https://doi.org/10.2352/EI.2024.36.8.IMAGE-243

Neuro-Symbolic Ensembles for Assistance at the Edge

Yan-Ming Chiou; SRI International; Palo Alto, CA, USA
Dick Crouch; SRI International; Palo Alto, CA, USA
Bob Price; SRI International; Palo Alto, CA, USA
Jiaying Shen; SRI International; Palo Alto, CA, USA
Michael Youngblood; SRI International; Palo Alto, CA, USA
Charlie Ortiz; SRI International; Palo Alto, CA, USA

Abstract

Intelligence assistance applications hold enormous potential to extend the range of tasks people can perform, increase the speed and accuracy of task performance and provide high quality documentation for record keeping. However, the computational complexity of modern perception and reasoning techniques based on massive foundation model networks cannot run on devices at the edge. A remote server can be used to offload computation but latency and security concerns often rule this out. Distillation and quantization can compress networks but we still face the challenge of obtaining sufficient training data for all possible task executions. We propose a hybrid ensemble architecture that combines intelligent switching of special purpose networks and a symbolic reasoner to provide assistance on modest hardware while still allowing robust and sophisticated reasoning. The rich reasoner representations can also be to identify mistakes in complex procedures. Since system inferences are still imperfect, users can be confused about what the system expects and get frustrated. An interface which makes the capabilities and limitations of perception and reasoning transparent to users dramatically improves the usability of the system. Importantly, our interface provides feedback without compromising situational awareness through well designed audio cues and compact icon-based feedback.

Introduction

While the internet is full of knowledge, it is still not easy to get exactly the knowledge we need in the moment to solve practical physical tasks. Youtube is a great source of information about the details of removing, fixing and assembling devices such as phones or lawn equipment, but it can be hard to find a quality video and hard to search through a video to get what you need at a certain step. Perhaps our lawn mower will not start and we are not sure why. Finding a solution on the web often depends on situational variables: How do you describe the failure in terms of sounds or appearances? What is the type of mower? Is it gas or electric? What is the make and model? When was it last serviced and what was done? Once the user has located the relevant information, the user still has the problem of applying the knowledge to their situation. For obvious diagnostic steps such as checking if there is gas or a good electrical connection, the average owner might feel confident performing the required action, but the user may be hesitant to attempt maintenance steps such as checking for clogged filters or fouled plugs even though these are within the capabilities of the average owner using common tools. The tasks often require identification of parts and judgments about condition that are hard to make based on a couple of photos in a manual.

In professional applications such as manufacturing, medical and military operations it may also be important to verify completion of steps and provide a record that can be signed off and referred to in the future to explain problems and formulate policy changes.

Recent advances in large language models and multi-modal models hold out the promise of using contextual prompts such as an image or some free text spoken or typed into an app to pull out relevant knowledge from a user manual or internet. However, access to LLM based knowledge typically requires either a network connection or a powerful GPU equipped host to run the inference and it is not always clear how to ground this knowledge in the context of a real world (what does a fouled spark plug look like and how would I get to it and remove it?).

Augmented reality systems can improve the usefulness of knowledge by using overlays on the performer's view to highlight relevant parts. Augmented reality systems need a way of relating functional knowledge to appearance of objects in the real world and require real time performance to be usable. Latency from network services can be excessive making context relevant assistance difficult to achieve with devices in the field. A perceptive augmented reality system can also observe when tasks are complete and advance to the next step freeing the user's hands and attention to focus on the task.

Finally, a user might perform the procedure in a different order that is different from the manual. This ordering may still be valid but more appropriate if the user was waiting for a part to be delivered. It is not possible to write manuals or record videos covering all possible task orderings. Users may also occasionally make a mistake or leave out a step. The system needs to figure out if the mistake will compromise the task. Given a mistake, the system needs to be able to recommend how to recover from the error. Again, it is not practical to produce manuals or videos covering all possible mistakes and recovery actions and even if it was, it wouldn't be practical for the user to locate the right version for their circumstance.

SRI's AMIGOS system is an autonomous augmented reality assistant which develops computationally tractable perception and reasoning capabilities that could elevate augmented reality from a passive display to an intelligent observing partner that understands the context and supplies information to the performer using modest hardware. This would require efficient perception, a rich representation of task and an interface that allows the performer and system to understand each other.

