PrISM-Tracker: A Framework for Multimodal Procedure Tracking Using Wearable Sensors and State Transition Information with User-Driven Handling of Errors and Uncertainty

RIKU ARAKAWA, Carnegie Mellon University, United States
HIROMU YAKURA, University of Tsukuba, Japan
VIMAL MOLLYN, Carnegie Mellon University, United States
SUZANNE NIE, Carnegie Mellon University, United States
EMMA RUSSELL, Case Western Reserve University, United States
DUSTIN P. DEMEO, Case Western Reserve University, United States
HAARIKA A. REDDY, Case Western Reserve University, United States
ALEXANDER K. MAYTIN, Boston University, United States
BRYAN T. CARROLL, University Hospitals of Cleveland Department of Dermatology, United States
JILL FAIN LEHMAN, Carnegie Mellon University, United States
MAYANK GOEL, Carnegie Mellon University, United States

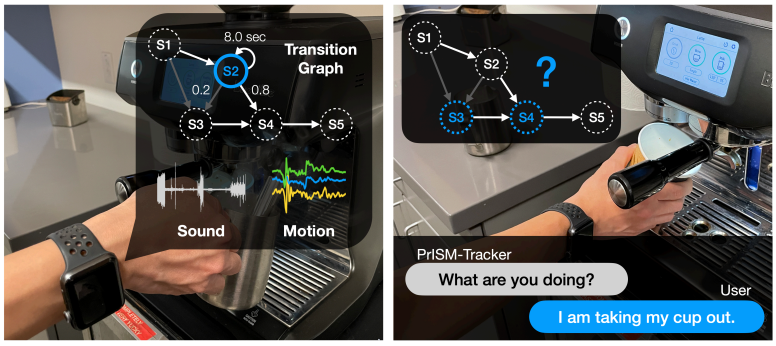

Fig. 1. PrISM-Tracker tracks a user's steps in procedural tasks using a multimodal (sound and motion) activity recognition system augmented by state transition information (left). PrISM-Tracker identifies moments of uncertainty and proactively asks the user for speech prompts to improve performance (right).

Authors' addresses: Riku Arakawa, rarakawa@cs.cmu.edu, Carnegie Mellon University, Pittsburgh, United States; Hiromu Yakura, University of Tsukuba, Tsukuba, Japan; Vimal Mollyn, Carnegie Mellon University, Pittsburgh, United States; Suzanne Nie, Carnegie Mellon University, Pittsburgh, United States; Emma Russell, Case Western Reserve University, Cleveland, United States; Dustin P. DeMeo, Case Western Reserve University, Cleveland, United States; Haarika A. Reddy, Case Western Reserve University, Cleveland, United States; Alexander K. Maytin, Boston University, Boston, United States; Bryan T. Carroll, University Hospitals of Cleveland Department of Dermatology, Cleveland, United States; Jill Fain Lehman, Carnegie Mellon University, Pittsburgh, United States; Mayank Goel, Carnegie Mellon University, Pittsburgh, United States.

2474-9567/2022/12-ART156
https://doi.org/10.1145/3569504

A user often needs training and guidance while performing several daily life procedures, *e.g.*, cooking, setting up a new appliance, or doing a COVID test. Watch-based human activity recognition (HAR) can track users' actions during these procedures. However, out of the box, state-of-the-art HAR struggles from noisy data and less-expressive actions that are often part of daily life tasks. This paper proposes *PrISM-Tracker*, a procedure-tracking framework that augments existing HAR models with (1) graph-based procedure representation and (2) a user-interaction module to handle model uncertainty. Specifically, PrISM-Tracker extends a Viterbi algorithm to update state probabilities based on time-series HAR outputs by leveraging the graph representation that embeds time information as prior. Moreover, the model identifies moments or classes of uncertainty and asks the user for guidance to improve tracking accuracy. We tested PrISM-Tracker in two procedures: latte-making in an engineering lab study and wound care for skin cancer patients at a clinic. The results showed the effectiveness of the proposed algorithm utilizing transition graphs in tracking steps and the efficacy of using simulated human input to enhance performance. This work is the first step toward human-in-the-loop intelligent systems for guiding users while performing new and complicated procedural tasks.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing systems and tools**; *Interactive systems and tools.*

Additional Key Words and Phrases: activity recognition, procedure tracking, human in the loop, human-AI interaction

ACM Reference Format:
Riku Arakawa, Hiromu Yakura, Vimal Mollyn, Suzanne Nie, Emma Russell, Dustin P. DeMeo, Haarika A. Reddy, Alexander K. Maytin, Bryan T. Carroll, Jill Fain Lehman, and Mayank Goel. 2022. PrISM-Tracker: A Framework for Multimodal Procedure Tracking Using Wearable Sensors and State Transition Information with User-Driven Handling of Errors and Uncertainty. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 4, Article 156 (December 2022), 27 pages. https://doi.org/10.1145/3569504

1 INTRODUCTION

We perform numerous procedures or sequences of steps in our daily life [47]. For example, installing new appliances, assembling furniture, cooking, medical self-care, *etc.* We are adept at a few of these tasks, and for many, we need support. One popular form of support is instruction manuals, but these are often inadequate. Another popular form includes demonstration videos but they are often hard to find and do not adapt to the user's situation. One domain where the user finds the support especially lacking is medical self-care. After leaving the clinic, patients may neglect self-care, call provider helplines, use social media platforms, or watch YouTube videos when they have questions [16]. In the case of helplines, patients either end up calling prematurely or wait for too long before seeking expert guidance [62].

An intelligent agent that navigates a user through procedural tasks where there is a set of instructions is therefore of significant importance. While researchers have proposed camera-based approaches to track users' actions in procedures [23], cameras raise significant cost and privacy issues. At the same time, researchers have proposed ways of using other sensor modalities for recognizing human activities [64]. Specifically, human activity recognition (HAR) using watches has shown promising results [66]. The ubiquity and accessibility of these watches make such approaches more private, portable, mobile, practical, and usable than cameras. The biggest challenge though is accuracy. Motion or sound signals as captured by a wrist-worn watch are not going to be as expressive as a video. Recent advances in HAR definitely help; especially the multimodal approaches that combine audio and motion signals [8, 40]. However, most sensor-based HAR studies have been primarily proposed for detecting either coarse and isolated activities such as running and walking, or highly expressive actions such as knocking on doors and using noisy machines. In the real world, procedural tasks often contain ephemeral and inexpressive steps. For instance, Spriggs *et al.*'s dataset on cooking brownies recipe [56] included several steps that are hard to detect with wearable sensors, such as walking up to a counter and getting a fork, which lasted for less than a few seconds. As a result, directly applying conventional HAR models to estimate frame-by-frame user activity has not reached practical accuracy in procedure tracking (*i.e.*, most of them falling below 70%) [7, 56].

This paper presents *PrISM-Tracker*, a multimodal architecture for tracking **Pr**ocedural **I**nformation using **S**ound and **M**otion captured by a watch. The framework augments conventional human activity recognition models in two ways: (1) using a transition graph that embeds sequence information as prior and (2) user-driven handling of errors and uncertainty (See Figure 1). First, PrISM-Tracker reduces noisy inferences with a Viterbi algorithm-based graphical and temporal representation of the transitions between steps [17]. Then, to further handle the errors and uncertainty while working with real-world procedures, the architecture supports a human-in-the-loop approach that identifies moments of uncertainty and asks for user help. The architecture optimizes when to use human input based on its graph representation of the probability that the user is at each step of the procedure. PrISM-Tracker also outputs the performance gain and user-burden trade-offs to allow developers to optimize the approach for their procedures. PrISM-Tracker is modular, and developers can switch to different HAR models as long as the data is in the supported format.

To evaluate PrISM-Tracker, we use two tasks: latte-making in an engineering lab and wound care on skin cancer patients in a clinic. We chose these two tasks to have a breadth of application domains, user population, environmental noise, and user training levels. Latte-making involved 19 distinct steps. Fifteen participants made a latte using an espresso machine while they wore a smartwatch that collected synchronized audio and motion data. PrISM-Tracker tracked users' actions with 71.0% frame-level accuracy and 52.9% F1-score in the condition of real-time estimation, while conventional HAR (based on [40]) without any graphical and temporal information achieved 57.1% and 39.7%, respectively. Next, we identified moments of uncertainty for the model and simulated perfect human prompts (*i.e.*, oracles) at those times. Assuming a user gave the correct prompts at steps specified by PrISM-Tracker, we saw 4.4% and 9.3% improvement with two human responses for frame-level accuracy and macro F1-score, respectively.

To examine how PrISM-Tracker performs in a real-world situation in a noisier task with a less trained user population, we recruited skin cancer patients to perform a wound care procedure on themselves in a clinic. With data from 23 participants performing a procedure with 12 steps, PrISM-Tracker achieved 56.7% frame-level accuracy and 50.7% F1-score, while conventional HAR achieved only 37.3% and 27.7%. PrISM-Tracker handled the moments of uncertainty with oracles. These human input-based oracles are an upper bar for performance, as human input will not work 100% of the time. The oracles improved the frame-level accuracy and macro F1-score by 15.6% and 18.9% with just two prompts. This result shows a big improvement over an HAR model and highlights the inadequacy of state-of-the-art HAR models with no human input while tracking real-world tasks. We acknowledge that the assumed 100% effectiveness of human input is impractical. Thus, we also examined the effectiveness of speech as a medium for users to interact with the system. We used the patients' narration speech and achieved 85.1% accuracy by using a BERT-based model [14]. Overall, this clinical study demonstrated the effectiveness of PrISM-Tracker in the real world as well as a promise of it being used as a real-time system with human speech as an interaction medium.

While tracking by wearable devices has a huge potential for the systems to be ubiquitously used in our daily lives, current HAR techniques become easily noisy in the real world. This paper presents a framework for developing such systems by introducing a novel human-in-the-loop architecture in multimodal HAR. We share the code for the framework as well as the dataset (https://github.com/cmusmashlab/prism-tracker) to help other researchers and developers build a tracker for their procedural tasks. However, tracking a user's actions is only the first step in guiding them through complicated procedures. In the future, the research team will continue to work with different user populations and application domains to evolve PrISM-Tracker from a passive tracker to a responsive agent that tracks steps and their quality, alerts the user if they miss something, and generates performance improvements when needed (*e.g.*, training nurses or health workers).

2 RELATED WORK

Our work contributes to approaches to tracking procedures using smartwatches. We first review procedure tracking systems. Then, we discuss existing works on procedure tracking using smartwatches, highlighting its difficulty compared to conventional HAR tasks. Lastly, we review previous works on human-in-the-loop systems since we are inspired by their idea to address the difficulty of procedure tracking with user-driven handling of errors and uncertainty of HAR.

2.1 Systems Based on Procedure Tracking

Researchers have several systems to support users in various procedural tasks such as cooking [52, 61], crafting [23, 53, 69], and medical care support [7, 43]. Uriu *et al.* [61] proposed a sensor-embedded frying pan that provides users with context-aware information about a recipe. Sato *et al.* [52] further proposed a system for guiding users along a recipe in cooking. They used a depth camera to track users' actions and a projector to present step-by-step instructions to users. Their system recognized cooking steps based on visual information and asked users to use predefined gestures to transition between steps.

Augmented reality (AR) is also gaining popularity as a medium to provide feedback to the user. As one of the early works, Servan *et al.* [53] proposed an AR-based training system for assembly, in which users could see assembly instructions for a gully trap through AR glasses. At the same time, users needed to transition between steps in Servan *et al.*'s system manually. Yamaguchi *et al.* [69] presented a system for visualizing interactive 3D Augmented Reality tutorials based on 2D video input by utilizing vision-based object tracking techniques. AdapTutAR [23] is an adaptive task tutoring system that monitors learners' tutorial-following status and provides feedback. Since AR technologies are entwined with vision sensors, these works utilize computer-vision approaches to track users' activity throughout procedures.

There are several other domains where procedure tracking can be useful. For example, there are systems to track exercise by a smartwatch [42] or an ambient camera in a gym [27]. Moreover, the needs become remarkable in healthcare domains, such as self-care [35], where users often perform procedures to preserve their health alone and at home in difficult conditions such as having a wound [54]. For instance, Motahar *et al.* [43] conducted a qualitative study to explore patients' current self-care behavior. They found that their understanding of how and when to perform self-care differed by the individual; ideally, physicians should be able to monitor and discuss it with the patients. Thus, procedure tracking would be beneficial in terms of offering the physicians a means to monitor self-care, *e.g.*, whether a patient conducts the care every day, or how much time they spend at each step.

