# Building Jarvis - A Learner-Aware Conversational Trainer

**Shiwali Mohan, Kalai Ramea, Bob Price, Matthew Shreve, Hoda Eldardiry, Les Nelson**
Palo Alto Research Center
Palo Alto, California
shiwali.mohan@parc.com

## ABSTRACT

Our long-term research goal is to develop intelligent systems that can support human learning. We are particularly interested in developing an approach to *apprenticeship* learning which occurs during the physical context of task execution and is known to be very effective in learning procedural tasks such as equipment maintenance or artifact assembly. We describe our initial progress in building such system - Jarvis - that leverages real-time computer vision, high-level inference, and augmented reality technology to monitor and support human task learning through apprenticeship.

## CCS CONCEPTS

• **Computing methodologies → Discourse, dialogue and pragmatics**; **Spatial and physical reasoning**; **Vision for robotics**; *Planning with abstraction and generalization.*

## KEYWORDS

task-oriented dialog, situated interactive instruction, interactive task learning, hybrid AI, embodied dialog, cognitive agent, human-agent interaction, Soar, computer vision for agents, real-time reasoning

**ACM Reference Format:**
Shiwali Mohan, Kalai Ramea, Bob Price, Matthew Shreve, Hoda Eldardiry, Les Nelson. 2019. Building Jarvis - A Learner-Aware Conversational Trainer. In *Joint Proceedings of the ACM IUI 2019 Workshops, Los Angeles, USA, March 20, 2019*. ACM, 9 pages.

## 1 INTRODUCTION

Conversational systems are becoming pervasive in our world; they support us in accessing information, performing tasks, and communicating with other people. As strides are made in AI, ML, and NLP algorithms, it is reasonable to expect that intelligent conversational systems will become more sophisticated and will integrate seamlessly with our physical and online worlds, assisting us in a variety of tasks. We are motivated to develop conversational systems that can support human task learning in physical worlds. Tasks include maintaining and repairing a complex machine such as an industrial printer or building an artifact such as Ikea furniture. To be effective trainers, conversational systems have to be aware of and adaptive to two types of contexts: physical context of task performance and cognitive context of the human learner. In this paper, we explore how a conversational system can reason about and adapt to these contexts. To do this, we bring together deep learning approaches for computer vision and planning approaches for adaptive instruction reasoning.

### Training by Apprenticeship

Learning from social interactions is one of the most fundamental forms of learning in the human society [1, 20, 21]. Parents contribute to early learning by helping children to segment, categorize, and classify their perceptions through language. Teachers train students in arithmetic, problem solving, and conceptual learning. A master guides skill development in apprentices through coaching. During such interactions, the facilitator instructor and the primary learner form a system of joint learning, with the former helping the latter in achieving critical conditions of learning. The instructor may provide examples through demonstrations, restrict exploration through supervision, provide feedback and encouragement, and scaffold learner's performance. Interactive learning critically distributes the onus of learning between the interacting participants. The instructor takes initiative in identifying the relevant objects and relationships in the shared environment, structuring and decomposing tasks, and providing relevant explanations. The learner takes initiative in actively interpreting the instructions, applying them to the current situation, analyzing successes and failures, posing relevant queries that elicit useful information from their instructors, and assimilating information with their world knowledge.

Our research aims at developing intelligent conversational systems that can support human learning via *apprenticeship*. Apprenticeship learning is process of acquiring knowledge within the actual, physical context of practice. When compared to classrooms where learning is more theoretical and focuses on the *why*, apprenticeship learning focuses on the *how*. Consequently, such training is shown to be very effective for learning physical, procedural tasks such as equipment maintenance or furniture assembly. This paper introduces Jarvis - Just an Augmented Reality/Virtual Instruction System - that can guide people as they are performing new, physical tasks. Through an augmented reality headset, such as the HoloLens, Jarvis has a shared view of the human trainee's world. It can observe the trainee's workspace through the egocentric camera on *HoloLens* and follow human task execution. If the trainee struggles to make progress towards their task goal, Jarvis provides instructions on how to proceed further through instructional dialog. The dialog may incorporate mixed-reality visual elements such as highlighting relevant objects, overlaying expected motion, etc.

### Challenges for Conversational Trainers

There are several challenges that must be addressed in order to build adaptive conversational training systems that can guide human task performance and learning in physical worlds. There is significant perceptual reasoning complexity. First, Jarvis must inspect its video stream in real-time to identify relevant objects, their parts, their relationships with other objects, and their functional state. Then, it must adapt to partial observability of the domain due to the ego-centric camera. As the human trainee looks through *HoloLens* at the workspace, only a portion of the workspace is visible. The perceptual information changes rapidly as the trainee human moves about their workspace. Jarvis must integrate partial perceptual information it receives into a stable state representation of the environment so that it is able to reason about task execution meaningfully. To adapt instructions to be most relevant, Jarvis must model the task the trainee is attempting to perform. The task model crucially includes relevant objects, relationships, actions, and goals so that Jarvis can dynamically generate plans to help the trainee achieve task goals based on the current problem state. Human trainees cannot be guaranteed to follow the instruction script exactly, i.e., human trainees may divert from a specific script and explore their environment. Therefore, Jarvis must be adapt to such non-determinism in its reasoning. Further, Jarvis must model the specific trainee, their needs and knowledge, to provide instructions relevant to them. Finally, these instructions should be provided naturally with minimum interference with the trainee's task performance. The conversational training system should be mixed-initiative, providing opportunities to the trainee to ask for information relevant to their knowledge and performance.