Unlike traditional strictly feed forward systems, we use the task structure to actively guide the perception system computation to focus on perception of things relevant to the current task. Where many systems use either a probabilistic, neural or logical framework, we use a combination of machine learning and symbolic hierarchical task networks (HTNs) to reduce the time to define tasks while preserving the ability to deal with the

combinatorics of multiple valid execution orderings with minimal training data.

Method

The system (see Figure 1) uses several perceptual modules to identify the state of the task. Task reasoning relates observations from the perception system to steps in tasks and decides when a step has been achieved. The guidance manager takes into account the user's state and level of expertise to decide when and how to provide guidance to the user. The dialog manager chooses specific language for interventions and can respond to questions from the user. An AR interface delivers speech, auditory prompts and visual aids such as text, diagrams and animations.

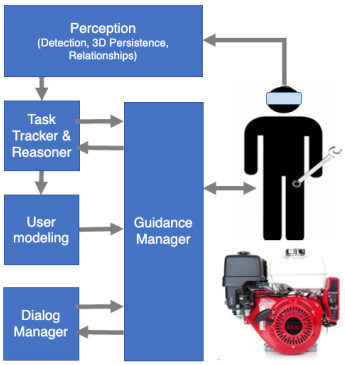

Figure 1 AMIGOS System Architecture

Perception

The perception system provides the context to the assistant about the state of the workspace and the changes the user is making. Perception is difficult in systems based on head-mounted ego-centric cameras as the view changes erratically as the operator moves around and looks at different parts. This leads to fleeting glimpses of the scene and motion blur. Explicitly introducing varying amounts of motion blur (not Gaussian blur!) into the computer vision training pipeline improves recognition of objects. Key parts of the scene can also be occluded by the hands or tools or clipped at the edges of the camera frame. We augment the data with artificial occlusions to handle this. It is also important to optimize the framerate of the vision pipeline to maximize the stability of object tracking.

To provide context to the reasoning system, the perception system must identify objects (e.g., air filter), attributes of those objects (e.g., clean or dirty), relationships between objects (e.g., the air filter is removed from the engine). In physical tasks it also important to track specific instances of objects. The clean air filter to be installed on the machine might look similar to the dirty filter already removed from the machine when sitting on the work bench, but the system should identify the correct instance to install.

Because of the transient nature of the camera view, any one frame may or may not reflect relevant properties leading to errors in perception. It is also important to be able to detect negative evidence. The fact that the filter is not on the engine is suggested by observing the absence of the relationship between the filter and the engine.

In computer vision datasets, multi-modal transformer approaches have shown high performance in offline competitions [1], but in our experiments these approaches fail to provide high frame rates for real-time assistance applications and require significant training data on many sequences. We therefore use a lighter weight approach that exploits factorization to improve generalization. Factoring recognition into detection of isolated objects and then calculating relationships between the objects allows the system to be trained on less data and to robustly recombine object models for new tasks (See Figure 2).

To integrate the sparse information from camera frames, the system maintains a persistent set of objects that are updated based on the vision system. A light weight YoloX [2] based detector finds domain specific objects in frames. The augmented reality headset also produces a depth map. For each detected object, we determine a sampling region inside the object detection borders and then use the corresponding sampling region to estimate the distance of the object to the camera. We discard depth points where the depth is invalid and use the median of the remaining set to get a robust estimate. If the percentage of valid points falls below 20%, we discard the observation as unreliable.