While these previous works suggest the needs and benefits of procedure tracking, most of them rely on vision-based sensing. One advantage of vision-based approaches is their accuracy in detecting various actions with the power of recent data-driven techniques like deep learning. However, these approaches often suffer from privacy and power consumption issues. Moreover, preparing cameras every time users conduct a procedure can be burdensome. In response, there have been emerging sensing techniques without using cameras for recognizing events in human activity recognition (HAR). In the next section, we review the field of HAR and its applicability to procedure tracking.

2.2 HAR and Procedure Tracking

Human activity recognition (HAR) aims to recognize different human actions from a sequence of sensor observations. It is a field with extensive prior research covering many approaches involving various sensors. For full review, please refer to [11, 24, 34]. Since we aim to develop systems with everyday wearables, we focus on reviewing motion- and audio-based HAR techniques [8, 46]. Such techniques involve HAR based on microphones [25, 32, 57] and IMUs [6, 31, 33, 36, 39, 72]. For example, Laput *et al.* [32] proposed a deep-learning-based approach to detecting 30 daily activities based on audio, such as coughing and typing. The trained model achieved

84.1% accuracy for classifying data on smartwatches. Becker *et al.* [6] proposed combining audio and motion sensing for recognizing gestures such as clapping and knocking using smartwatches. Their model achieved 97.2% accuracy in classifying nine distinct gestures. Given the practical accuracy of detecting specific activities, commercial products incorporate HAR these days, such as Apple Watch's handwashing detection [3].

Given the high accuracy in classifying various activities, we expect that such HAR techniques can also be used for tracking procedures by individually detecting each step. However, in contrast to most of the existing HAR works that detect isolated activities containing definitive signals, steps in everyday tasks are less distinctive and more noisy [58].

The most straightforward approach to procedure tracking is to directly apply HAR models to classify incoming data into steps during a procedure. For example, Spriggs *et al.* [56] collected data on a cooking procedure consisting of 29 steps as a brownies recipe. They classified IMU data collected at e-watches worn on users' wrists into steps, achieving 57.8% accuracy. Bernal *et al.* [7] proposed medical procedure monitoring using wearable sensors. They tracked the self-injection procedure consisting of seven steps, such as sanitizing hands and injecting insulin. Their model classifying segmented motion data into seven steps achieved 57% accuracy.

Researchers have applied filtering techniques to the frame-by-frame prediction to leverage the temporality of procedures. Korematsu *et al.* [30] proposed an approach based on acoustic event detection for tracking a cooking procedure, recognizing cutting, grilling, and other steps. Their frame-level prediction with a smoothing filter achieved roughly 70% classification accuracy in the three-class classification Xia *et al.* [67] proposed using a particle filter for factory activity recognition by assuming each step happens in a fixed order. Moreover, a few works tried to leverage the transition relationship between steps. For example, Kojima *et al.* [29] proposed a multimodal cooking recognition system while utilizing recipes as background knowledge represented in Hierarchical Hidden Markov Model. They achieved 63% accuracy in classifying 11 actions, such as washing and cutting vegetables. In addition, Nakauchi *et al.* [45] presented a system that predicts a cooking procedure based on a Markov model by explicitly considering transition probabilities between steps.

This prior work suggests that procedure tracking based on wearable sensing is less accurate than conventional HAR tasks. The difficulty lies in that some steps are hard to detect solely by sensors and that the steps are performed continuously without explicit start and end separations. As a result, the raw prediction of windowed data using HAR approaches can get noisy. This is in contrast to the case of conventional HAR problem settings, where steps are isolated and often have sensor signals clear and consistent enough to detect (*e.g.*, washing hands). Therefore, we aim to enable procedure tracking with reasonable accuracy by proposing a novel framework. Specifically, our key technical contributions are (1) the use of transition relationship and temporal information to refine the output of windowed HAR predictions and (2) an architecture to incorporate human prompts efficiently to correct HAR errors and uncertainty.

2.3 Human-in-the-loop Systems

Previous works have leveraged human as a data source to intelligent systems with various purposes such as accelerating models' training [4, 9, 41], correcting models' predictions interactively [5, 49], customizing or adjusting systems to users' environments or preference [12, 19, 59], and constructing better interactions between human and AI [2, 18]. Such approaches are called interactive machine learning [1, 15] or human-in-the-loop machine learning [18, 51, 68]. The approach can be beneficial in solving computationally challenging problems, such as cases without much data for training models through interactions with humans and cases where a model's performance cannot reach a sufficient level by itself [22].

While the popular approach in this area is to facilitate agents' learning by involving the users in the process [4, 13, 71], some works have involved humans to handle errors and uncertainty of models. Arakawa and Yakura *et al.* [5] developed a speech annotation tool where human workers correct models' recognition outputs of their uttered

speech interactively to finalize the transcription. Hatori *et al.* [21] presented a method for robots to resolve instruction ambiguity through dialogue to recognize user context correctly. Unhelkar *et al.* [60] also emphasized the importance of communication between agents and humans, and presented a computational framework that decides if, when, and what to communicate to reduce user burden.

Our key idea of involving humans in procedure tracking was inspired by these prior works to support machine learning models through human inputs to handle errors and uncertainty in the prediction. It is, thus, necessary to develop an architecture that can refine its outputs with a human-in-the-loop. Here, as previous works [2, 60] emphasized, we cannot involve human prompts limitlessly, and systems need to utilize them efficiently and optimize how and when to involve them. Thus, it is desirable to enable developers to see trade-offs between end-user load and performance gain in designing their own systems. Importantly, the advantage of PrISM-Tracker is not only its performance but also the flexibility in allowing such a control, which is emphasized as one of the key aspects in constructing better human-AI interactions [2].

3 PRISM-TRACKER FRAMEWORK

In this section, we formulate our framework for procedure tracking using multimodal sensing and a transition graph of steps in a procedure. Then, we describe ways to improve tracking accuracy by incorporating human inputs into its architecture. We also show how to optimize moments when the system asks for user input. PrISM-Tracker aims to support real-time inferences and enable context-aware interactions as an intelligent agent, such as navigating cooking instructions. To enable real-time interactions, PrISM-Tracker must be able to run online. Thus, we implemented an algorithm that applies window-by-window processing to the data stream instead of offline batch processing. Also, our algorithm needs to be fast enough; the processing time for one window must be less than the window stride to avoid dropping frames.

3.1 Architecture Overview

Figure 2 shows the overview of our architecture. It takes sensor inputs consisting of synchronized audio and motion data collected on a smartwatch. The architecture processes the incoming data using a window with overlap. An HAR model processes the windowed data to output the likelihood of the windowed data of that moment belonging to each step, *i.e.*, frame-level prediction. The interval between two consecutive frames is equivalent to the length of the stride of the window. Next, the architecture estimates the transition of the steps based on the frame-level likelihood and the procedure's transition graph. The transition graph contains the transition probability and time information (*i.e.*, how much time users usually spend in each step), which we obtain through a data-driven manner. To utilize the transition graph effectively, we introduce a novel approach inspired by the Viterbi algorithm that updates the likelihood periodically on new frame data [17]. Our evaluations show that this algorithm significantly improves tracking accuracy. Furthermore, our architecture can utilize human inputs to enhance the tracking performance, namely, interactive oracle requests. Even after using the transition graph, some of the steps in procedures remain hard to detect, given the insufficient frame-level HAR accuracy in procedure tracking. Thus, PrISM-Tracker can update its hidden states by leveraging oracles provided by a user. Our architecture can learn to identify steps where human prompts significantly help the model's prediction. We design several kinds of interactions in which users can provide oracles, and the system can calculate the expected performance gain by varying the numbers and types of the oracles. This will help developers understand trade-offs between user labor and performance, such as how many requests are enough to achieve their desired accuracy. We will describe each architecture component in detail in the following subsections.

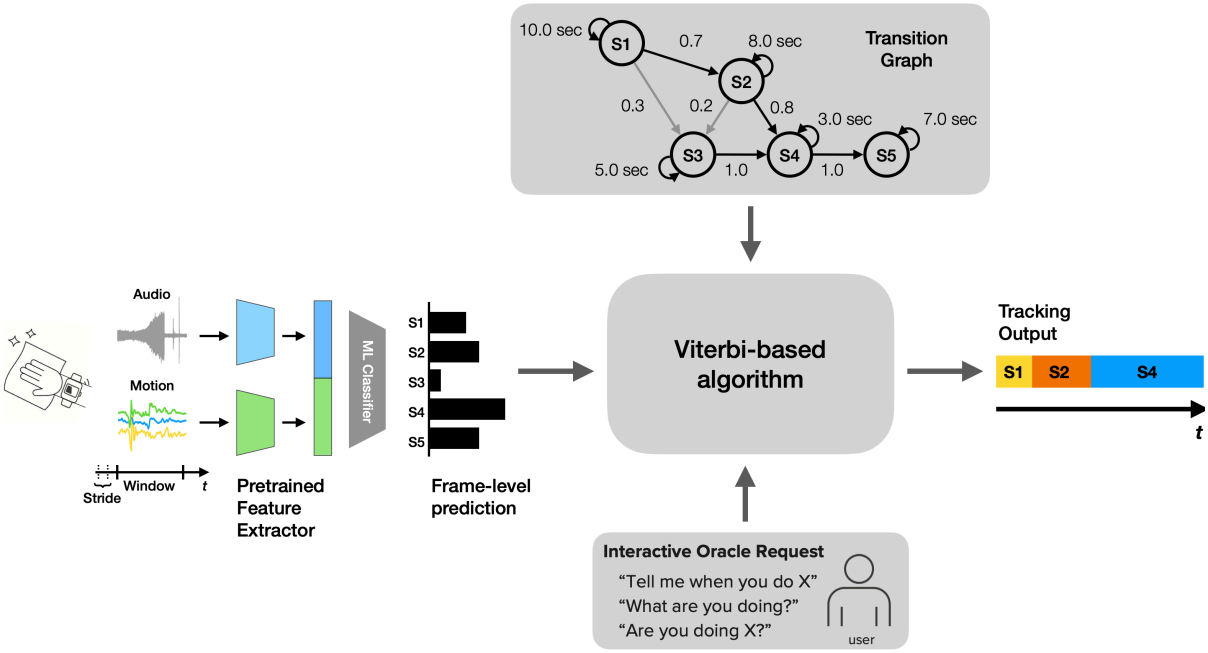

Fig. 2. Overview of PrISM-Tracker architecture. To improve the robustness and handle errors in outputs of HAR applied to procedure tracking, we use a transition graph and simulated oracles provided by users.

3.2 Frame-level Multimodal Sensing

We first trained CNN-based neural networks as feature extractors for audio and motion data individually using existing datasets [32, 40]. These HAR datasets contain data collected from smartwatches for everyday activities such as wiping and hand washing, covering a broad spectrum of signals in human activities. The structures of the trained networks are shown in Figure 3. We fine-tuned a publicly available model to train the audio model [32]. We replaced the last fully connected layer with a fully connected layer with 27 nodes. This model was then retrained on Mollyn *el al.*'s dataset [40] to output the probabilities for its 27 classes. For the motion model, we trained a model using Mollyn *el al.*'s dataset. After training these networks, we used their pre-final layer outputs as extracted features for further downstream tasks. The dimensions of the features for audio and motion data are 128 and 256, respectively. The window sizes for audio and motion signals are 2.88 seconds and 2.0 seconds, respectively. The window stride is 0.2 seconds for both signals, and this is the length for one frame.

Note that we trained feature extractors independently from steps and procedures we used for the experiments. In other words, it is necessary to collect data for a given procedure to train a classifier for the frame-level prediction of the steps. Here, we used a random forest classifier implemented in scikit-learn [48] v1.1.3 with default parameters. The model outputs probabilities of the data belonging to each step, and we regard these values as frame-level predictions.