These challenges are tremendous and require integration of algorithms from several diverse AI disciplines including computer vision, knowledge representation and reasoning, conversational systems, and human modeling among others. In this paper, we describe the preliminary steps taken towards building this adaptive conversational system. We focus on two problems:

- **Perceptual state integration:** The conversational system should maintain a stable environmental state in presence of partial vision from ego-centric camera for instruction generation, and
- **Contextual instruction generation**: The conversational should adapt its instruction to be most relevant given real-time environmental state.

To do this, we bring to bear deep learning vision algorithms [6, 8] and the Soar cognitive architecture [7]. The main contribution of this paper is integrating deep learning architectures that operate with numeric, continuous representations with rule-based inference architectures that rely on relational representations for intelligent adaptive behavior in a conversational system.

## 2 PRELIMINARIES

We begin by a brief overview of deep learning architectures for computer vision and that of the Soar cognitive architecture. Later we describe how both of these are brought together in a system that can reliably perform real-time perceptual reasoning and provide contextual instructions.

### Deep Learning Architectures for Vision

Modern computer vision techniques are based on an old idea called convolutional neural networks (CNNs) [8]. In this model, patterns are captured in terms of real-valued weight patterns. These patterns are repeated across various locations with identical weights in order to enable the ability to recognize the same pattern in different locations in an image.

In combination with larger datasets and GPU based computation accelerators, deep versions of convolutional neural networks that have many layers of convolutions now dominate the field [6]. These deep networks allow patterns to be composed hierarchically. Low-level patterns identify edges or color textures. Higher level patterns identify relevant features such as the presence of eyes. The highest level layers identify semantic entities such as a face.

Deep convolutional neural networks typically do well on large datasets. However, the most popular networks such as AlexNet [6], GoolgLeNet [18], VGGNet [17], and ResNet [3] have not only shown promising results in large datasets, but have also revolutionized the field of computer vision through transfer learning, where pretrained weights from these models, specifically those that can identify low-level patterns can now be used as a starting point for several tasks where the dataset size is modest.

While specialized architectures have been developed for each task such as image classification and pixel-wise segmentation, we are particularly interested in object detection that localize multiple instance of objects of various classes in an image frame. One example method is the Single shot multibox detector (SSD) [9], which has demonstrated to have high throughput and achieve impressive accuracy. In the SSD architecture, an output array is constructed with distinct dimensions which represent distinct discrete hypotheses about the position, aspect ratio, and scale of an object in each image frame. These dimensions are predicted using a real valued regression function which fine tune the position and dimensions as well as provide a confidence value about the class of the object.

### Soar Cognitive Architecture

The cognitive reasoning components of Jarvis are implemented in the Soar cognitive architecture [7]. Soar has been used extensively to develop interactive intelligent systems [4, 10, 12–14] as well as cognitive models of human task learning [11]. Through its various long-term memories and corresponding learning algorithms, a Soar agent can represent different kinds of knowledge that aid in reasoning about perceptual state as well as instructions.

**Working Memory:** A Soar agent's beliefs about the current state are held in its working memory. These beliefs are described using relational representations and are derived from its current and recent sensory data of the world, current goals, and its interpretation of the situation given its goals. The data in working memory is represented as a symbolic, labeled graph of working memory elements (WMEs). Working memory input and output buffers provide interfaces to the perceptual and interactive interface. WM also has interfaces to other long-term memories.

**Procedural Memory:** Soar's *how-to* knowledge is encoded as if-then rules called productions. A rule has two parts: conditions and actions. The *conditions* part of the rule match against the working memory graph and changes it by either adding or removing nodes in it as per the *actions* part. The rules fire in parallel. Soar does two kinds of inference:

(1) *elaboration*: Elaboration rules perform non-persistent monotonic inference given a elements in the current state. This means that the actions part of the rule is only part of the working memory graph only until the conditions part matches the working memory. When conditions stop matching, the changes to the working memory graph are reversed.
(2) *operator-supported*: Operator proposal, *selection*, and *application* rules perform persistent inference. Based on the current state,the proposal rules are similar to elaboration rules and suggest an operator to be applied, the selection rules evaluate all the operators that have been proposed, and once a selection is made the application rules make a persistent changes to the working memory graph. This change can only be reversed through another operator-supported process. An operator is the locus of decision making in Soar and this process of proposing, selecting, and applying operator is at the core of deliberative reasoning in Soar.

Soar has a meta-cognitive process built into it. Where there are no further inferences to be made or there is not a set of rules that can select one specific operator, Soar agent has an *impasse*. The impasse results in a new sub-state with the previous state (in which reasoning became stagnant) becoming a super-state. The sub-state persists until further reasoning in the sub-state can determine the next inference step or operator for the superstate. For example, if the next operator cannot be determined for the super-state, the sub-state may execute a one-step search to evaluate which available operator when applied directly to the superstate results into a goal state. This impasse-driven resolution process is at the center of all problem solving in Soar.