This method is heuristic and can fail for objects with a large central hole (e.g., tire). The method also only provides information about one point on the front surface of the object and can be thrown off by occluding items in front of the object. Because we see the same object many times, these errors are generally averaged away. We extract telemetry about the 6-DOF position and orientation of the user's head from the augmented reality system localization. We then align the telemetry to the image frames using temporal interpolation and then project the depth from the camera frame into a 3D world position. We then use a database of object meta data to associate the points with a spatial extent which is approximated by a cylinder with the appropriate height and diameter. Once the candidate objects have a cylindrical spatial extent and are in a 3D world frame, they can be matched to stored persistent instances with the same object class using a Hungarian algorithm and the 3D Euclidean distance for similarity. If the detection is too far from an existing instance, a new instance is created. The notion of "too far" is defined relative to the spatial extent definition so that large objects need to be farther apart than

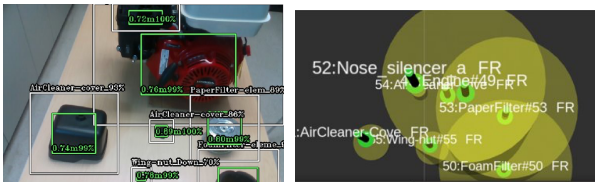

Figure 2 The left hand image shows object detections and the depth sampling windows. The right hand image shows generated 3D instances with cylindrical spatial extents.

smaller ones. The framework can also merge instances that appear to be the same object. The persistent instances create a stable world view. If the user turns their head away and then back again, new detections of the object will be associated with the old persistent instance since the new object will be projected back to the original 3D location. Using the object's 3D location and spatial extent, we can compute relationships such as above, in front of inside, etc. These relationships will be used by the reasoning system to infer the current state and what the user has done. Object instances and relationships based on them also keep track of time since first seen and the number of observations since being seen. Detection confidence is also recorded. The extra information is used to the determine confidence and relevance of the extracted information.

Identifying objects and tracking their position and relationships gives the system a crude idea of what resources are available and how they are being manipulated. In many cases, more detailed kinds of knowledge are required. For instance, to verify that the correct amount of a substance is used, the system might need to be able to read the display of a scale. Running optical character recognition continuously would be too expensive.

The task structure can be used to make perception more efficient or reliable [11]. In our case, instead of conditioning perception on the previous step, we condition on the current step to select specific perceptual modules. The task-informed switching framework uses

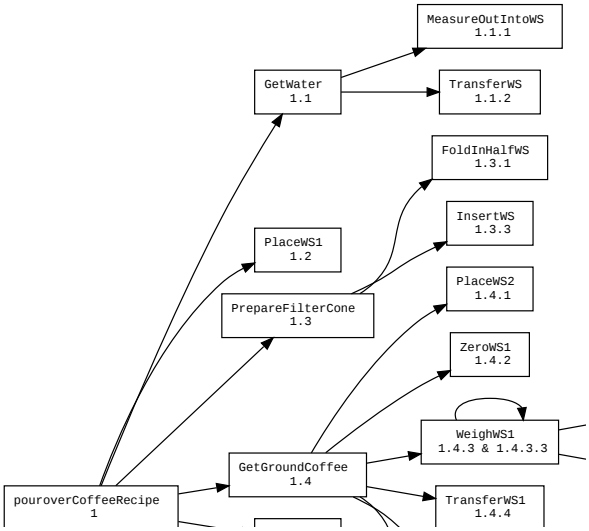

Figure 4 A fragment of a hierarchical task network for making pour over coffee shows how the main task is decomposed into subtasks which are decomposed into primitive actions.

explicit knowledge about the step the system is on to determine when OCR is required and enables a previously loaded network to accept the camera feed to process characters. Enabling allows the system to switch networks in real time to read characters off the scale without waiting to load the network or access remote servers.

In some cases, actions do not have observable effects on the world. A person might press a button or kneed some dough resulting in a change to the environment that is not readily perceptible. For this case, we fine-tuned a heavy weight transformer based action model called MViT2 [3]. The system used knowledge of the step in the task being performed to dynamically enable this preloaded network to detect critical actions relevant to that step but leaving the network disabled otherwise.

Task Reasoning

The task reasoning component accepts input from the perception system and uses it to infer what the user is doing. If the user brings the water pitcher over the kettle, the user is likely working on a tea or coffee making task and likely on the step where the kettle is being filled with water so that it can be heated. The reasoner relies on having a formal representation of the task. Our system employs Hierarchical Task Networks (HTNs) [4][5] to represent tasks, substasks and specific steps in a procedure (See Figure 4). Each sub-step specifies the objects involved as inputs and outputs, and any pre-conditions and post-conditions that must apply to those objects (See Figure 3). Additional temporal ordering constraints can be imposed between sub-steps. While HTNs are typically employed for planning, in this instance they are compiled out into a special optimized form used for plan recognition [6]. The ordering constraints can be hard or soft. Hard constraints require a previous step. For instance, water needs to be in the kettle before it is set to boil. In other cases, there might be a soft constraint, but it is not absolute. It is better to place the teabag in the cup before pouring water to create more agitation, but you can leave the tea to steep longer if you forget.