3.3 Tracking Transition with Viterbi-Based Algorithm

Frame-level predictions of HAR systems in procedure tracking can get noisy quickly. In contrast to previous works that used filtering techniques [30, 67], we present a new approach based on the Viterbi algorithm that makes use of a transition graph of the procedure. Here, the transition graph provides: (1) the probabilities of

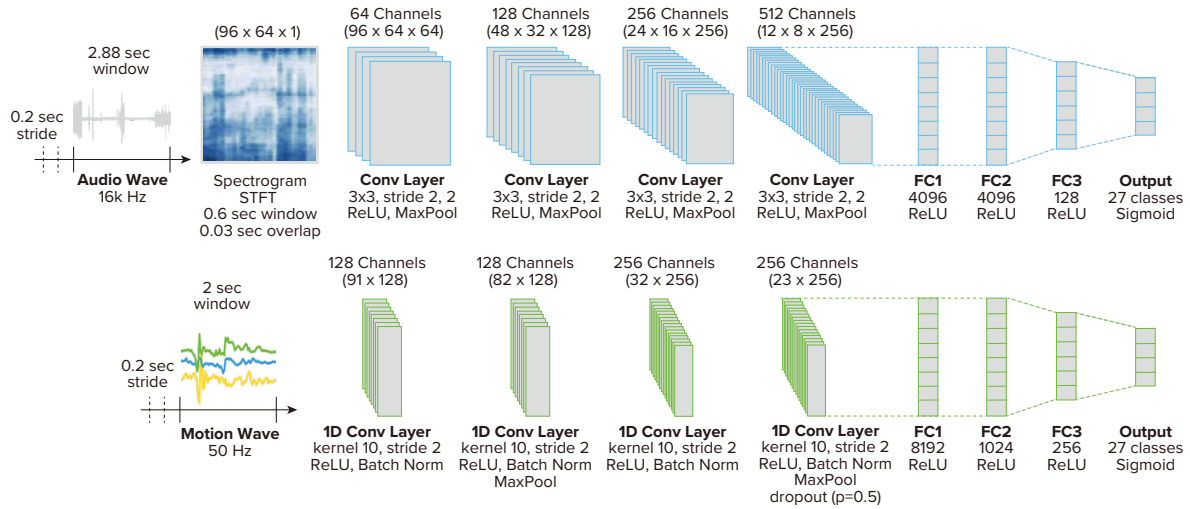

Fig. 3. PrISM-Tracker uses CNN-based neural networks as feature extractors for audio and motion data of human activities.

transition between steps; and (2) the time distribution of how long users spend on each step. This information was not used in previous studies that consider transition relationships [29, 45].

Viterbi algorithm [17] estimates the most likely sequence of hidden states from observed inputs. In our case, the hidden states and the observed inputs correspond to the possible steps and their predicted probabilities obtained by the Random Forest classifier; then, we can get the most likely sequence of the steps. Here, we can explicitly incorporate the transition graph of the procedure by eliminating the possible transitions between the hidden states to those available in the graph.

Still, the naïve Viterbi algorithm cannot handle the time information because it assumes the Markov property in the sequence of hidden states (*i.e.*, the transition between hidden states should be independent of past states). Thus, we extended the algorithm to explicitly model how long average users spend on each step. As shown Figure 4, when we have the predicted probabilities over T frames and want to estimate the transition of S steps from them, our algorithm prepares $S \times T$ hidden states, each labeled as $s(i,t)$ $(1 \leq i \leq S, 1 \leq t \leq T)$. Here, $s(i,t)$ denotes that the user has spent *at least* t frames on the i-th step. Then, assuming that $p(i,t)$ yields the probability that an average user spent at least t frames on the i-th step, we can calculate the probability that the user transitioned from $s(i,t)$ to $s(i,t+1)$ with $\frac{p(i,t+1)}{p(i,t)}$. If we model the variability of the time spent on each step with normal distribution, $p(i,t)$ is calculated as $1 - \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{t-\mu_i}{\sigma_i\sqrt{2}}\right)\right]$ using the mean μ_i and standard deviation σ_i of the duration with the Gauss error function erf $(\cdot)$.

Our algorithm then merges this information with the information about the inter-step transitions provided by the transition graph. The transition graph informs the probability of the transition between all possible pairs of $s(i)$ and $s(j)$. In the example of Figure 4, the probability of the transition from $s(1)$ to $s(2)$ is 0.7 while that of the transition from $s(1)$ to $s(3)$ is 0.3. Then, the probability that the user transitioned from $s(1,t)$ to $s(3,1)$ is calculated as $0.3 \times \left(1 - \frac{p(1,t+1)}{p(1,t)}\right)$. This Viterbi correction estimates the sequence of steps in real-time by updating the hypotheses of step sequences by incorporating newly predicted probabilities in an online manner. Also, the computational complexity of this algorithm can serve real-time estimation, as it can be denoted as $O\left(T^2 \times (S+E)\right)$ given that E is the number of possible transitions between the steps. In other words, regular computers and

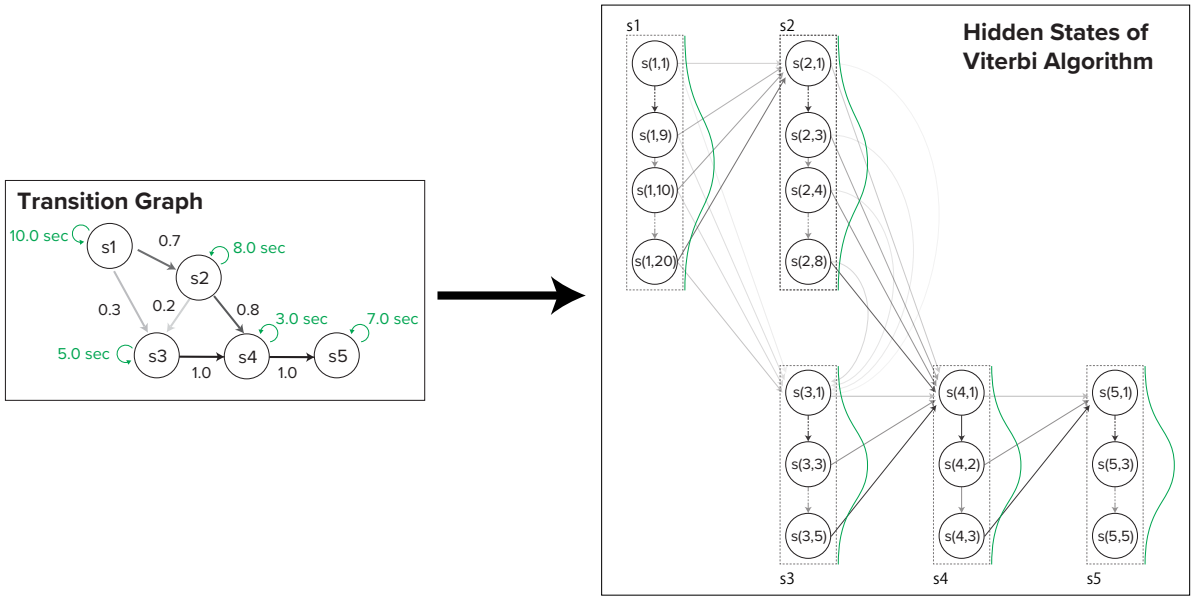

Fig. 4. Illustrative example of how our algorithm converts the given transition graph into the relation of hidden states.

smartwatches can estimate the sequence consisting of 50 steps from the probability data of 10^4 frames in the online estimation.

3.4 Interactive Oracle Request

Incorporating the transition probabilities and timing information still does not provide usable performance for most tasks. Thus, PrISM-Tracker supports adding human input whenever the model identifies a moment of uncertainty. For most of the paper, we treat the human response as an oracle and assume 100% effectiveness. We take this approach because we are yet to test the human input in real-time in our studies. This is important future work for this project. To show the initial feasibility of getting useful human input, we show the effectiveness of voice input for one of our studies in Section 5.3. We can consider several kinds of oracle requests, and there is a trade-off between the cognitive load imposed on users and the performance improvement led by oracle. Developers can experiment and choose the appropriate way of getting human input by balancing performance and user burden. In this section, we propose three possible ways to get human input.

3.4.1 "Tell Me When You Do X.". We first consider the case where PrISM-Tracker requests users to indicate when they start a specific step. Given this oracle (*e.g.*, the user's saying "I am starting the step X"), the algorithm updates its hypotheses to those currently on the hidden state $s(X, 1)$. In addition, this oracle request allows the algorithm to infer that, until the oracle has not been provided, the user has not reached step X. No doubt, this is a rigid assumption as the user might forget to tell the system. In the future, we will conduct studies to evaluate the usability of this approach. The current paper focuses on the theoretical effectiveness of the proposed approach.

Having the ability to utilize human inputs does not mean that querying is always necessary or desirable. In particular, we want to query when the other sources of information available – audio, motion, graph – do not combine into a high-accuracy prediction. We get an oracle input in two cases: (1) when a step exhibits low prediction accuracy by the Random Forest model; (2) when a step almost splits the graph into multiple branches.

PrISM-Tracker simulates the expected performance improvement due to oracle responses and optimizes the best set of steps in a data-driven manner. This allows developers to observe the maximum performance improvement per oracle count, *i.e.*, on how many steps the algorithm requests users to indicate. Then, the developers can decide the number of steps to request based on their performance expectations and the anticipated cognitive load of users and predetermine the optimal set of steps within the decided count.

3.4.2 "What Are You Doing?" We also consider the case where PrISM-Tracker requests users to explain the current step when the algorithm has low confidence in its estimation. Here, we utilize the likelihood of the hypotheses that our Viterbi-based algorithm calculates. When the likelihood is lower than a specific threshold, PrISM-Tracker asks the user on the spot to say what they are doing. This oracle has less information than "tell me when you do X," which specifies not only the step the user is doing but also the exact time information of starting the step. However, this "what are you doing?" oracle request reduces the user's cognitive load; they do not need to memorize on which step they have to indicate.

The threshold can be analogously optimized via a data-driven simulation. Here, the larger threshold results in more frequent requests to the user while helping the algorithm improve the tracking accuracy. Again, PrISM-Tracker allows developers to adjust the number of requests on the basis of the outputted trade-off between the number of oracles and the performance improvement.

3.4.3 "Are You Doing X?". Furthermore, the oracle request of asking "what are you doing?" can be substituted with asking "are you doing X?" Specifically, even when the likelihood of the hypotheses is low, the algorithm has a specific candidate step. PrISM-Tracker can ask a scoped question confirming if the user is actually on the step, which can be instantly responded to by the user. When the user responds negatively, the information is used to eliminate the hypothesis that the algorithm had and switch to other hypotheses. It is also possible to ask a follow-up question "what are you doing?" to update its hypotheses.

Note that we can combine these oracle requests and adapt to the uncertainty in real-time. This flexible human-in-the-loop scheme is enabled by the explicit modeling of the transition probability in the proposed algorithm.

4 AN IN-LAB TASK WITH TRAINED POPULATION

To evaluate PrISM-Tracker, we first collected a dataset in a laboratory. Participants conducted a prepared procedural task while wearing a smartwatch.

4.1 Task: Latte-Making

We chose a procedure that involves the common difficulties in the procedure tracking task we discussed in Section 2. Specifically, we took into account two criteria: 1) the procedure has multiple steps, some of which may be hard to detect by sensors, and 2) the order in which the user completes steps is not rigid. As a result, we chose a "latte-making" task for this data collection. We use a shared espresso machine[1] in an academic building. Users can grind beans, brew coffee, and steam milk with the machine. The proper use of the machine is complicated, and each user of the machine needs to be trained. However, issues often arise due to misuse, such as forgetting to clean parts of the machine after use or doing several steps in the wrong order, which could damage the machine. Thus, this procedure is a suitable application domain for PrISM-Tracker as we can help users and remind them of possibly forgotten steps.

We identified 19 discreet steps of latte-making, and they are presented in Figure 5. Note that users can take several possible orders to perform these steps. We also constrained that users would not perform multiple steps

[1] Breville, the Oracle Touch. https://www.breville.com/us/en/products/espresso/bes990.html?sku=BES990BSS1BUS1

simultaneously, *e.g.*, washing filter (Step 19) while steaming milk (Step 9), which our current architecture cannot handle.

4.2 Apparatus

We used a custom application on Apple Watch Series 6 for data collection. The watch application measured the acceleration and recorded the audio. The acceleration and audio were synchronized and sampled at 50 Hz and 16,000 Hz, respectively. In addition, we developed a web-based annotation tool to record the timing when participants started and ended each step. In a pilot trial, we found that participants sometimes took actions unrelated to any of the steps in the procedure. Here, we did not remove those steps from the data or provide negative labels to them for two reasons: (1) to keep such natural behaviors and test our architecture's performance and (2) to lower the annotation cost as a framework for developers. For consistency in our study, the annotator assigned the same label as the previous step for such moments.

4.3 Data Collection Procedure

We asked university students and faculty members to participate in the data collection through classes and word of mouth. As a result, we had 15 participants in total, including three of the authors. They were all right-handed except one author, and their age distribution was 24.3 ± 3.8 years. Seven of them regularly used the machine. For eight participants who had never used the machine, we conducted a tutorial before they did the procedure, in which we showed them the list of steps and briefly demonstrated each step. Here, we emphasized that some of the steps are order-less, such as cleaning the wand (Step 14 to Step 16) and washing the filter (Step 17 to Step 19), and participants could decide on the ordering. Throughout the data collection, the participants wore the watch on their right wrist. We asked participants to clap their hands at the beginning of the procedure for synchronization. Note that since the machine is placed in a shared building area, there was ambient noise of people chatting aloud nearby during the data collection. Such noises increase the real-world generalizability of our models. After the participant finished the last step, they again clapped their hands and stopped the recordings. It took approximately seven minutes for one session. Participants got a cup of latte they made in the session as compensation for their participation.

