input from the neural network, the persistence of the query variable from the previous timestamp, and the values of both descendants and ancestors of the query variable from the previous timestamp.

We estimate the query variable using the classical particle filtering algorithm [1]. This algorithm employs sequential importance sampling to approximate the query variable by using particles. Following evaluation and weighting based on evidence probability and prior states, we duplicate higher-weighted particles and eliminate less accurate ones through re-sampling. This process yields the states of each recipe step for presentation in the AR interface.

5 Discussion and Future Work

We have presented an AR task guidance system capable of tracking multiple concurrent cooking recipes and providing real-time feedback and instruction. As was stated in Section 2, the ability to correct the model on the state of the active recipe(s) is core to the design of the user interface. As we continue development on the system, we intend to make this functionality less and less necessary.

The current *step prediction* model principally relies on data about ongoing activities and user-object interactions. These methods may falter in scenarios involving occluded objects, steps with no physical actions, and instances where actions or objects closely resemble or form a subset of others. Enhancements to the model could address these challenges through the incorporation of hand and object tracking and deducing actions from their respective trajectories. While our dynamic probabilistic graphical model compensates for certain inaccuracies, its efficacy could be improved by reducing noisy input signals. Thus, accurate prediction and localization of steps in egocentric procedural activities is a challenging problem. We have also introduced a new dataset, CaptainCook4D [6], to encourage research in this area.

In terms of the user interface, the interactions rely heavily on physically pressing buttons, a desktop-oriented interaction technique. The ability to press buttons with gaze fixation (see Section 2) was added as a step toward AR-friendly interactions, but the interface itself is still analogous to desktop applications. In future iteration, we will explore alternative designs for 3D interaction techniques.

Acknowledgments

This work was supported in part by the DARPA Perceptually-enabled Task Guidance (PTG) Program under HR00112220005.

References

[1] A. Doucet, N. de Freitas, and N. J. Gordon. *Sequential Monte Carlo Methods in Practice*. Statistics for Engineering and Information Science. Springer, 2001.

[2] R. Girdhar, M. Singh, N. Ravi, L. Maaten, A. Joulin, and I. Misra. Omnivore: A single model for many visual modalities. *Computer Vision and Pattern Recognition*, 2022. doi: 10.1109/CVPR52688.2022.01563

[3] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

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

[5] K. Palanisamy, Y.-W. Chao, X. Du, Y. Xiang, et al. Proto-clip: Vision-language prototypical network for few-shot learning. *arXiv preprint arXiv:2307.03073*, 2023.

[6] R. Peddi, S. Arya, B. Challa, L. Pallapothula, A. Vyas, J. Wang, Q. Zhang, V. Komaragiri, E. Ragan, N. Ruozzi, Y. Xiang, and V. Gogate. Captaincook4d: A dataset for understanding errors in procedural activities, 2023.

[7] C. Zhang, D. Han, Y. Qiao, J. U. Kim, S.-H. Bae, S. Lee, and C. S. Hong. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*, 2023.