Meta action analysis was performed on thousands of recipes to mine good action primitives [7] to use as the basis for representing recipes. The procedure steps were then translated into these action primitives. We experimented with some methods of automatically translating procedure descriptions into action primitives but they are not fully reliable at this time, especially when a high level task such as “measure coffee beans” needs to be broken into substeps by common sense (e.g., get out the scale, place container on scale, zero scale, add beans until weight is achieved). For this reason, the results here are based on manual construction of HTN representations.

```
{"id": "pouroverCoffeeRecipe_getWater_transferWater",
  "utensils": [ "water2", "liquidMeasuringCup1",  "kettle1"   ],
  "preconditions": {
       "expr": {
           "predicate": "relative_location",
           "arguments": ["water2",    "liquidMeasuringCup1"] }
},

  "postconditions": {
       "expr": {
          "predicate": "relative_location",
          "arguments": [   "water2",     "kettle1",  } }

  "duringconditions": null     }
```

Figure 3 A fragment of a hierarchical task network (HTN) for transferring water from measuring cup to kettle showing pre and post conditions.

Matching the real-time, transient, noisy observations at 25 Hz to the logical world of hierarchical networks is challenging. Temporal stabilization is used to filter out observations that are unreliable (e.g., only seen a small number of times, or haven’t been seen in a long time) and to convert a continuous stream of observations to discrete changes in the environment. So repeated observations of kettle on table followed by observations of kettle over mug are just translated into a single event “The kettle has moved from the table to over the coffee mug”.

Experience has shown that articulating the exact conditions required to recognize a procedure step using the presence of objects, the relationships between these objects and confidence factors such as detection confidence, time since last seen, etc. are difficult to manually articulate in a general way. For this reason, we employed a hybrid framework which used a hierarchical task network (HTN) to represent the structure of the task and a machine learned classifier trained on high-level features from the perception system to recognize common objects and relationship relevant to recognizing subtasks in the procedure. The task and step classifiers were trained on a half a dozen or so recordings of executions for each task, where the timestamps for the start and end of the steps were hand-labeled. The HTN allows flexibility of execution due to its ability to represent multiple ways of decomposing a task. The learned classifier simplified the definition of conditions to identify steps and can be trained on a very small number of examples because the hard part of recognizing objects from pixels is done by another pretrained network.

Specifically, the stabilized perceptions of instances and types of objects observed, object co-ordinates and attributes, and 3D spatial relations between objects (over, near, etc.) together with confidence scores are turned into feature vectors for input to (1) a task classifier, and (2) step classifiers, one for each task. The dimensions of the vectors correspond to objects and selected attributes and spatial relations (e.g., kettle present, measuring cup

is over kettle). Each value in the vector represents a temporal duration of the respective variable: if negative, the duration represents how long since that feature was last observed, if positive the duration represents how long that feature has continuously been observed, and if zero the feature has never been observed.

The task tracker is initialized with compiled out HTNs for all known tasks. The stabilizer and classifiers translate perceptions into preconditions for the HTN reasoner. As preconditions are satisfied, the tracker checks off subtasks in the procedure whose preconditions are satisfied.

It was discovered that using the temporal feature values encoding how long something had been seen rather than just binary present/not-present values significantly increased the accuracy of the classifiers (see results) as well as producing smaller models. It is hypothesized that this is because the temporal values capture some of the preceding run up to each step. For example, a step like "stirring a mixture" begins when the stirring occurs but is typically preceded by picking up the stirring implement and moving it toward the mixture. A simple snapshot would not capture this preparatory transition.