4.4 Data Summary

Three participants provided session data multiple times, while other participants conducted the session once. One participant accidentally conducted multiple steps in parallel, so we removed their data. In total, we had 23 sessions with 14 unique participants. The authors' data was always treated as training data in the subsequent evaluations and not used for calculating accuracy numbers. Figure 5 presents all the transitions between steps, generated by all users' transition history in the dataset. As can be seen from this figure, there are many possible transition paths while some steps have a solid order (*e.g.*, Step 3, Step 4, Step 5). Such a graph structure would represent some types of procedures with loosely-constrained orders. In cooking, for example, users can decide the order of cutting vegetables freely to some extent. Note that, in the clinical task that we discuss later in Section 5, the wound care procedure has a strict step-by-step order.

4.5 Evaluation

We first describe the used metrics and compare the performance of PrISM-Tracker and a conventional HAR approach (*i.e.*, without the Viterbi-based correction). Then we discuss the effect of the simulated oracles. Additionally, we show the result of a sensor ablation study. We adopted a leave-one-participant-out approach for training the Random Forest model and the weights of the transition graph.

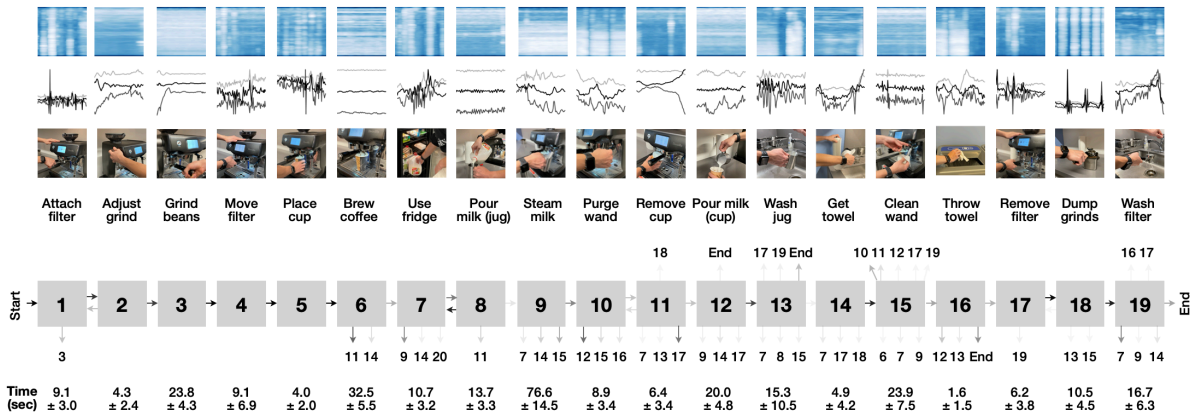

Fig. 5. Step description and transition graph in the latte-making procedure along with corresponding signal visualizations (upper: audio, lower: motion). For example, after Step 8, participants took either Step 7, Step 9, or Step 11. The opacity of the arrows represents the probability of the transition. In other words, the sum of the transitions of arrows from a single step is 1.0.

4.5.1 Metrics. We evaluated the coincidence between the ground truth labels and the output of algorithms at the frame level. For comparison, we also calculated the accuracy without applying the Viterbi correction. However, we found that the frame-level output of the Random Forest model often contains noisy predictions and would not be a fair baseline. Thus, in the same manner as the previous work that introduced filtering techniques [30] to mitigate such noises, we applied a smoothing filter that outputs the most prominent prediction within 15 frames to the frame-level predictions. Hereinafter, we denote this result as raw prediction.

Note that we could consider two different situations in which we use procedure tracking systems: in a post-hoc or real-time manner. A post-hoc manner can be used for the logging purpose and the real-time one for context-aware interactions while users perform procedures. Depending on the two scenarios, we can calculate accuracy in two ways: offline and online accuracy.[2] For offline accuracy, we feed the data of the entire session and examined the corrected prediction for each frame. For online accuracy, we limit the access to the data to those from the beginning of the session to $t + 15$ frames when performing the correction of the frame at t in order to make a fair comparison with the smoothing filter of the raw prediction. This corresponds to a delay of 3 seconds (recall that our window stride is 0.2 sec) and would be reasonable to achieve near real-time performance. It is expected that the offline accuracy become higher than the online accuracy since the Viterbi model can utilize more data to find more plausible hypotheses. However, given scenarios of interactive intelligent systems, we need to evaluate not offline but online performance. Therefore we mainly use the online frame-level accuracy to discuss the results while presenting the offline accuracy as a supplemental metric.

4.5.2 Results of the HAR Model and Viterbi Correction. The results demonstrated that our Viterbi-based algorithm successfully improved performance. Specifically, while raw Random Forest prediction achieves the macro F1-score of 39.7% (accuracy: 57.1%, precision: 45.0%, recall: 38.7%), the Viterbi-based algorithms (offline and online) achieves higher inference performance, macro F1-score = 55.1% (accuracy: 72.4%, precision: 61.1%, recall: 53.1%) and 52.9% (accuracy: 71.0%, precision: 58.8%, recall: 51.0%) for offline and online inferences, respectively. The confusion matrices comparing the result of the raw prediction by the HAR model and after the Viterbi correction (online)

[2]Since the Random Forest outputs predictions for each frame independently, it results in always the same accuracy for the offline and online conditions.

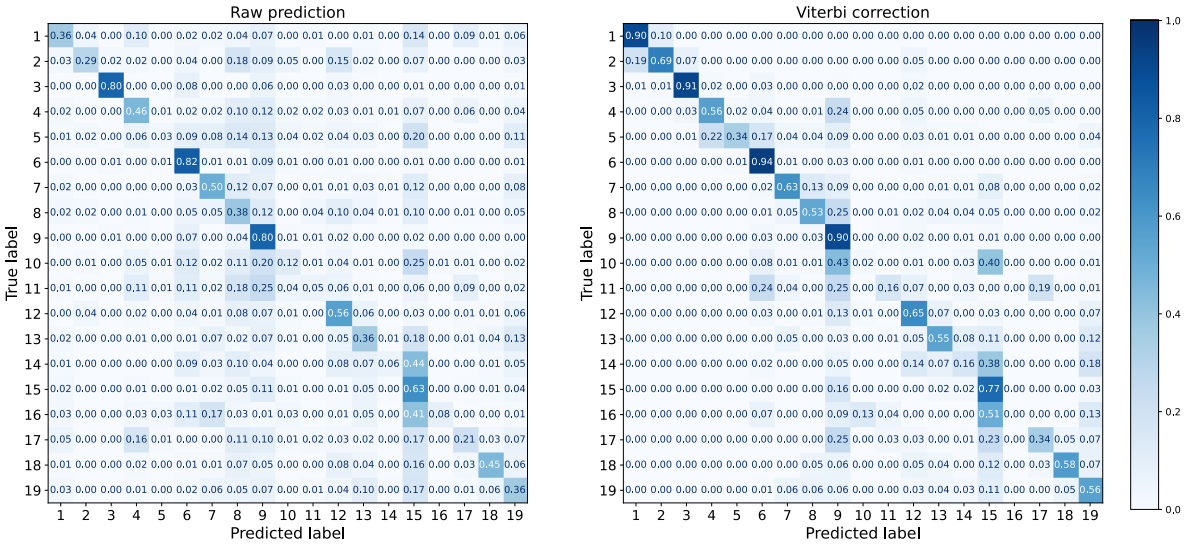

Fig. 6. Confusion matrices of the latte-making task. (left) raw prediction from the HAR model and, (right) online performance after Viterbi correction. The macro F1-score is 39.7% and 52.9%, respectively.

are shown in Figure 6. The Viterbi-based algorithm significantly improved the recall of some steps, such as Steps 3, 4, and 5. This can be attributed to the temporality of procedures. For example, based on Figure 5, it is clear that these three steps happen in order. The result highlights the advantage of our Viterbi correction that can utilize such relationships effectively. Also, given that Step 5 (placing cup) itself is a subtle action and hard to detect by sensors since it does not involve much audio and motion characteristics and does not last for a long duration (4.0 seconds on average), the result reconfirms the difficulty of applying HAR to procedure tracking.

To quantify the effectiveness of the proposed approach compared to conventional approaches (See Section 3.3), we also calculated the result of using a particle filter as proposed by Xia *et al.* [67]. However, we found a large reduction in performance , *i.e.*, the macro F1-score of 15.1% (accuracy: 24.5%, precision: 18.1%, recall: 13.5%) in the offline inference and the macro F1-score of 12.9% (accuracy: 19.5%, precision: 16.1%, recall: 11.3%) in the online inference. Here, the output tended to reach the end state before going through complex transitions, even though we implemented it to explicitly utilize the transition graph (*i.e.*, both transition probability and time information) in its sampling process. The big difference in performance can be attributed to the fact that the particles in the Particle Filter approach concentrate on high-likelihood paths during the forward inference, while our algorithm calculates the likelihood of all possible paths. This result highlights the robustness of the Viterbi-based algorithm in complex and noisy situations.

4.5.3 Results of Using Oracles. Next, we analyze the effectiveness of the interactive oracle request as a post-hoc analysis. Figure 7 shows the relationship between the accuracy (macro F1-score) and the number of oracles in the "Tell me when you do X" and "What are you doing?" oracles. This chart can provide developers using PrISM-Tracker with a sense of how many steps they would need to ask users to achieve the desired accuracy.

In the "Tell me when you do X" oracle, we observed a large improvement as the number of oracle steps increased. For example, if we use a single oracle, the model reaches 58.4% macro F1-score. PrISM-Tracker also outputs which steps to ask users with a priority order. In this case, the oracles used are in the following order according to the usefulness (*i.e.*, gain in macro F1-score): Step 7 → Step 5 → Step 15 → Step 16 → Step 10 → Step

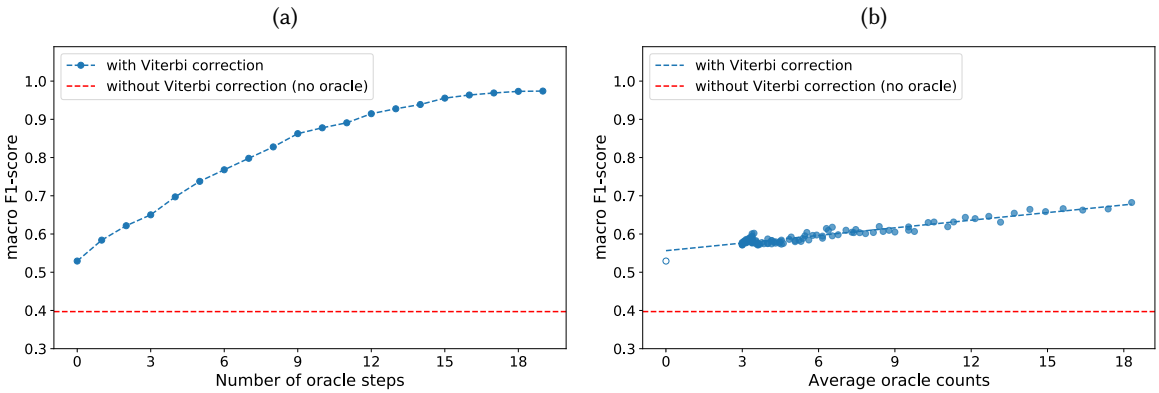

Fig. 7. The relationship between macro F1-score and the number of oracles. (a) "Tell me when you do X" oracle (b) "What are you doing?" oracle. Note that the y-axis does not start from zero.

14 → Step 19 → Step 9 → Step 11 → Step 2 → Step 17 → Step 18 → Step 6 → Step 12 → Step 13 → Step 8 → Step 4 → Step 3 → Step 1. This order indicates that oracles for the steps which have many transition possibilities or those which are hard to detect by the algorithm. Anecdotally, for example, there are two major paths after Step 6 (See also Figure 5: preparing milk (Step 7, Step 8, Step 9) and washing filter (Step 11, Step 17, Step 19)). These two paths are orderless from a semantic view and our participants chose the order they liked. Knowing when Step 7 happens is, thus, helpful for PrISM-Tracker to narrow down candidates of possible sequences, *i.e.*, inferring which path the user has taken first. At the same time, the second prioritized oracle, which is Step 5, is one of the steps hard to recognize even after the Viterbi correction (Figure 6). On the other hand, the steps which do not have many transitions and can be detected by the algorithm without oracles are placed in the latter part of this list (*e.g.*, Step 1, Step3, Step 4).