**Long-term Declarative Memories:** Apart from these components, a Soar agent also has an episodic memory that is a store of all past experiences of the agent. Episodic memory is a graph that efficiently encodes transitions in the working memory graph over the lifetime of the agent. The Soar agent also has a semantic memory that is a store of declarative facts about the world. The agent can access the contents of these memories by executing a query against them. The information is retrieved in a working memory buffer and is available for problem solving.

## 3 ENVIRONMENT OVERVIEW

To study apprenticeship training, we are exploring the domain of equipment operation and maintenance. A printer is a common piece of equipment found in office. While the typical use of printers is fairly straightforward, complicated tasks such as printing a booklet are not trivial and can benefit from contextual assistance. As printers become more complex in what they can do, maintenance tasks such as replacing various toner cartridges are becoming more challenging for a typical user. Repairing a complex, electronic machine is challenging for most users of the machine. Further, manufactures have a large variety of offerings that differ along several dimensions. As a domain, equipment operation and maintenance provides a wide array of tasks of

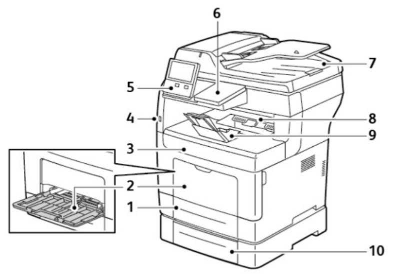

6
7
5
4
8
9
3
2
1
10
1. Tray 1
2. Bypass Tray
3. Front Door
4. USB Memory Port
5. Control Panel
6. Card Reader Bay
7. Duplex Automatic Document Feeder
8. Output Tray
9. Output Tray Paper Stop
10. Optional 550 Sheet Feeder, maximum of 3 (Trays 2–4)

**Figure 1: VersaLink B405 and its outer components. Reproduced from the manual that accompanies the printer.**

varying difficulty that can be used to study training efficacy of intelligent systems. From a usability perspective, autonomous instruction systems can allow the users to perform maintenance tasks, fix problems and use advanced features without waiting for a service call or trainer to come out to their site. Such systems also reduce the cost of service by eliminating the need to fly out technicians to remote sites.

For this paper, we greatly simplify the range of tasks that Jarvis provides instructions on and focus on maintenance operations on a specific Xerox printer - VersaLink B405 (in Figure 1). Our task environment consists of the following:

- Components: The printer components relevant for the discussion in this paper are:
  (1) door-type components: *bypass-tray*(2), *front-door*(3), *document-feeder*, *paper-tray*,
  (2) movable components: *drum-cartridge*, and *toner-cartridge*
  (3) locks: *drum-cartridge-lock*

  The door-type components remain attached to the printer and can be in states *open* or *closed*. The movable objects can be removed from the printer and they can be in states *in* or *out*. Some movable components can be in the *locked* state in which they are fastened to the assembly via a locking mechanism and have to be unlocked before being moved.
- Actions: The actions a human trainee can perform are targeted towards changing the state of various components. This state change can expose other components of the printer. For example, the trainee may *open the front-door* changing the state of the component to *open*. This will expose the components *drum-cartridge* and toner-cartridge.
- Human-Agent Interaction: The instructional interaction occurs through a visual interface using which the human trainee can *set a task* to a goal they are interested in or ask *what's next?* Jarvis responds via text as well. This interaction interface is preliminary in design and we expect Jarvis to interact with the human in a audio-visual conversation in which the next action is described through speech and the relevant components are highlighted in augmented reality. This paper focuses on the knowledge representations and algorithms behind instructional reasoning.

## 4 AGENT DESIGN

### Generating Perceptual Events

Jarvis' vision system uses the single shot detector (SSD) model described previously to generate candidate detections of parts and their current states. The detector model is trained in a separate process, using ego-centric video collected using the HoloLens as well. During the data collection process, bounding boxes are constructed around the objects, and ego-centric videos are recorded through a custom-made Hololens application called ARLabeler [16]. There are two significant advantages of this approach compared to the traditional labeler used on 2D images: (a) Hololens can record ego-centric views as opposed to third-person view of the objects, and consequently, the model can be directly incorporated in Jarvis for detection; (b) the markers are placed on a mesh that follows the object's curvature, therefore, the bounding box coordinates do not shift their positions when viewed from different angles. The videos recorded from the Hololens are saved for pre-processing (each frame is saved as an image). These images along with their bounding box coordinates are then converted to a training dataset in PASCAL format [2] for developing a detection model.

The SSD model we implemented uses pretrained weights from reduced VGG-16 network [9] and fine-tunes the final detection layers on the training images collected for the printer. The trained weights are saved and used in real-time detection. As part of the detector, overlapping detections of the same class are suppressed; we used a modest non-maximum suppression (NMS) value of $0.45$, so that it suppresses weak object detections to avoid multiple detection of the same object but at the same time allows strong detections of multiple objects that may overlap in the real-world.