Unfortunately, the high accuracy of the task and step classifiers does not directly translate into accurately recognizing the exact start times and especially the end times of steps. For the starts of steps there is a danger of brief false positives, and it is prudent to wait for a few successive high confidence predictions that the step is in progress before deciding it has started. For the ends of steps, absence of prediction is not necessarily prediction of absence. Particularly in a multi-task setting some steps can be paused in the middle while a step from a different task is worked on. And in a single task setting users can also pause the step to consult instructions.

To ameliorate this, a second set of step transition classifiers were trained. These used a combination of features from the first set of classifiers (e.g. step confidence, precision/recall for the step classification) plus features derived from the current HTN-based task state (e.g. if there is a step currently in progress; is it one that has already been marked as completed?; if the step is not in progress, is it the one that is expected to occur next?; or is it one that could legitimately occur next but is not the expected one?; if the step is for a task that is not in progress, is it a reasonable first step for the task?). Classifiers to predict start-of-step, end-of-step, and start-of-task are trained from the same labeled data as the step classifiers. These transition classifiers in effect combine purely observational data with expectations derived from reasoning about the HTN task state to produce a hybrid form of task tracking.

The HTN representation also allows detection of errors that is difficult to do with hidden-Markov model-based trackers [8][9] that simply assign low probabilities to events not in the procedure. In HMMs it can be difficult to discriminate between task irrelevant activities and erroneous steps that affect the task outcome. Training neural network sequence recognition models is currently popular [10] but training requires recording many possible error scenarios to cover all of the things that could be done incorrectly or out of order. With an HTN, errors in task execution can be detected when sub-steps violate obligatory constraints and warnings can be triggered when sub-steps violate optional constraints.

Guidance Manager

The task tracker provides information about how the user's behavior matches or departs from the formal model captured by the HTN. The guidance manager determines whether to intervene during the user's performance. The guidance manager takes into account the seriousness of departures and uses a model of the user's expertise level and communication preferences. Messages that rise above the threshold are sent on to the user interface, along with positional information about the objects involved in the message.

Figure 5 AMIGOS Augmented Reality interface showing multitask assistance and status bar

User Interface

Figure 5 illustrates the view a user sees when performing two distinct tasks: making coffee, and making tea. A master task dialog can be used to manage tasks by creating new tasks or deleting existing ones (Figure 5 left). These tasks were displayed through an Augmented Reality (AR) interface on the user's AR glasses (Figure 5 right). Upon starting a task, a task panel appears in the user's field of view. The panel has a task specific icon and is anchored to the initial key object for the task (see Figure 6). In this case, the mug serves as an anchor. Physically localizing assistance dialogs to task objects keeps tasks separate in space so that it is easy to keep track of which task is at which step.

Figure 6 Once a recipe is selected, the instructions automatically anchor to a focal object for the recipe in the user's view (e.g., mug). When multiple tasks are in progress, the task bar shows an icon for each task and the dialog for the non-focal task can be compressed to just its icon. The background color of the task icon shows its status.

To simplify the management of multiple tasks, a compact task bar appears at the bottom of the users view (Figure 5 right). The task bar shows the status of all tasks using compact graphical status icons. Active tasks are displayed with full-color icons and inactive tasks are displayed with non-filled outlines.

Instruction panels may optionally include text instructions, sample images, a timer to time key events (e.g., grind beans for 20 seconds) and 3D animations to explain complex steps. When the user needs to wait during a task or decides to switch to a different task, the system cleverly shrinks the currently active panel into a small icon. This efficient design saves visual space and lowers cognitive load, aiding the user in focusing on the next tasks. The process of moving between tasks is designed to keep the user's concentration focused.

The perception system and task tracker keep track of the user's progress and automatically advance the instructions as long as the user is correctly executing the task. Auto advance technology enables hands-free interaction which is critical when users are executing complex tasks. When the user completes a step in the task, the system will flash the task icon and set its background green to reassure the user they have successfully completed the step before automatically moving forward to the next step (Figure 8 left). When immediate attention is needed (e.g., the user has made a critical error) the system employs multi-modal alerts that combine auditory signals from the AR glasses with visual cues like red blinking icons to ensure the user's attention is drawn to the relevant task (Figure 8 right). To accommodate user's with color blindness, a chime, blinking and a unique icon symbol provide an alternate method of drawing the user's attention.