Using the "Tell me when you do X" oracles multiple times can increase user cognitive load, and we thus explored other ways to request oracles. Thus, we examined the efficacy of the "What are you doing?" oracle. Here, PrISM-Tracker can ask users based on a given threshold for the confidence of the prediction. Thus, we changed the threshold from 2^{-1022} to 2^{-22} exponentially.[3] By changing the threshold, the average number of oracles the system would ask and the average macro F1-score changes. Figure 7 (b) shows their relationship. While the oracles do improve the recall (*e.g.*, approximately 5% with three oracles), the gain is not as much as the "Tell me when you do X" oracle. Figure 7 suggests an example of the trade-off between the gain of different oracle methods and their potential cognitive load posed to users. Our future work will provide further insights by incorporating actual end-user behavior.

Lastly, regarding the "Are you doing X?" oracle, we computed how many of the questions would be answered positively by users. In detail, we calculated the most probable step at each of the frames where PrISM-Tracker used "What are you doing?" oracles and compared them with the corresponding actual step. If we specify to use the "Are you doing X" query three times, 56.4% of them would result in a positive response from the user. This result indicates a possibility of designing interactions such that, instead of using "What are you doing?" oracle every time, PrISM-Tracker could use "Are you doing X?" oracle first, and if users responded negatively, then PrISM-Tracker can ask the users to provide what they are doing. In the future, we plan to evaluate how these strategies affect end-users in different tasks.

[3] 2^{-1022} corresponds to the minimum positive floating-point value of IEEE754.

Fig. 8. F1-score for each step in the ablation study regarding the sensing modality. (top) motion-only (middle) audio-only (bottom) audio+motion.

In sum, the improvements in accuracy would not be made possible without PrISM-Tracker. This is because conventional HAR models do not consider transition information and they are not capable of utilizing oracles to improve the prediction as PrISM-Tracker. Thus, as we discussed in Section 2.2, the procedure tracking with a wearable cannot always be achieved with practical accuracy. Moreover, as we mentioned in Section 3.4, the users of PrISM-Tracker can also combine these interactions to reach desired accuracies while controlling expected end-user cognitive load to use their applications. Here, while we optimized the order of oracles to use in "Tell me when you do X" based on the macro F1-score, we can optimize it using other metrics such as recall. As an example result, if we design PrISM-Tracker to use "Tell me when you do X" once and use "What are you doing?" three times, it reaches 60.8% macro F1-score (accuracy: 75.2%, precision: 66.3%, recall: 58.7%). Future work will test how end-users feel and behave when asked to provide oracles. Importantly, PrISM-Tracker has enabled simulating the trade-off between the number or types of oracles and the expected performance gains.

4.5.4 Ablation Study. We also conducted a sensing modality ablation study. In practice, both sensor channels are not always available, for instance, to avoid using the audio data due to privacy concerns. Therefore, an ablation study provides a reference performance so that developers using PrISM-Tracker for their own procedural tasks can consider potential sensor channels based on performance. As a result, our audio-only and motion-only models achieve 45.4% and 37.3% macro F1-score with the online Viterbi correction (recall our model using both audio and motion information achieved 52.9%). Figure 8 presents F1-score for each step when we used different models. From this chart, we can infer that effective sensor modality is different by steps. For example, while the audio-only model outperforms the motion-only model in overall accuracy, motion is significantly helpful in detecting Step 18 (dumping grinds to trash), which involves a large arm movement. If system developers prioritize detecting Step 18 in order to remind people if they forget to do it, they may just use the motion-only model.

5 AN IN-CLINIC TASK WITH UNTRAINED USERS

Apart from the latte-making task, we also evaluated the effectiveness of PrISM-Tracker in tracking procedures in a more real-world setting. We recruited skin cancer patients and tracked their steps as they learned how to perform a wound care procedure after a Mohs surgery. Skin cancer is the most common malignancy in the United States. Most skin cancer patients are elderly as the incidence of skin cancer increases with age. Mohs surgery is a surgical excision technique that applies oriented microscopic evaluation to achieve high cure rates for skin

cancer. Mohs is most applied to skin cancers on the head and neck, where tissue conservation is most important. Post-operative swelling and bandages can limit vision, hearing, eating, talking, and balance. Mohs patients are overwhelmed by the multiple steps and asymmetric schedules in written instructions. For one week, every three hours, alternating pain medications, icing hourly while awake, and changing dressings daily but not for the first two days are conceptually challenging for most patients. Thus, lowering the chances of infection and maximizing patient independence and self-efficacy is an important application of PrISM-Tracker.

We walked the patient through the procedure step-by-step using a wizard-of-oz interface. The interface, controlled by a nurse, generates audible instructions for the next step of the process. We collected the data in a fully-directed setting because patients were not familiar with the procedure and required detailed information and training while they were in the clinic. Also, the cost of failure is too high for a self-care procedure.

Apart from testing PrISM-Tracker on this new dataset, we also wanted to examine the feasibility of using speech as a medium to provide oracles to PrISM-Tracker in actual interactions. Thus, we asked users to self-narrate as the wizard guided them through the procedure. We based this decision on the fact that speech is one of the most natural ways for users to communicate with systems, and it is now prevalent and socially acceptable [44]. Given that, at this point in the study, we did not have a model built to track steps of this procedure, we could not reactively ask for human input. Thus, the users narrated each step, and we used their speech input as emulated responses to human input requests. We removed the wizard's sounds from the data before running PrISM-Tracker's HAR model.

Although we asked the users to try and narrate at the start of each step, the users did not always remember or comply. This user error is subsumed in the final performance numbers. In this paper, we analyzed the user narration to quantify the possibility, emulating the "What are you doing?" interaction. In the future, the user's narrations could be used to facilitate several other interactions. For example, in some cases, the designers might want the users to narrate every step they perform. That way, speech would become a parallel modality to motion and sound.

5.1 Data Collection

We collected data from patients undergoing Mohs surgery for skin cancer at University Hospitals Health System, Cleveland, Ohio. The patients performed wound care on a fake wound opposite of the real site on the face or neck. Participants wore a smartwatch on their dominant hand. A total of 63 patients participated in our study. However, we found that there was a bug[4]in the data collection app for this study, resulting in 40 of the data being not usable. As a result, we had usable data from 23 participants. Their age distribution was 70.0 ± 10.2. Participants were asked to self-narrate their actions as they performed each step. Note that their narration was recorded on the smartwatch in the same channel as the audio data. In the subsequent analysis, we separated the narration part from the entire audio based on the timestamp, as well as removing the corresponding part from the motion data. This means that the HAR analysis was done without using the part containing narration. It took roughly 5–10 minutes for one participant to complete the procedure.

There were 12 steps in the prepared wound care task. Figure 9 summarizes the step description and shows the transition graph of the procedure along with corresponding signals. In contrast to the latte-making task, the graph has only one possible path. Given the patients were getting trained on the procedure in this study, we kept the order of the steps consistent. As the patients will use PrISM-Tracker at home, this rigidity will go away and the models will need to adapt. Right now, this task adds significantly more environmental noise, the user is untrained, the stakes are higher for the user, but the transition graph is deterministic. Interestingly, we found that the standard deviation of the time participants spend is relatively large for each step. The patients were less trained user population, causing large individual differences in performing the task, which reflects an interesting

[4]The collected audio and motion data were not synchronized

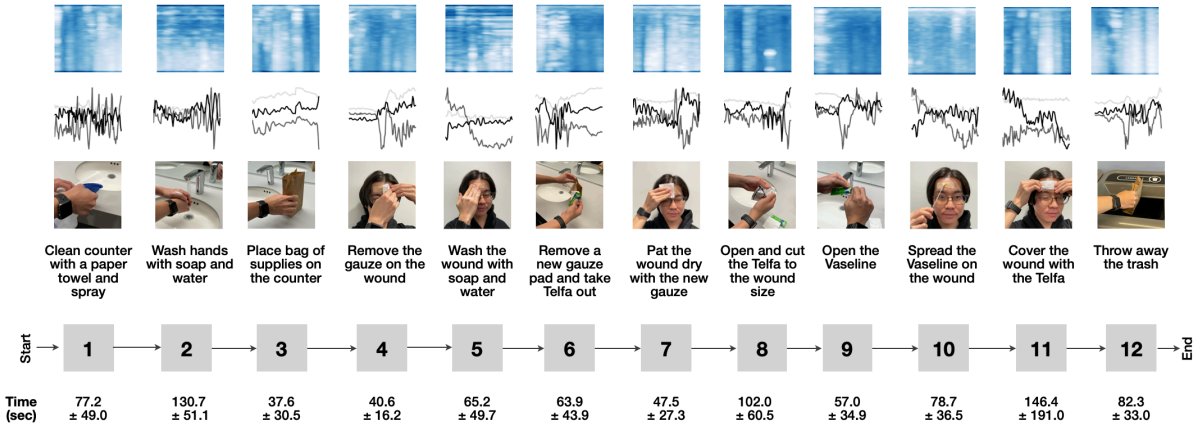

Fig. 9. Step list and the transition graph in the wound care procedure along with corresponding signal visualizations (upper: audio, lower: motion). Steps are directional, and there is one possible path.

real-world user behavior to test PrISM-Tracker. We anticipate this variability in time spent will be even higher in homes.

5.2 Results

With this dataset, too, the results demonstrated that our Viterbi-based algorithm increased the performance. Specifically, while raw Random Forest prediction achieves the macro F1-score of 27.7% (accuracy: 37.3%, precision: 29.6%, recall: 28.7%), the Viterbi-based algorithms (offline and online) achieves higher accuracy, 51.1% (accuracy: 57.0%, precision: 53.2%, recall: 50.4%) and 50.7% (accuracy: 56.7%, precision: 52.8%, recall: 51.2%), respectively. The confusion matrices comparing the result of the raw prediction and the Viterbi correction (online) are shown in Figure 10. The raw accuracy was very low; the sensor signals for most steps are not descriptive here and hard to distinguish. It got improved significantly thanks to the Viterbi correction because the transition graph is one-directional and is a strong cue in this task.

Figure 11 shows the relationship between the macro F1-score and the number of oracles in the "Tell me when you do X" and "What are you doing?" oracles. If PrISM-Tracker uses a single oracle of "Tell me when you do X", it achieves 60.3% macro F1-score (See Figure 11 (a)). The gain is also larger than the latte-making task. This result implies that the "Tell me when you do X" oracle is more helpful when the transition graph does not have many possible paths. The oracles used are in the following order according to the usefulness (*i.e.*, gain of the macro F1-score): Step 8 → Step 3 → Step 6 → Step 9 → Step 5 → Step 10 → Step 4 → Step 2 → Step 7 → Step 1.

Figure 11 (b) shows the relationship between the number of the "What are you doing?" oracle and the improvement of the macro F1-score. The macro F1-score went up by approximately 7% when PrISM-Tracker used two oracles. In addition, in the same way we did in the latte-making task, we computed how many of the top-1 candidate steps were correct in the "What are you doing?" oracles to emulate "Are you doing X?" oracles. As a result, the accuracy macro F1-score was 56.0% when the system asked for the oracles 2.25 times on average. In a similar way to the latte-making task, we can obtain the result by combining the two oracles. As an example result, if we design PrISM-Tracker to use "Tell me when you do X" once and use "What are you doing?" twice, it reaches 66.2% macro F1-score (accuracy: 70.7%, precision: 68.2%, recall: 66.2%).