The model is intended to detect objects in each frame of the video feed. Because the user operates the system with a hand-held or head-mounted device, the image frame can jump around dramatically, and motion blur can be introduced into the frame. This can result in spurious detections or false positives. In order to make the detections more robust, we pass the real-time detections through a temporal smoothing filter which consists of an appearance threshold and a sliding window across the frames. The appearance threshold is defined as the ratio of the number of times the object is detected in each frame to the total number of frames in the sliding window. For example, if the appearance threshold is 0.8, and sliding window length is 20 frames, then the objects should have been detected at least in 16 frames to make to the final detection list. Larger sliding window length leads to more robust predictions, but the final detection list output is slower, so there is a trade-off between latency of detection and stability that must be tuned empirically.

Figure 2 shows the perceptual events generated by the vision system. Detected component states such as *document feeder open*, *drum cartridge in* etc. along with their bounding boxes and detection confidence are included in the event stream. This event stream is fed into the state integrator described below.

## Maintaining Task State

As we discussed earlier, the HoloLens camera is ego-centric and at a time point only has partial perceptual information of the task. For example, Figure 2 shows various images collected while the trainee attempts to remove the *drum cartridge* (visible in the right most image). However, to reason about and provide assistance for the task, Jarvis must maintain a holistic representation of the current task state. To maintain a stable state representation of the task, we propose a novel approach of bringing together long-term knowledge of a world model with current perceptions. This approach is implemented in the Soar cognitive architecture and relies on its memories and inference mechanisms.

*State Representation.* Jarvis's beliefs about the current state of the task environment as a set of unary predicates defined over the set of relevant components. These predicates are of following three types:

- Real perceptual predicates: Jarvis uses predicates such as *exists(front-door)*, *closed(front-door)*, or *in(drum-cartridge)* (describing that the component *drum cartridge* is in the printer) to represent its world. While these predicates are symbolically represented in the cognitive system, they eventually ground out to sub-symbolic information in the deep learning vision system. This connection is unidirectional, i.e. the vision system can inform the cognitive system about existence of certain predicates however, the cognitive system has no influence over the perceptual system's reasoning & inference.
- Assumed perceptual predicates: When Jarvis doesn't have a complete representation of the task state, it assumes a set of perceptual predicates to be true in order to reason about the task. These predicates are resolved when there is more information available as the camera moves during task performance. For example, in Figure 2 left, the component *drum cartridge* is not visible. However, if Jarvis was asked to provide assistance for *removing the drum cartridge* task, it would assume a the predicate *closed(front_door)* and generate an instruction plan. Previous theoretical research [15] suggests using such representation for environments with incomplete information.
- Task predicates: Jarvis maintains a set of task predicates that are inferred using the current perceptual information and its long-term model of the world (described in the next section). Examples include *visible(c)* when the component *c* is in view, *accessible(c)* when component *c* can be acted upon, and *restricted(c)* when a component *c* is restricted due to some property of the state.

*World Model.* Jarvis has long-term beliefs about the its task environment that are stored in its semantic memory indexed by the printer model name. They pertain to the structure of the printer and relationships between different components. In particular, we are interested in modeling three aspects of the task environment:

- Structure: This part of the model contains a list of all the components (such as *front-door*) and their perceptual states(*[open,close]*). As we will see later, knowing what components exists in the environment is helpful in maintaining a holistic task state.
- Default state predicates: This part of the model encodes which state predicates can be assumed to be true about which components when no information is available from the vision system. For example, *front door* is assumed to be *closed.*
- Occlusion: This part of the model encodes which components what state occlude which others. For example, *front door* when *closed* occludes *drum-cartridge* and *toner-cartridge.*
- Restriction: This part of the model encodes which components in what state restrict which others. For example, when *locked*, *drum cartridge lock* restricts the movement of *drum cartridge.*

*State Inference.* Jarvis' state inference system is encoded in a set of productions in Soar's procedural memory and occurs as follows:

(1) Initialize current task state (CTS): For initializing task state corresponding to a printer, Jarvis queries for a relevant world model in its semantic memory. Then, it creates a working memory representation of all the components represented in the world model by adding

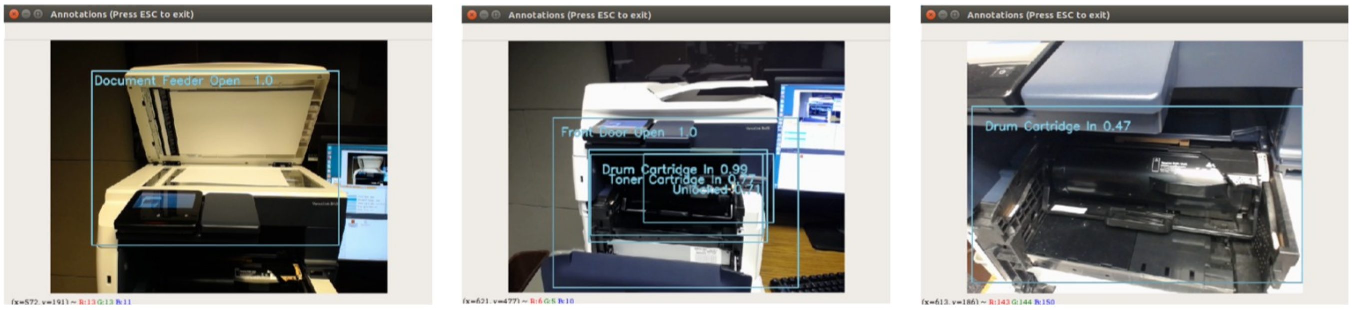


**Figure 2: Scene capture from the camera and perceptual events detected by vision classifiers**

the existence predicate for all of them. It marks the visibility of each component to be not be true. Let this working memory representation be the CTS. CTS represents the beliefs of Jarvis about the state of its task environment. Initializing the CTS is an operator-supported persistent inference and it changes only when there is deliberate decision to update it. Persistent CTS ensures that even when perceptual events flicker in the vision system, Jarvis's beliefs about the task are stable.