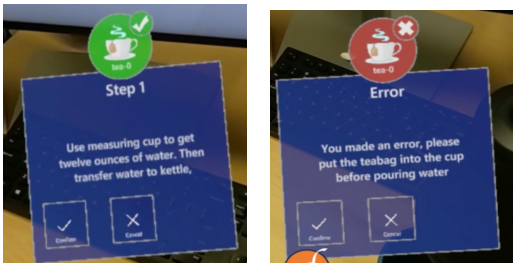

Figure 8 Active perception provides feedback when steps are completed correctly with a momentary green check and chime before auto advancing to the next step enabling hands free operation. When an error is detected (step skipped, or wrong ingredient used), a red error icon and audio alert are presented.

In addition to specific instructions provided on panel, the system also included audio prompts to provide hints to the user about what objects the system had seen and what the system was expecting next. For instance, the system might say, "I am looking for the teabag in the mug". This helps the user understand the condition the system is looking for. The user quickly learns that moving a bit closer so the system can see over the edge of the mug to reveal the teabag inside can advance the system to the next step. By ensuring the system clearly communicates its expectations, the user learns to work effectively with the system.

Results

Factoring of the perception system into multiple modules dramatically increased the performance while maintaining the ability to perform advanced perceptual tasks when required. Using a large foundation model was taking on the order of 200-300 ms reducing frame rates to 3-5 frames per second. This made it more difficult to associate objects to persistent instances as the objects move a lot between distantly spaced frames. Using YoloX and occasionally relaying frames to separate networks for tasks such as digit recognition on displays or action recognition allowed us to get frame rates in the mid to high 20s allowing smooth tracking of objects. Before implementing task switching, the system required multiple GPUs to run and experienced significant latency. With intelligent task switching, a single modest GPU allowed real time operation without significant lag in the application.

Experiments showed that the use of high-level classifiers on the temporally stabilized features were effective in identifying both tasks and the steps within tasks. As shown at the bottom of Figure 7, a classifier trained on a dozen example runs achieved 84% accuracy. Other task domains (e.g., Tea) showed similar performance.

```
Number of labels: 9
Number of features: 194
Accuracy:
              precision    recall  f1-score   support

         0.0       0.81      0.89      0.85      2059
         1.0       0.84      0.75      0.79       296
         2.0       0.75      0.74      0.74        57
         3.0       0.76      0.73      0.74       235
         4.0       0.92      0.88      0.90      1359
         5.0       0.88      0.74      0.80       205
         6.0       0.80      0.75      0.77       163
         7.0       0.80      0.67      0.73       221
         8.0       0.80      0.63      0.70       100

    accuracy                           0.84      4695
   macro avg       0.82      0.75      0.78      4695
weighted avg       0.84      0.84      0.84      4695

Confusion matrix:
[[1828   38    9   35   83   13   15   23   15]
 [  61  222    0    7    3    3    0    0    0]
 [  15    0   42    0    0    0    0    0    0]
 [  51    4    3  172    2    1    1    1    0]
 [ 133    1    2   13 1196    1    3   10    0]
 [  43    0    0    0   10  151    0    1    0]
 [  34    0    0    0    3    2  122    2    0]
 [  56    0    0    0    2    1   12  149    1]
 [  35    0    0    0    2    0    0    0   63]]
```

Figure 7 Precision, recall and F1 score for recognition of the 9 steps of the coffee recipe using values of extracted relationships

Subsequent experiments showed that including the context information about how long it had been since the object or relationship had been observed dramatically increased accuracy into the high 90s. In Figure 9, tracking on the steps of the Coffee task reached 98% accuracy.

```
Number of labels: 9
Number of features: 194
Accuracy:
              precision    recall  f1-score   support

         0.0       0.97      0.99      0.98      2059
         1.0       0.98      0.98      0.98       296
         2.0       1.00      0.93      0.96        57
         3.0       1.00      1.00      1.00       235
         4.0       1.00      0.99      1.00      1359
         5.0       0.99      0.95      0.97       205
         6.0       0.96      0.95      0.96       163
         7.0       0.99      0.94      0.96       221
         8.0       1.00      0.92      0.96       100

    accuracy                           0.98      4695
   macro avg       0.99      0.96      0.97      4695
weighted avg       0.98      0.98      0.98      4695

Confusion matrix:
[[2039    5    0    1    4    1    6    3    0]
 [   5  291    0    0    0    0    0    0    0]
 [   4    0   53    0    0    0    0    0    0]
 [   1    0    0  234    0    0    0    0    0]
 [   9    0    0    0 1350    0    0    0    0]
 [  11    0    0    0    0  194    0    0    0]
 [   8    0    0    0    0    0  155    0    0]
 [  14    0    0    0    0    0    0  207    0]
 [   8    0    0    0    0    0    0    0   92]]
```