We also conducted the ablation study regarding the sensor modality. The audio-only and motion-only models achieve 52.3% and 39.7% macro F1-score with the online Viterbi correction (recall our multimodal model using

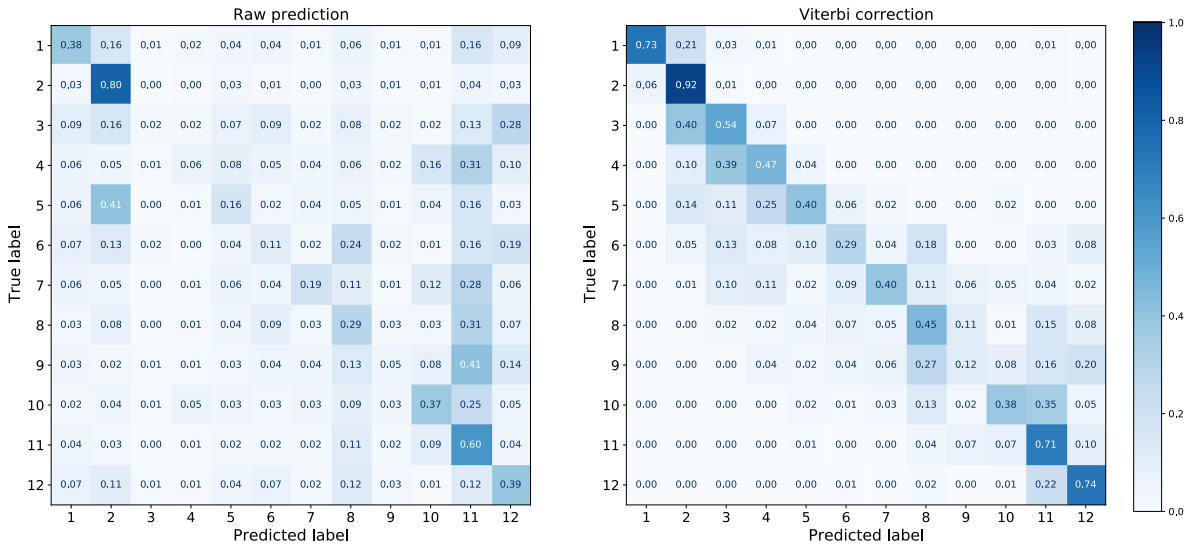

Fig. 10. Confusion matrix of the wound care task. (left) raw prediction (right) online Viterbi correction. The F1-score is 27.7% and 50.7%, respectively.

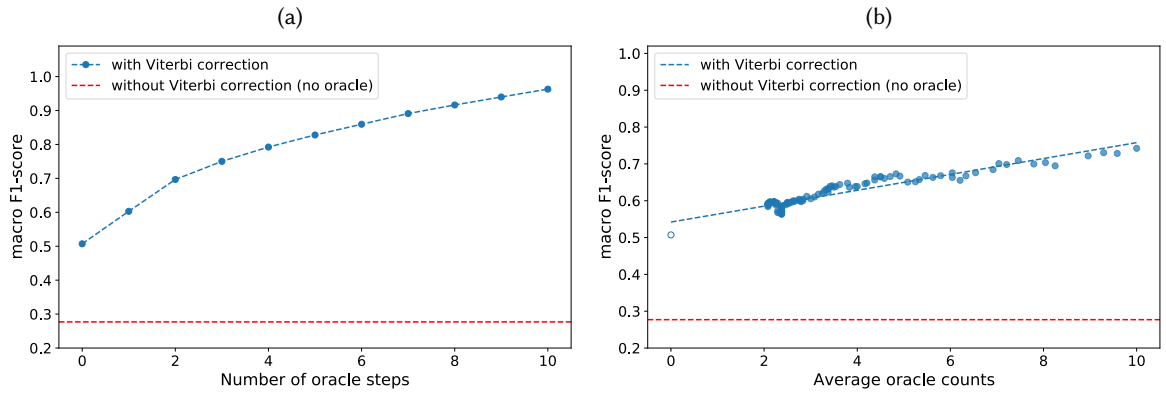

Fig. 11. The relationship between macro F1-score and the number of oracles. (a) "Tell me when you do X" oracle (b) "What are you doing?" oracle. Note that the y-axis does not start from zero.

both audio and motion data achieved 50.7%). Figure 12 presents the F1-score for each of the 12 classes in different models. The audio-only model is slightly better than the multi-modal model. We found that the F1-score for steps where the motion-only model does not work well (*e.g.*, Step 4, Step 10) got degraded in the multi-modal model compared to the audio-only model. Since the feature dimension is large and the training data was not very big, the model might have overfitted to the training data. As we discuss in Section 6, PrISM-Tracker is modular, and developers can fine-tune or switch the feature extractors and the machine learning models freely.

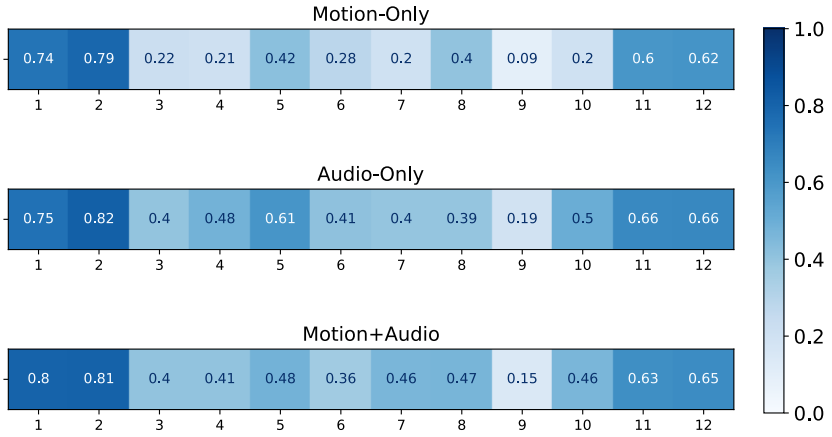

Fig. 12. F1-score for each step in the ablation study regarding the sensing modality. (top) motion-only (middle) audio-only (bottom) audio+motion.

Taking the collected data (*i.e.*, audio file, motion data csv file, and annotation csv file) as inputs, PrISM-Tracker outputs the ML models and the graphs shown in this section. Developers can input their data to generate similar results for their procedures and better design how to include human responses for their tasks.

5.3 Narration Analysis

Next, we analyzed the effectiveness of the participants' narration in providing inputs to the model. Previous studies [65, 70] demonstrated the feasibility of classifying text description into different household tasks in human-robot interaction using NLP techniques. Our analysis can be more challenging since the data source is audio that contains noise (*e.g.*, hand-washing at Step 2), and participants might be unable to describe steps well.

We first applied Google Text to Speech API [20] to the recorded audio, which gave us transcription segments with both start and end times. Then, based on the timestamp, we matched each transcription segment with the steps listed in Figure 9. In other words, we obtained 12 transcriptions for each of the 23 participants.

We trained and evaluated a simple model as a baseline for text classification to check the feasibility. As a baseline model, we used the pretrained BERT [14] model as a feature extractor that outputs a 768-dimensional vector as an embedding for a given sentence. We added a fully-connected linear layer to process this vector to output the final probability of the given sentence belonging to each of the 12 classes. We added a dropout layer before the fully-connected layer with a dropping ratio of 0.4. This entire model (including BERT and the attached fully-connected layer) was trained with the Adam optimizer [37] with an initial learning rate of 0.00002. The batch size was 32, and the model was trained up to 50 epochs. We trained and evaluated the model in a leave-one-participant-out (LOPO) manner. We used 20% of the training data as validation data for each LOPO fold. As a result, we obtained the confusion matrix for the narration classification shown in Figure 13. The classification accuracy was 85.1%.

We then looked at the misclassified predictions and found that they were attributed to the low quality of the recorded speech that caused the error in speech recognition. Since we did not control how the participants narrated, their voices often became small or unclear. It has been empirically shown that such unclear speech is hard to recognize computationally, resulting in high error rates [5]. We believe that users' speech could be recorded more clearly if they know that they interact with the system via speech, by speaking louder or bringing

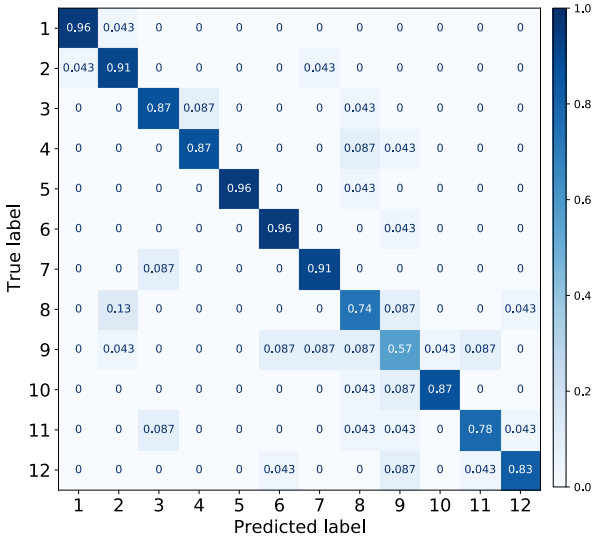

Fig. 13. Classification results of the self-narration describing each step. The narration was first processed with an existing speech recognition API, and the transcriptions were used for classification. The overall accuracy was 85.1%.

the smartwatch closer to their mouth [50]. In addition, we can consider addressing this misclassification by detecting ambiguous transcriptions and asking the user to repeat the narration clearly.

From this preliminary result, we conclude that speech input is feasible for users to provide oracle inputs to PrISM-Tracker in procedure tracking. As we mention later in Section 7.2, we will implement the speech recognition and text classification model in the watch for a self-contained real-time system in future work. We want to emphasize that the introduction of this narration-based interaction does not hinder the application of PrISM-Tracker to real-time tracking scenarios. This is because recent streaming speech recognition algorithms can run with an average latency of less than 1 second [55]. In addition, the inference of the text classification model on one sentence can run on a regular laptop (MacBook Pro M1) with a 50 ms processing time.

6 OPEN SOURCE FRAMEWORK AND DATASET

To enable future research to build on our work, and contribute to this domain, we have made the code of the iOS application used for the data collection and the Python implementation of the framework freely available at https://github.com/cmusmashlab/prism-tracker. Given the various needs mentioned in Section 2.1, we envision developers will use this framework to design their own interactive systems for procedural tasks.

Developers can use our framework through the following steps: First, we have to define the step list for our own tasks. Then, we install the data collection app on a smartwatch and collect the data for several sessions. Here, we are supposed to emulate the situation where end-users would use the system. As we mentioned in Section 4.3, we need to clap our hands at the beginning and end of the procedure. We also need to take a video for the annotation purpose.

Then, we need to annotate the video in a csv file to mark when each step starts and ends while aligning the first clapping frame as 0 seconds. PrISM-Tracker takes the annotation files as inputs to first generate a transition graph as a Python pickle file. Here, we can also manually check and customize the graph in Python, like explicitly making probabilities of some transitions zero to prohibit such transitions.

The data collection app produces audio data in a .wav format, motion data in a .csv format. PrISM-Tracker takes these two files along with the pickle file of the transition graph to output the results we showed in Section 4.5 and Section 5.2. In addition, PrISM-Tracker is modular, and we can switch to different HAR models as long as the data format is the same. For example, if we develop a system for cooking tasks, it is possible to use a feature extractor of the motion data that is pretrained with cooking HAR datasets [56].

In addition to the code, we publicize the dataset of the latte-making task for replicability in the same repository. We plan to collect more procedures in various tasks to make a larger dataset and report the accuracy of PrISM-Tracker as a baseline.

7 DISCUSSION

So far, we have shown how PrISM-Tracker allows us to build procedure tracking systems with improved accuracy by leveraging the transition graph and the involvement of users for handling errors and uncertainty inherent in HAR for real-world applications. In this section, we discuss the limitation of our current work and future work.

7.1 Limitations

In our evaluations, we simulated oracles to verify the effectiveness of PrISM-Tracker in incorporating human inputs. It assumes that users would respond ideally to the system. This assumption is not likely to hold in the real world. For example, users might forget to provide the "Tell me when you do X" oracles if they need to provide them many times. Additionally, if we use speech as a method for a user to provide oracles, misclassification might occur in ASR and text classification. When PrISM-Tracker receives wrong information, its tracking performance would be lower than shown in this work. From the perspective of human-AI interaction [2], however, we conjecture that end-users might not lose much trust in PrISM-Tracker since they could understand why the system does not track steps properly (*e.g.*, users' speech is not transcribed accurately, or users forget to provide prompts.) While we will deploy real-time interactive systems to study actual user behavior, our work is the first step toward such interactive intelligent systems for procedure tracking by proposing an algorithm to use human input efficiently.

As we mentioned in Section 4.3, our current architecture cannot capture moments when users conduct multiple steps simultaneously. In addition, our current architecture does not assume that users add new steps to the procedure on the spot while performing the procedure (*e.g.*, adding sugar to the coffee in the latte-making task). We can also extend our human-in-the-loop framework to adjust to such user behaviors. For example, we can extend the current architecture to apply out-of-distribution detection for HAR [38] and detect moments where the HAR model does not expect. Then the model asks users using the "What are you doing?" request so that the model can collect oracles for the data, which in turn helps retrain the model.