(2) Compare with perceptions: As the perceptual stream begin coming in, Jarvis compares its CTS with the perceptual stream. If a component in the CTS is not visible but the perceptual system is detecting a state event for the component, it marks the component visible. Similarly, if a component in visible in the CTS, but the perception system is not detecting any state event for the component, that component is marked to be not visible. This is an elaboration inference.

(3) Update current model: If a component is visible, Jarvis starts tracking any perceptual event detected related to the component and adds or removes relevant predicates from the current model. For example, if *front door* is visible, Jarvis, upon opening it the CTS will include *open(front-door)*. This is an operator-supported inference and is persistent. Consequently, if a component is not visible, Jarvis maintains the predicates from the last time it was visible. These changes are supported by operators and are persistent.

(4) Infer accessibility: If a component is visible, an elaboration rule also creates a predicate indicating that the component is also accessible to be acted upon. This inference encodes the assumption that that if a printer component is in the human's field of view (correlated with Jarvis'), they can manipulate it. If something is not in the field of view either because the human is looking somewhere else or that they are occluded by another component (*front door* occludes *toner cartridge* in the domain), the occluding component has to be removed (by opening the *front door*). This is an elaboration inference.

(5) Infer restrictability: If for a component $c$, the world model encodes that there exists another component $c'$ which in a specific state restricts $c$ and the current model has a predicate indicating $c'$ in the specific state, $c$ is considered restricted. Jarvis adds a restricted predicate for $c$ in its CTS. This is an elaboration inference as well.

Every decision cycle, Jarvis' reads on input from the vision system and performs the outlined inference. A typical decision cycle is Soar runs in about $50ms$.

## Planning Task Actions

In the previous section, we described how Jarvis maintains a stable, cohesive state representation of the task (CTS) in a partially observable environments. Maintaining a cohesive state is critical for reasoning about the task. Currently, the assistance Jarvis provides is limited to what action to do next given a task that the trainee is performing and is determined using a planning formulation.

*Background planning knowledge.* In addition to the state representation, a planning formulation requires a *goal* that is defined in the terms of the state representation and a set of *actions* defined in terms of their pre-conditions and effects. In the current variation, Jarvis knows of two tasks: *removing a drum cartridge* and *removing a toner cartridge.* These goals are represented as a composition of perceptual predicates that perceptual system can detect individually in the scene. Through an elaboration inference, Jarvis can also detect when the goal composition is true in the scene.

Actions are represented using the operator-supported inference in Soar. Operator proposal rules test for the pre-conditions of actions against the current working memory (containing CTS) and propose the action if there is a match. For example, if the CTS encodes that *front door is open*, *close(front-door)* will be proposed. Selection of a specific action operator is determined through planning and is described in the following text. Once an operator is selected, the operator application rules will change the CTS in accordance with action part of the rule. Jarvis has two different types of operator application rules that are used contextually. These contexts are called *problem spaces* in Soar. When Jarvis is planning (described in detail in the next section), the application rules encode how the world will change when the particular action operator is applied. I.e, for the action *close(front-door)*, the operator application rules encode that the resulting world will contain the predicate *front door is closed* and not contain the predicate *front door is open*. When Jarvis moves into the real-world interactive context, the operator application rules encode a communicative act. For example, for operator *close(front-door)*, the application rule will encode *inform(close, front-door)*. This communicative act is then passed to the human-agent interaction system which may use speech or visual modality for conveying this to the human trainee.