Figure 9 Precision, recall and F1 score of the 9 steps of the coffee recipe with the addition of time since last seen used as a proxy for confidence. Including time since last seen significantly decreases confusion and increases recognition from 84% to 98%.

As show in Figure 10, it was more difficult to estimate the exact start and end times of the steps within a procedure with accuracies dipping into high 70s and low 80s. In many steps, the onset may not be clearly and unambiguously signaled.

```
Coffee
start: 81% of 62; early 7.4s x 3; late 2.0s x
 end: 66% of 62; early 0.0s x 0; late 38.1s
1. getWater:
  start: 100% of 9; early 1.0 x 1; late 6.9 x 8
   end: 89% of 9; early 0.0 x 0; late 37.4 x 8
2. placeDripper:
  start: 88% of 8; early 0.0 x 0; late 0.6 x 7
   end: 75% of 8; early 0.0 x 0; late 26.7 x 6
3. prepareFilter:
  start: 75% of 8; early 21.0 x 1; late 0.5 x 5
   end: 75% of 8; early 0.0 x 0; late 15.3 x 6
4. getGroundCoffee:
  start: 75% of 8; early 0.0 x 0; late 0.9 x 6
   end: 75% of 8; early 0.0 x 0; late 20.1 x 6
5. checkWaterTemperature:
  start: 75% of 8; early 0.0 x 0; late 1.0 x 6
   end: 75% of 8; early 0.0 x 0; late 8.6 x 6
6. wetGrounds:
  start: 75% of 8; early 0.0 x 0; late 1.1 x 6
   end: 62% of 8; early 0.0 x 0; late 58.4 x 5
7. pourWater:
  start: 75% of 8; early 0.1 x 1; late 3.1 x 5
   end: 62% of 8; early 0.0 x 0; late 128.9 x 5
8. drainCoffee:
  start: 83% of 6; early 0.0 x 0; late 4.2 x 5
   end: 0% of 6; early 0.0 x 0; late 0.0 x 0Coff
```

Figure 10 Detecting the exact begin and end point of a step is more difficult and somewhat subjective, but the system is able to generate reasonable estimates.

The hybrid HTN and machine learning based classifier was much easier to configure and get running than explicitly articulating the conditions for each task step. This holds out the potential to increase the rate at which new tasks could be added to the system.

The user interface feedback significantly increased the usability of the system. When users were able to understand what

system had seen and not seen they could modify their behavior to make it more observable or if appropriate, confidently override the system when they understand why something was not observed. This kind of adaptation also occurs in human-human systems where a second inspector needs to check off tasks. Humans learn to communicate to make difficult to observe actions observable. These observations were confirmed by a third party evaluator for an initial version of the system that found the feedback significantly improved the usability over systems that did not provide feedback about the state of the system and the system's expectations about what needed to happen next.

Conclusions

Pure learning approaches based on continuous video recognition have the potential to learn very rich representations but require enormous amounts of training data to handle variation in context, execution variations and the ability to detect common errors. Factoring the problem into object detection and digit recognition with separate networks using task state to schedule networks accelerates performance. Formal models with plan recognition enable the system to handle a wider variety of task execution orders and explicit inference about possible errors without exhaustively recording all possible executions. This paper shows that machine learning, symbolic computation and interface transparency used together can enable a practical and effective design for physical task execution at the edge.

Acknowledgements

Funded by DARPA Perceptually-enabled Task Guidance contract *HR001122C0009*

References

[1] Liu, Shilong, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li et al. "Grounding dino: Marrying dino with grounded pre-training for open-set object detection." *arXiv preprint arXiv:2303.05499* (2023).