7.2 Future Work

7.2.1 Real-time Interactive System. In this paper, we focused on formulating and quantifying the performance of the general architecture for detecting users' activities throughout procedures by introducing the human-in-the-loop scheme. As a next step, we plan to deploy the framework into real-time interactive systems. Notably, our current implementation suggests that the processing of PrISM-Tracker can run fast enough for real-time applications. In detail, our multimodal HAR and frame-level classification latency is approximately 40 ms on MacBook Pro with an M1 chip with 16 GB memory. Our Viterbi-based algorithm also runs fast enough, taking 0.20 ms for processing the predicted probabilities of one window data. Given the window stride (*i.e.*, an interval of each prediction) is 200 ms in our architecture, our model can run fast enough for real-time tracking using a laptop. Moreover, regarding the human prompts, we also showed that the speech input could be reliably used and run fast enough (50 ms delay for narration classification) in Section 5.3 using the same laptop environment.

Currently, we are trying to implement each module in a self-contained smartwatch app. This effort requires engineering optimizations such as model compression or floating point quantization [28].

7.2.2 User Behavioral Study. Next, after implementing the real-time system, we plan to use it to explore user behavior concerning the interactive oracle request feature. As we mentioned in Section 2.3, users will feel a burden if they need to provide many prompts to the intelligent agent. As a baseline design, we proposed three ways to provide the prompts in this work: "Tell me when you do X", "What are you doing?", and "Are you doing X?". Evaluating how users would feel toward these requests with different frequency settings in various procedures is desirable. Such a study would shed light on how the perceived merit of providing prompts in human-AI systems would affect users' behaviors to help intelligent agents. The advantage of PrISM-Tracker is that it enables such explorations by outputting the trade-offs between the accuracy gain and the combination of oracles.

Moreover, a longitudinal study is desirable to be conducted to look into how PrISM-Tracker fits users' lives. On the one hand, users' speech behavior will likely change over time if they repeatedly use the system for a procedure they do. For example, users may provide more inputs to the system at the beginning but do less often as they get used to the procedure. Such a change would affect PrISM-Tracker's performance as we discussed in Section 7.1. On the other hand, the model can also learn from user prompts as a new label and update the model parameters. Such a feature would enable the system to adjust the timing of requesting oracles.

7.2.3 Wound Care in Patients' Homes. The current in-clinic wound care study serves as initial patient training. The actual use of PrISM-Tracker happens in patients' homes. In future work, we will take our phone or watch-based implementations of PrISM-Tracker to patients' homes, and they will use the system for their recovery week. The app will keep track of their steps and procedures. Being in a real-world situation where the user is untrained will throw many usability and logistical challenges. The effort would require several iterations of deployment but wound care represents an important use for PrISM-Tracker. The research team wants to ultimately build an intelligent conversational agent that helps the user keep track of their progress, help them correct their mistakes, and send final reports to the medical staff.

7.2.4 Prioritize Some Steps. Currently, PrISM-Tracker assumes that all steps of a procedure are equally important. This assumption is invalid as many procedures have some critical and some optional steps. For example, the hand washing step in the wound care task is much more important than opening the vaseline bottle. The bottle's opening could also be implied if the system detects the user applied the vaseline. Thus, in the future, PrISM-Tracker will allow developers to prioritize steps for inference and adjust their weights accordingly in the ML models.

7.2.5 Use of Prior Knowledge about Transition. In the current implementation, the transition graph is obtained in a fully data-driven manner. However, it is possible to incorporate human knowledge to build the graph, for example, by explicitly enumerating possible transitions. In the future system of PrISM-Tracker, we will enable such interactive control by preparing a dedicated interface for a developer to input such knowledge. This would be particularly helpful when the procedure includes repetition of steps (*e.g.*, opening a fridge in a cooking task). If we know which steps are to be performed repeatedly, we can enable PrISM-Tracker to distinguish them by assigning separate states in the transition graph, and thus to consider the order of the repeated steps.

7.2.6 Algorithm Refinement for Offline Use. Although our evaluations demonstrated the effectiveness of PrISM-Tracker using oracles, the accuracy without oracles has room for improvement by employing the recent advancement in deep learning. Specifically, as mentioned in Section 3.2, we used a pretrained CNN-based neural network as feature extractors in combination with a random forest classifier to perform frame-level sensing. We believe that we can train an end-to-end model under the condition that a larger amount of data is available. For example,

by employing neural networks that can consider time-series dependencies (*e.g.*, Transformer [63]) or probabilistic structure (*e.g.*, deep graphical models [26]), we expect the accuracy will be improved. Still, we would like to emphasize that our Viterbi correction can be combined with such models to refine their output by explicitly incorporating the transition graphs. The human-in-the-loop approach via the oracles enabled by PrISM-Tracker would also be beneficial in such cases.

Furthermore, while we focused on online scenarios in our evaluation, our algorithm can be further improved in the offline estimation, such as for logging time and making reports. Specifically, upon receiving an oracle of one frame (*e.g.*, "What are you doing?"), it would be possible to update the estimations until the moment by finding feasible transition paths backwardly, while our Viterbi correction uses the information forwardly and does not update the past estimations. A naïve implementation of such an algorithm is to conduct a full search on the possible sequences given the start step and the step the user provided as the oracle, and find the one having the highest likelihood according to the past predicted probabilities of the ML models. However, this algorithm would take a long time because the search space gets exponentially larger as the complexity of the transition graph increases. It could be possible to incorporate the modified version of the Viterbi algorithm [10] that can introduce constraints to its results and re-run the estimation from the beginning whenever the system retrieves the oracle.

8 CONCLUSION

We have presented *PrISM-Tracker*, a framework for procedure tracking using a smartwatch. Tracking users' activity during procedures is challenging, where conventional human activity recognition (HAR) predictions can be noisy. To improve the tracking accuracy, we proposed an algorithm that leverages a transition graph (*i.e.*, transition probability and time distribution for each step). Moreover, we enabled our architecture to incorporate a human-in-the-loop error-correction approach to handle inevitable errors of the HAR model. We demonstrated the efficacy of PrISM-Tracker in two tasks, the latte-making task in a laboratory and the wound care task in a clinic. Additionally, we showed the feasibility of using speech for users to communicate with the system to provide oracles. While future work remains, our proposed framework is a first step to achieving practical accuracy of procedure tracking and helping developers build a wealth of new and interesting end-user applications.

ACKNOWLEDGMENTS

This work was supported in part by Google, Japan Science and Technology ACT-X (JPMJAX200R), and Japan Society for the Promotion of Science KAKENHI (JP21J20353).

REFERENCES

[1] Saleema Amershi, Maya Cakmak, W. Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. *AI Magazine* 35, 4 (2014), 105–120. https://doi.org/10.1609/aimag.v35i4.2513

[2] Saleema Amershi, Daniel S. Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi T. Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for human-AI interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 3. https://doi.org/10.1145/3290605.3300233

[3] Apple. 2022. Handwashing on Apple Watch. https://support.apple.com/guide/watch/set-up-handwashing-apdc9b9f04a8/watchos

[4] Riku Arakawa, Sosuke Kobayashi, Yuya Unno, Yuta Tsuboi, and Shin-ichi Maeda. 2018. DQN-TAMER: Human-in-the-loop reinforcement learning with intractable feedback. In *Proceedings of 2nd Workshop on Human-Robot Teaming Beyond Human Operational Speeds and Robot Teammates Operating in Dynamic, Unstructured Environments*. 2 pages.

[5] Riku Arakawa, Hiromu Yakura, and Masataka Goto. 2022. BeParrot: Efficient interface for transcribing unclear speech via respeaking. In *Proceedings of the 27th International Conference on Intelligent User Interfaces*. ACM, New York, NY, 832–840. https://doi.org/10.1145/3490099.3511164

[6] Vincent Becker, Linus Fessler, and Gábor Sörös. 2019. GestEar: Combining audio and motion sensing for gesture recognition on smartwatches. In *Proceedings of the 23rd International Symposium on Wearable Computers*. ACM, New York, NY, 10–19. https://doi.org/10.1145/3341163.3347735

[7] Edgar A. Bernal, Xitong Yang, Qun Li, Jayant Kumar, Sriganesh Madhvanath, Palghat Ramesh, and Raja Bala. 2018. Deep temporal multimodal fusion for medical procedure monitoring using wearable sensors. *IEEE Transactions on Multimedia* 20, 1 (2018), 107–118. https://doi.org/10.1109/TMM.2017.2726187

[8] Sarnab Bhattacharya, Rebecca Adaimi, and Edison Thomaz. 2022. Leveraging sound and wrist motion to detect activities of daily living with commodity smartwatches. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 2 (2022), 42:1–42:28. https://doi.org/10.1145/3534582

[9] Samuel Budd, Emma C. Robinson, and Bernhard Kainz. 2021. A survey on active learning and human-in-the-loop deep learning for medical image analysis. *Medical Image Analysis* 71 (2021), 102062. https://doi.org/10.1016/j.media.2021.102062

[10] Yin-Wen Chang, Alexander M. Rush, John DeNero, and Michael Collins. 2014. A constrained Viterbi relaxation for bidirectional word alignment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Association for Computer Linguistics, Stroudsburg, PA, 1481–1490. https://doi.org/10.3115/v1/p14-1139

[11] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu. 2021. Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *Comput. Surveys* 54, 4 (2021), 77:1–77:40. https://doi.org/10.1145/3447744

[12] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Proceedings of the 2017 Annual Conference on Neural Information Processing Systems*. Neural Information Processing Systems Foundation, La Jolla, CA, 4299–4307.

[13] Christian Arzate Cruz and Takeo Igarashi. 2021. A survey on interactive reinforcement learning: Design principles and open challenges. *arXiv* 2105.12949 (2021), 15 pages. https://doi.org/10.48550/arXiv.2105.12949

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Stroudsburg, PA, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[15] John J. Dudley and Per Ola Kristensson. 2018. A Review of User Interface Design for Interactive Machine Learning. *ACM Transactions on Interactive Intelligent Systems* 8, 2 (2018), 8:1–8:37. https://doi.org/10.1145/3185517

[16] Amelia Fiske, Alena Buyx, and Barbara Prainsack. 2020. The double-edged sword of digital self-care: Physician perspectives from Northern Germany. *Social Science & Medicine* 260 (2020), 113174. https://doi.org/10.1016/j.socscimed.2020.113174

[17] G David Forney. 1973. The Viterbi algorithm. *Proc. IEEE* 61, 3 (1973), 268–278.

[18] Miriam Gil, Vicente Pelechano, Joan Fons, and Manoli Albert. 2016. Designing the Human in the Loop of self-adaptive systems. In *Proceedings of the 10th International Conference on Ubiquitous Computing and Ambient Intelligence*, Vol. 10069. Springer, Cham, Switzerland, 437–449. https://doi.org/10.1007/978-3-319-48746-5_45

[19] Steven Goodman, Ping Liu, Dhruv Jain, Emma J. McDonnell, Jon E. Froehlich, and Leah Findlater. 2021. Toward user-driven sound recognizer personalization with people who are d/deaf or hard of hearing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 2 (2021), 63:1–63:23. https://doi.org/10.1145/3463501

[20] Google. 2022. Text-to-Speech. https://cloud.google.com/text-to-speech

[21] Jun Hatori, Yuta Kikuchi, Sosuke Kobayashi, Kuniyuki Takahashi, Yuta Tsuboi, Yuya Unno, Wilson Ko, and Jethro Tan. 2018. Interactively picking real-world objects with unconstrained spoken language instructions. In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation*. IEEE, New York, NY, 3774–3781. https://doi.org/10.1109/ICRA.2018.8460699

[22] Andreas Holzinger. 2016. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics* 3, 2 (2016), 119–131. https://doi.org/10.1007/s40708-016-0042-6

[23] Gaoping Huang, Xun Qian, Tianyi Wang, Fagun Patel, Maitreya Sreeram, Yuanzhi Cao, Karthik Ramani, and Alexander J. Quinn. 2021. AdapTutAR: An adaptive tutoring system for machine tasks in augmented reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 417:1–417:15. https://doi.org/10.1145/3411764.3445283

[24] Zawar Hussain, Quan Z. Sheng, and Wei Emma Zhang. 2020. A review and categorization of techniques on device-free human activity recognition. *Journal of Network and Computer Applications* 167 (2020), 102738. https://doi.org/10.1016/j.jnca.2020.102738

[25] Kyuwoong Hwang and Soo-Young Lee. 2012. Environmental audio scene and activity recognition through mobile-based crowdsourcing. *IEEE Transactions on Consumer Electronics* 58, 2 (2012), 700–705. https://doi.org/10.1109/TCE.2012.6227479

[26] Matthew J. Johnson, David Duvenaud, Alexander B. Wiltschko, Ryan P. Adams, and Sandeep R. Datta. 2016. Composing graphical models with neural networks for structured representations and fast inference. In *Proceedings of the 2016 Annual Conference on Neural Information Processing Systems*. 2946–2954.