*Interactive, Contextual Planning.*

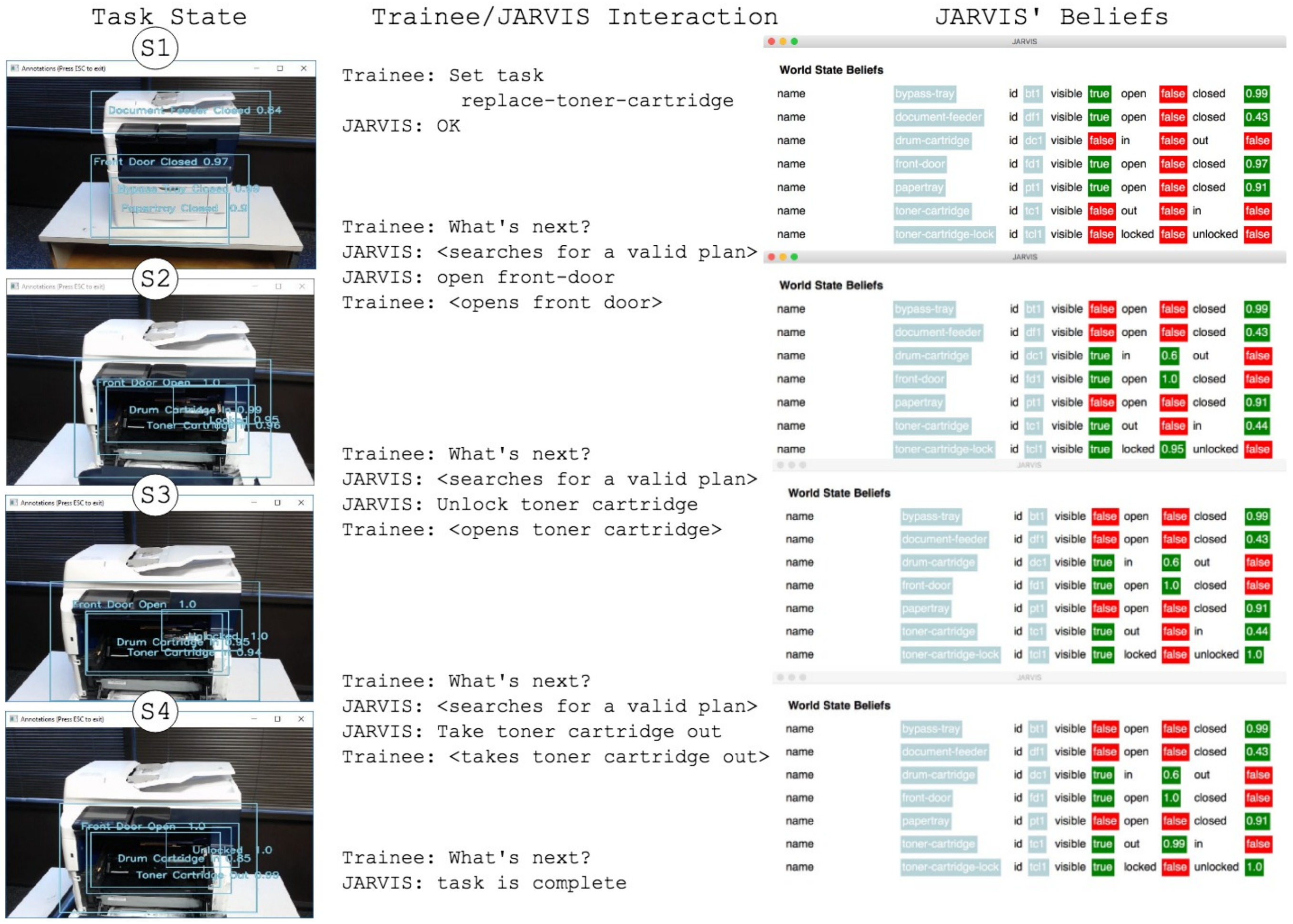
(1) Set a planning goal: The human trainee can use the *set-task* interaction to set a goal for the planner. Upon receiving this instruction, Jarvis instantiates the definition of this goal in its working memory and begins actively tracking if this composition is true in the environment. Upon asking, *whats' next?* after setting a goal, Jarvis can begin planning.
(2) Switch problem space (context): Upon being asked *what's next?*, Jarvis switches to the planning context and creates a copy of the CTS in this context.
(3) Complete the world: As we discussed earlier, Jarvis operates under partial-observability. Consequently, it could be the case that when it is asked to plan, it does not know the complete task state. If this is the case, Jarvis applies its world model to include certain predicates in CTS about objects that are not visible to assume a certain state about the world. This process results in an augmented CTS which is based on some information from the perception system and some from the world model encoded in Jarvis.
(4) Search for the plan: Once the CTS is augmented, Jarvis begins planning. The planning process in Soar is build on its *impasse* structure. In the initial CTS, several actions can be taken and consequently, several action operators are proposed. As Jarvis does not have any rules to select between the proposed operators, an impasse occurs. Due to this impasse, a sub-state is created which will persist until the impasse in the super-state is resolved (i.e. a specific action operator can be selected). In this sub-state, a copy of the CTS is created. Then, Jarvis evaluates a randomly selected action operator to see if applying that operator will result in the goal state. To do this, the selected action operator is applied. The application of the selected operator makes changes in the copied CTS in accordance with the application rules. If the resultant state does not include the goal, an impasse will result again causing sub-state creation and action evaluation to occur recursively. If the resultant CTS contains the goal, evaluation of the action operator will be marked as a success causing the impasse in the super-state to resolve as now the action evaluated a success can be applied. The success evaluation will pass up in the recursion chain eventually selecting an action in the initial CTS. Various kinds of search strategies can be applied adding heuristics about how actions are selected for evaluation as well as constraining the depth of recursion. The current implementation uses iterative-deepening search. The output of the planning process is an action-operator that when applied to the CTS will move the task closer to the selected goal. Let this be *result-operator*.
(5) Deliver instruction contextually: When the planning operator succeeds, Jarvis moves into the real-world context. In this context, application of the *result-operator* generates a communication act of informing the trainee of the action. Currently, the next recommended action is displayed in an interaction window. After completing the action in the world, the trainee can ask *what's next?* to trigger planning again. The planning process can be triggered whenever the trainee needs more assistance. A valid plan is generated for the CTS in which assistance was requested. Contextual generation of plans results in *mixed-initiative* instruction in which the trainee can choose to explore the task by themselves but can ask for support if they aren't sure how to make progress.