[2] Ge, Zheng, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. "Yolox: Exceeding yolo series in 2021." *arXiv preprint arXiv:2107.08430* (2021).

[3] Li, Yanghao, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. "Mvitv2: Improved multiscale vision transformers for classification and detection." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4804-4814. 2022.

[4] Nau, Dana, Yue Cao, Amnon Lotem, and Hector Munoz-Avila. "SHOP: Simple hierarchical ordered planner." In *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, pp. 968-973. 1999.

[5] Georgievski, Ilche, and Marco Aiello. "An overview of hierarchical task network planning." *arXiv preprint arXiv:1403.7426* (2014).

[6] Höller, Daniel, Gregor Behnke, Pascal Bercher, and Susanne Biundo. "Plan and goal recognition as HTN planning." In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 466-473. IEEE, 2018.

[7] Schank, Roger C. "Conceptual dependency: A theory of natural language understanding." *Cognitive psychology* 3, no. 4 (1972): 552-631.

[8] Lane, Terran. "Hidden markov models for human/computer interface modeling." In *Proceedings of the* IJCAI-*99 Workshop on Learning about Users*, pp. 35-44. 1999.

[9] Asghari, Parviz, Elnaz Soleimani, and Ehsan Nazerfard. "Online human activity recognition employing hierarchical hidden Markov models." *Journal of Ambient Intelligence and Humanized Computing* 11 (2020): 1141-1152.

[10] Wang, Xiang, Shiwei Zhang, Zhiwu Qing, Yuanjie Shao, Zhengrong Zuo, Changxin Gao, and Nong Sang. "Oadtr: Online action detection with transformers." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7565-7575. 2021.

[11] Stanescu, Ana, Peter Mohr, Mateusz Kozinski, Shohei Mori, Dieter Schmalstieg, and Denis Kalkofen. "State-Aware Configuration Detection for Augmented Reality Step-by-Step Tutorials." In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 157-166. IEEE, 2023.

Author Biography

Yan-Ming Chiou received his Ph.D. from the University of Delaware. He has research interests in augmented reality, generative AI, multi-modal foundation models, and large language model. He now works at SRI International in augmented reality and computer perception for assistance systems

Dick Crouch received his PhD in Computer Science from the University of Cambridge (1993). He has maintained a research interest in formal representations to support computational semantics for natural language and has applied this in various guises at research institutes (SRI Cambridge, Defense Research Agency, University of Nottingham, Xerox PARC) as well as in industry (Microsoft, Nuance, Amazon, Chegg). He now works at SRI International on conversational systems.

*Bob Pric*e received his PhD in Computer Science from University of British Columbia (2003), and did a post doc in user modeling at University of Alberta (2006). He worked at the Xerox Palo Alto Research Center on modeling mobile users, diagnostics for industrial printers, credit card fraud, health care outcome prediction, and medical image analysis and industrial processing via video. He now works at SRI International in augmented reality and computer perception for assistance systems.

Jiaying Shen; recived her PhD in Computer Science from University of Massachusetts Amherst (2007). Her research interests span across various AI fields including plan-based dialog management, natural language understanding, and multi-agent systems. Her expertise has been applied successfully in R&D efforts across both large organizations (Nuance, Samsung) and startups (npc.work). She currently works at SRI International on collaborative conversational systems.

Michael Youngblood received his PhD in Computer Science and Engineering from the University of Texas at Arlington (2005) and earned tenure as an Associate Professor at the University of North Carolina at Charlotte (2012). He worked at the Xerox Palo Alto Research Center (2012-2023) on augmented social cognition involving health and wellness goal-driven teams, explainable AI, and machine learning with low-targeting-high fidelity simulations. He now works at SRI International (2023-) on collaborative conversational systems.

Charlie Ortiz; received his PhD from the University of Pennsylvania Computer and Information Science (1996). He has experience leading and executing work in knowledge represent and dialogue systems for both commercial (Nuance) and government projects. He has also done research in multiagent and multirobot systems. He is currently Associate Director of Collaborative Conversational Systems group at SRI; Palo Alto, CA, USA.