[27] Rushil Khurana, Karan Ahuja, Zac Yu, Jennifer Mankoff, Chris Harrison, and Mayank Goel. 2018. GymCam: Detecting, recognizing and tracking simultaneous exercises in unconstrained scenes. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 4 (2018), 185:1–185:17. https://doi.org/10.1145/3287063

[28] Hyeji Kim, Muhammad Umar Karim Khan, and Chong-Min Kyung. 2019. Efficient neural network compression. In *Proceedings of the 2019 CVF/IEEE Conference on Computer Vision and Pattern Recognition*. CVF/IEEE, New York, NY, 12569–12577. https://doi.org/10.1109/CVPR.2019.01285

[29] Ryosuke Kojima, Osamu Sugiyama, and Kazuhiro Nakadai. 2016. Multimodal scene understanding framework and its application to cooking recognition. *Applied Artificial Intelligence* 30, 3 (2016), 181–200. https://doi.org/10.1080/08839514.2016.1156461

[30] Yusaku Kormatsu, Daisuke Saito, and Nobuaki Minematsu. 2019. Cooking state recognition based on acoustic event detection. In *Proceedings of the 11th Workshop on Multimedia for Cooking and Eating Activities*. ACM, New York, NY, 41–44. https://doi.org/10.1145/3326458.3326932

[31] HyeokHyen Kwon, Catherine Tong, Harish Haresamudram, Yan Gao, Gregory D. Abowd, Nicholas D. Lane, and Thomas Plötz. 2020. IMUTube: Automatic extraction of virtual on-body accelerometry from video for human activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 3 (2020), 87:1–87:29. https://doi.org/10.1145/3411841

[32] Gierad Laput, Karan Ahuja, Mayank Goel, and Chris Harrison. 2018. Ubicoustics: Plug-and-play acoustic activity recognition. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, 213–224. https://doi.org/10.1145/3242587.3242609

[33] Gierad Laput, Robert Xiao, and Chris Harrison. 2016. ViBand: High-fidelity bio-acoustic sensing using commodity smartwatch accelerometers. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, New York, NY, 321–333. https://doi.org/10.1145/2984511.2984582

[34] Oscar D. Lara and Miguel A. Labrador. 2013. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys and Tutorials* 15, 3 (2013), 1192–1209. https://doi.org/10.1109/SURV.2012.110112.00192

[35] Lowell S Levin and Ellen L Idler. 1983. Self-care in health. *Annual review of public health* 4, 1 (1983), 181–201.

[36] Wesllen Sousa Lima, Eduardo Souto, Khalil El-Khatib, Roozbeh Jalali, and João Gama. 2019. Human activity recognition using inertial sensors in a smartphone: An overview. *Sensors* 19, 14 (2019), 3213. https://doi.org/10.3390/s19143213

[37] Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations*. OpenReview.net, 18 pages.

[38] Devraj Mandal, Sanath Narayan, Sai Kumar Dwivedi, Vikram Gupta, Shuaib Ahmed, Fahad Shahbaz Khan, and Ling Shao. 2019. Out-of-distribution detection for generalized zero-shot action recognition. In *Proceedings of the 2019 CVF/IEEE Conference on Computer Vision and Pattern Recognition*. CVF/IEEE, New York, NY, 9985–9993. https://doi.org/10.1109/CVPR.2019.01022

[39] Sara Ashry Mohammed, Reda Elbasiony, and Walid Gomaa. 2018. An LSTM-based descriptor for human activities recognition using IMU sensors. In *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics*. SciTePress, Setúbal, Portugal, 504–511. https://doi.org/10.5220/0006902405040511

[40] Vimal Mollyn, Karan Ahuja, Dhruv Verma, Chris Harrison, and Mayank Goel. 2022. SAMoSA: Sensing Activities with Motion and Subsampled Audio. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 3 (2022), 132:1–132:19. https://doi.org/10.1145/3550284

[41] Robert Munro Monarch. 2021. *Human-in-the-loop machine learning: Active learning and annotation for human-centered AI.* Simon and Schuster, New York, NY.

[42] Dan Morris, T. Scott Saponas, Andrew Guillory, and Ilya Kelner. 2014. RecoFit: Using a wearable sensor to find, recognize, and count repetitive exercises. In *Proceedings of the 2014 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 3225–3234. https://doi.org/10.1145/2556288.2557116

[43] Tamanna Motahar, Isha Ghosh, and Jason Wiese. 2022. Identifying factors that inhibit self-care behavior among individuals with severe spinal cord injury. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 3:1–3:16. https://doi.org/10.1145/3491102.3517658

[44] Christine Murad, Cosmin Munteanu, Leigh Clark, and Benjamin R. Cowan. 2018. Design guidelines for hands-free speech interaction. In *Adjunct Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, New York, NY, 269–276. https://doi.org/10.1145/3236112.3236149

[45] Yasushi Nakauchi, Takuo Suzuki, Akira Tokumasu, and Sho Murakami. 2009. Cooking procedure recognition and inference in sensor embedded kitchen. In *Proceedings of the 18th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, New York, NY, 593–600. https://doi.org/10.1109/ROMAN.2009.5326050

[46] Henry Friday Nweke, Ying Wah Teh, Mohammed Ali Al-garadi, and Uzoma Rita Alo. 2018. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications* 105 (2018), 233–261. https://doi.org/10.1016/j.eswa.2018.03.056

[47] Jennifer Ockerman and Amy Pritchett. 2000. A review and reappraisal of task guidance: Aiding workers in procedure following. *International Journal of Cognitive Ergonomics* 4, 3 (2000), 191–212. https://doi.org/10.1207/s15327566ijce0403_2

[48] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830. https://doi.org/10.5555/1953048.2078195

[49] Yevhenii Prokopalo, Meysam Shamsi, Loïc Barrault, Sylvain Meignier, and Anthony Larcher. 2021. Active correction for speaker diarization with human in the loop. In *Proceedings of the 5th IberSPEECH Conference*. ISCA, Baixas, France, 5 pages.

[50] Yue Qin, Chun Yu, Zhaoheng Li, Mingyuan Zhong, Yukang Yan, and Yuanchun Shi. 2021. ProxiMic: Convenient voice activation via close-to-mic speech detected by a single microphone. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 8:1–8:12. https://doi.org/10.1145/3411764.3445687

[51] Gonzalo A. Ramos, Christopher Meek, Patrice Y. Simard, Jina Suh, and Soroush Ghorashi. 2020. Interactive machine teaching: A human-centered approach to building machine-learned models. *Human–Computer Interaction* 35, 5-6 (2020), 413–451. https://doi.org/10.1080/07370024.2020.1734931

[52] Ayaka Sato, Keita Watanabe, and Jun Rekimoto. 2014. MimiCook: A cooking assistant system with situated guidance. In *Proceedings of the 8th International Conference on Tangible, Embedded, and Embodied Interaction*. ACM, New York, NY, 121–124. https://doi.org/10.1145/2540930.2540952

[53] Javier Serván, Fernando Mas, José Luis Menéndez, and José Ríos. 2012. Assembly work instruction deployment using augmented reality. *Key Engineering Materials* 502 (2012), 25–30. https://doi.org/10.4028/www.scientific.net/KEM.502.25

[54] Carrie L Shandra and Nihil Sonalkar. 2016. Health self-care in the United States. *Public Health* 138 (2016), 26–32. https://doi.org/10.1016/j.puhe.2016.02.030

[55] Yangyang Shi, Yongqiang Wang, Chunyang Wu, Ching-Feng Yeh, Julian Chan, Frank Zhang, Duc Le, and Mike Seltzer. 2021. Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition. In *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, New York, NY, 6783–6787. https://doi.org/10.1109/ICASSP39728.2021.9414560

[56] Ekaterina H. Spriggs, Fernando De la Torre, and Martial Hebert. 2009. Temporal segmentation and activity classification from first-person sensing. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Washington, DC, 17–24. https://doi.org/10.1109/CVPRW.2009.5204354

[57] Johannes Andreas Stork, Luciano Spinello, Jens Silva, and Kai Oliver Arras. 2012. Audio-based human activity recognition using non-Markovian ensemble voting. In *Proceedings of the 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, New York, NY, 509–514. https://doi.org/10.1109/ROMAN.2012.6343802

[58] Jie Su, Zhenyu Wen, Tao Lin, and Yu Guan. 2022. Learning disentangled behaviour patterns for wearable-based human activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 1 (2022), 28:1–28:19. https://doi.org/10.1145/3517252

[59] Fei Tan, David Caicedo, Ashish Pandharipande, and Marco Zuniga. 2018. Sensor-driven, human-in-the-loop lighting control. *Lighting Research & Technology* 50, 5 (2018), 660–680.

[60] Vaibhav V. Unhelkar, Shen Li, and Julie A. Shah. 2020. Decision-making for bidirectional communication in sequential human-robot collaborative tasks. In *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, New York, NY, 329–341. https://doi.org/10.1145/3319502.3374779

[61] Daisuke Uriu, Mizuki Namai, Satoru Tokuhisa, Ryo Kashiwagi, Masahiko Inami, and Naohito Okude. 2012. Panavi: Recipe medium with a sensors-embedded pan for domestic users to master professional culinary arts. In *Proceedings of the 2012 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 129–138. https://doi.org/10.1145/2207676.2207695

[62] Monica M Van Acker and MArk A Kuriata. 2014. Video education provides effective wound care instruction pre-or post-Mohs micrographic surgery. *The Journal of Clinical and Aesthetic Dermatology* 7, 4 (2014), 43.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. [n.d.].

[64] Michalis Vrigkas, Christophoros Nikou, and Ioannis A. Kakadiaris. 2015. A review of human activity recognition methods. *Frontiers Robotics AI* 2 (2015), 28. https://doi.org/10.3389/frobt.2015.00028

[65] Naoki Wake, Riku Arakawa, Iori Yanokura, Takuya Kiyokawa, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. 2021. A learning-from-observation framework: One-shot robot teaching for grasp-manipulation-release household operations. In *Proceedings of the 2021 IEEE/SICE International Symposium on System Integration*. IEEE, New York, NY, 461–466. https://doi.org/10.1109/IEEECONF49454.2021.9382750

[66] Gary M. Weiss, Jessica L. Timko, Catherine M. Gallagher, Kenichi Yoneda, and Andrew J. Schreiber. 2016. Smartwatch-based activity recognition: A machine learning approach. In *Proceedings of the 2016 IEEE-EMBS International Conference on Biomedical and Health Informatics*. IEEE, New York, NY, 426–429. https://doi.org/10.1109/BHI.2016.7455925

[67] Qingxin Xia, Atsushi Wada, Joseph Korpela, Takuya Maekawa, and Yasuo Namioka. 2019. Unsupervised factory activity recognition with wearable sensors using process instruction information. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 2 (2019), 60:1–60:23. https://doi.org/10.1145/3328931

[68] Doris Xin, Litian Ma, Jialin Liu, Stephen Macke, Shuchen Song, and Aditya G. Parameswaran. 2018. Accelerating human-in-the-loop machine learning: Challenges and opportunities. In *Proceedings of the 2nd Workshop on Data Management for End-To-End Machine Learning*. ACM, New York, NY, 9:1–9:4. https://doi.org/10.1145/3209889.3209897

[69] Masahiro Yamaguchi, Shohei Mori, Peter Mohr, Markus Tatzgern, Ana Stanescu, Hideo Saito, and Denis Kalkofen. 2020. Video-annotated augmented reality assembly tutorials. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, 1010–1022. https://doi.org/10.1145/3379337.3415819

[70] Iori Yanaokura, Naoki Wake, Kazuhiro Sasabuchi, Riku Arakawa, Kei Okada, Jun Takamatsu, Masayuki Inaba, and Katsushi Ikeuchi. 2022. A multimodal learning-from-observation towards all-at-once robot teaching using task cohesion. In *Proceedings of the 2022 IEEE/SICE International Symposium on System Integration*. IEEE, New York, NY, 367–374. https://doi.org/10.1109/SII52469.2022.9708836

[71] Ruohan Zhang, Faraz Torabi, Lin Guan, Dana H. Ballard, and Peter Stone. 2019. Leveraging human guidance for deep reinforcement Learning Tasks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. ijcai.org, Somerset, NJ, 6339–6346. https://doi.org/10.24963/ijcai.2019/884

[72] Wei Zhuang, Yi Chen, Jian Su, Baowei Wang, and Chunming Gao. 2019. Design of human activity recognition algorithms based on a single wearable IMU sensor. *International Journal of Sensor Networks* 30, 3 (2019), 193–206. https://doi.org/10.1504/IJSNET.2019.100218