## 5 SYSTEM DEMONSTRATION

We can demonstrate real-time contextual instructional support from Jarvis on tasks performed in four scenarios characterized by varying trainee behavior.

*Regular.* In this scenario, the trainee accurately follows Jarvis' instruction. As shown in Figure 3 (left), the task begins in the initial state S1 and the trainee selects a task - *replace-toner-cartridge*. When prompted for the next action, Jarvis begins planning for state S1 and recommends the action *open front door* to the trainee. The trainee applies this action in the task environment and the task transitions to S2. The trainee asks for the next action. This interactive task execution continues until the goal state is detected in the environment.

*Exploratory.* In this scenario, the trainee explores the environment by taking actions beyond what was instructed. In



```
Trainee: Set task
           replace-toner-cartridge
JARVIS: OK

Trainee: What's next?
JARVIS: <searches for a valid plan>
JARVIS: open front-door
Trainee: <opens front door>

Trainee: What's next?
JARVIS: <searches for a valid plan>
JARVIS: Unlock toner cartridge
Trainee: <opens toner cartridge>

Trainee: What's next?
JARVIS: <searches for a valid plan>
JARVIS: Take toner cartridge out
Trainee: <takes toner cartridge out>

Trainee: What's next?
JARVIS: task is complete
```

**World State Beliefs**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| name | bypass-tray | id | bt1 | visible | true | open | false | closed | 0.99 | | |
| name | document-feeder | id | df1 | visible | true | open | false | closed | 0.43 | | |
| name | drum-cartridge | id | dc1 | visible | false | in | false | out | false | | |
| name | front-door | id | fd1 | visible | true | open | false | closed | 0.97 | | |
| name | papertray | id | pt1 | visible | true | open | false | closed | 0.91 | | |
| name | toner-cartridge | id | tc1 | visible | false | out | false | in | false | | |
| name | toner-cartridge-lock | id | tcl1 | visible | false | locked | false | unlocked | false | | |

**World State Beliefs**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| name | bypass-tray | id | bt1 | visible | false | open | false | closed | 0.99 |
| name | document-feeder | id | df1 | visible | false | open | false | closed | 0.43 |
| name | drum-cartridge | id | dc1 | visible | true | in | 0.6 | out | false |
| name | front-door | id | fd1 | visible | true | open | 1.0 | closed | false |
| name | papertray | id | pt1 | visible | false | open | false | closed | 0.91 |
| name | toner-cartridge | id | tc1 | visible | true | out | false | in | 0.44 |
| name | toner-cartridge-lock | id | tcl1 | visible | true | locked | 0.95 | unlocked | false |

**World State Beliefs**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| name | bypass-tray | id | bt1 | visible | false | open | false | closed | 0.99 |
| name | document-feeder | id | df1 | visible | false | open | false | closed | 0.43 |
| name | drum-cartridge | id | dc1 | visible | true | in | 0.6 | out | false |
| name | front-door | id | fd1 | visible | true | open | 1.0 | closed | false |
| name | papertray | id | pt1 | visible | false | open | false | closed | 0.91 |
| name | toner-cartridge | id | tc1 | visible | true | out | false | in | 0.44 |
| name | toner-cartridge-lock | id | tcl1 | visible | true | locked | false | unlocked | 1.0 |

**World State Beliefs**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| name | bypass-tray | id | bt1 | visible | false | open | false | closed | 0.99 |
| name | document-feeder | id | df1 | visible | false | open | false | closed | 0.43 |
| name | drum-cartridge | id | dc1 | visible | true | in | 0.6 | out | false |
| name | front-door | id | fd1 | visible | true | open | 1.0 | closed | false |
| name | papertray | id | pt1 | visible | false | open | false | closed | 0.91 |
| name | toner-cartridge | id | tc1 | visible | true | out | 0.99 | in | false |
| name | toner-cartridge-lock | id | tcl1 | visible | true | locked | false | unlocked | 1.0 |

**Figure 3: Task states in execution of *remove toner cartridge*, trainee-Jarvis interaction, and Jarvis' beliefs about the states of different components in the task environment.**

S1 (Figure 3), after executing *open front-door*, the trainee *unlocks the toner-cartridge* without any prompting from Jarvis, consequently landing in s3. Jarvis is able to handle this independent action execution by the trainee and when prompted for the next action responds correctly by suggesting *take toner cartridge out*.

*Interruptive.* In this scenario, the trainee interrupts Jarvis' planning operation by executing an unprompted action. In S1 (Figure 3), as Jarvis is planning, the trainee *opens the front-door* transitioning the task to s2. Jarvis maintains the context of its planning while it is executing the iterative-deepening search. As the context changes due to the trainee's actions, Jarvis abandons its search and initiates a new search with the current task state. This correctly produces the relevant action open toner cartridge.

*Incorrect.* In this scenario, the trainee misinterprets the instruction. In S1 (Figure 3), upon being instructed to *open front-door*, the trainee *opens the document feeder* instead. While Jarvis is correctly able to reason that the task state still is S1 and determine the next action *open front-door*, from a training perspective this instruction is not very productive. A good trainer can recognize this as a learning opportunity and provide more information about the structure of the printer. For example, the trainer can respond *That is the document feeder. The front-door is in the front, towards the bottom.* Currently, Jarvis lacks the representations and reasoning for such instruction.

These observations of Jarvis' behavior directly follow from the approach implemented and are arguably trivial. However, they are very encouraging because they demonstrate that these two kinds of architectures can be brought together for reliable behavior in the real-world. The deep learning architecture can robustly extract state information about various components of the machine and Soar can integrate these perceptual events for a stable state representation and incorporate structural knowledge of the world for planning. Further, these observations illustrate the need for learner models and relevant pedagogical strategies for reasonable instructional behavior.

## 6 CONCLUSION

In this paper, we described the initial steps we have taken to build Jarvis, an intelligent system that can help people learn physical tasks via apprenticeship training. The use of augmented reality provides an opportunity to provide instructional support within the context of task performance. For learning procedural tasks in physical domains, this training can be very effective in comparison to classroom teaching in which the student must expend effort to translate theoretical knowledge to the performance context.

From an AI systems perspective, design of such instruction systems is a significant challenge. The requirement of robust, flexible, real-time instructional behavior in face of dynamic environment, necessitates computational formulations that can reason online. We take an hybrid approach towards this system design where we bring together deep learning architectures for computer vision and high-level reasoning architecture Soar for online, real-time, adaptive instructional support in the real-world. We demonstrate the robustness our approach by observing system behavior in scenarios with different trainee behavior. There are several challenges that have not been addressed in this preliminary work; from maintaining object identities during replacement tasks to modeling the state of the trainee to provide personalized training. Our future work will study these challenges and propose hybrid approaches similar to the one described in this paper.

Finally, this paper sets the stage for future research in design of intelligent technology for apprenticeship training. There is a long, rich history of using intelligent technology when combined with cognitive modeling to support human learning [5] that have achieved closed to human-level teaching performance [19]. However, a significant majority of research threads pursued have looked at augmenting academic learning in classroom environments, often focusing on learning coursework algebra, sciences, and programming. Approaches such as ours will greatly enhance the impact of intelligent tutoring technology by bringing it to the context for workplace training as well as for end-users of complex machines.

## REFERENCES

[1] John Bransford. 2000. *How People Learn: Brain, Mind, Experience, and School.* National Academies Press.

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. 2010. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision* 88, 2 (June 2010), 303–338.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *Conference on Computer Vision and Pattern Recognition* (2015).

[4] J Kirk and J Laird. 2014. Interactive Task Learning for Simple Games. *Advances in Cognitive Systems* 3 (2014), 11–28.

[5] Kenneth R. Koedinger, Emma Brunskill, Ryan S.J.d. Baker, Elizabeth A. McLaughlin, and John Stamper. 2013. New Potentials for Data-Driven Intelligent Tutoring System Development and Optimization. *AI Magazine* 34, 3 (9 2013), 27. https://doi.org/10.1609/aimag.v34i3.2484

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012.

[7] John E. Laird. 2012. *The Soar Cognitive Architecture.* MIT Press.

[8] Y. LeCun, F.J. Huang, and L. Bottou. 2004. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition CVPR*, Vol. 2.

[9] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. SSD: Single Shot MultiBox Detector. In *ECCV.*

[10] Aaron Mininger and John Laird. 2018. Interactively Learning a Blend of Goal-based and Procedural Tasks. In *Proceedings of the Thirty Second AAAI conference on Artificial Intelligence.*

[11] Shiwali Mohan, James Kirk, and John Laird. 2013. A Computational Model for Situated Task Learning with Interactive Instruction. In *Proceedings of the Twelfth International Conference on Cognitive Modeling.*

[12] Shiwali Mohan, James R Kirk, Aaron Mininger, and John Laird. 2012. Acquiring Grounded Representations of Words with Situated Interactive Instruction. *Advances in Cognitive Systems* (2012).

[13] Shiwali Mohan and John E Laird. 2014. Learning Goal-Oriented Hierarchical Tasks from Situated Interactive Instruction. In *Proceedings of the Twenty Eighth AAAI conference on Artificial Intelligence.* AAAI Press.

[14] Shiwali Mohan, Aaron Mininger, and John Laird. 2014. Towards an Indexical Model of Situated Language Comprehension for Cognitive Agents in Physical Worlds. *Advances in Cognitive Systems* (2014).

[15] Raymond Reiter. 1980. A logic for default reasoning. *Artificial intelligence* 13, 1-2 (1980), 81–132.

[16] Matthew Shreve, Sricharan Kumar, Jin Sun, Gaurang Gavai, Robert R Price, and Hoda Eldardiry. 2017. Augmented Reality For Efficient Collection Of Training Data For Machine Learning. (Aug 2017).

[17] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations* (2015).

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going Deeper with Convolutions. *IEEE Conference on Computer Vision and Pattern Recognition* (2015).

[19] Kurt Vanlehn. 2011. Educational Psychologist The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *Educational Psychologist* 46, 4 (2011), 197–221. https://doi.org/10.1080/00461520.2011.611369

[20] Lev Vygotsky. 1978. *Mind in Society.*

[21] James V. Wertsch. 1979. From Social Interaction to Higher Psychological Processes. *Human Development* 22 (1979), 1–22.