A Model-Based Human Activity Recognition for Human–Robot Collaboration

Sang Uk Lee Andreas Hofmann Brian Williams

***Abstract*— Human activity recognition is a crucial ingredient in safe and efficient human–robot collaboration. In this paper, we present a new model-based human activity recognition system called logical activity recognition system (LCARS). LCARS requires much less training data compared to learning-based works. Compared to other model-based works, LCARS requires minimal domain-specific modeling effort from users. The minimal modeling is for two reasons: i) we provide a systematic and intuitive way to encode domain knowledge for LCARS and ii) LCARS automatically constructs a probabilistic estimation model from the domain knowledge. Requiring minimal training data and modeling effort allows LCARS to be easily applicable to various scenarios. We verify this through simulations and experiments.**

I. Introduction

Human activity recognition (HAR) is crucial for successful human–robot collaboration. Consider a scenario in which a human and a robot are collaborating to make a wooden chair. A hammer and a drill are in drawer *A*, and a box of nails is in drawer *B*. If the robot recognizes that the human is picking up the hammer from drawer *A*, it can avoid a collision and be helpful by getting the nails from drawer *B*.

Today, learning-based (or model-free) approaches, such as convolutional neural networks, are popular in HAR. However, learning-based approaches have a major limitation. They require a large amount of training data, but data aquisition is difficult in HAR. For instance, the Opportunity Activity Recognition dataset used for training CNNs in [1], [2] was comprised of 25 hours of data collected from 12 subjects, each of whom personally performed activities repetitively [3]. This is a tedious and exhausting process. Moreover, datasets must be collected all over again if there are changes to the environment or activities.

Thus, we propose a new model-based HAR system called logical activity recognition system (LCARS). As we use a model-based approach, we rely more on domain knowledge, rather than training data. One limitation of previous model-based works is that they require much domain-specific modeling effort from users [4]–[6]. Thus, their applicability to various scenarios is an issue. On the other hand, LCARS requires minimal modeling effort for the following two reasons. First, we provide a systematic and intuitive way to encode domain knowledge for LCARS. Users can encode the domain knowledge using region connection calculus (RCC) and planning domain definition language (PDDL), both of which have been successful modeling tools in classical (or symbolic) artificial intelligence (AI) research. Second, LCARS automatically constructs a probabilistic estimation model from the domain knowledge. The probabilistic estimation model is designed as probabilistic concurrent constraint automata (PCCA).

Thus, LCARS has the following strengths. First, it requires much less training data compared to learning-based works. Second, it requires much less modeling effort compared to other model-based works. These two strengths allows LCARS to be easily applicable to various scenarios.

This paper is organized as follows. Section II provides the formal problem statement and an overview of LCARS. A background is provided in Section III. Section IV illustrates LCARS in detail. Evaluations are provided in Section V. Finally, the paper is concluded in Section VI.

II. Problem Statement and Solution Overview

Figure 1 visualizes the HAR process in general [7]. We first construct the *HAR estimation model* offline. Note that, in offline construction, a learning-based approach relies mostly on training data and a model-based approach relies mostly on domain-specific modeling. After the construction, the *HAR estimation model* estimates (or recognizes) high-level and symbolic *human activity* (e.g., "a human is picking up a hammer") online using low-level *observations*. We assume the low-level *observations* to be *pose observations* of objects relevant to human activities (e.g., a hammer, a drill, a human hand and so on). The *pose observations* can be acquired from robot's camera vision inputs by using any one of the many pose estimation works [8]–[11].

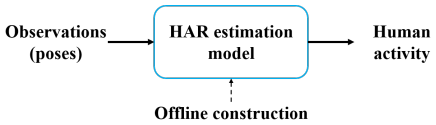


Fig. 1: General process of HAR

In this paper, we emphasize that learning-based approaches rquire a lot of training data; thus, we apply a model-based approach. In Figure 1, the *HAR estimation model* is a direct map from the low-level *pose observations* to the high-level *human activity*. This is a common approach in many previous model-based works [4]–[6]. However, this makes offline domain-specific modeling too complex, limiting the previous works from being applicable to various scenarios.

It might be better if we decompose the *HAR estimation model* to be a two-step map as LCARS (see Figure 2).

The authors are with the MIT CSAIL, Massachusetts Institute of Technology, Cambridge, MA, 02139, USA (e-mail: sangukbo@mit.edu, hofma@mit.edu, williams@mit.edu)

LCARS has two components. The first component, indicated as the *predicate estimator*, computes high-level *predicates* (e.g., "a hand is holding a hammer") from the *pose observations*. The second component, indicated as the *activity estimator*, uses the *predicates* to recognize the *human activity*.

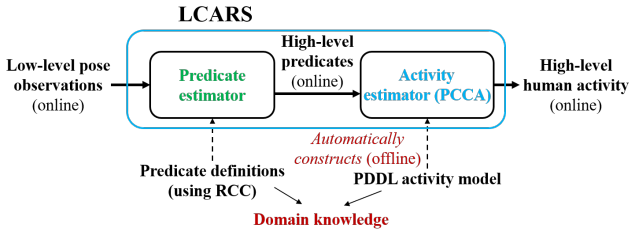


Fig. 2: Graphical representation of LCARS.

Each of the two components of LCARS (i.e., the *predicate estimator* and the *activity estimator*) requires domain knowledge as follows. For the *predicate estimator*, users need to provide the definitions over the *predicates* (i.e., *predicate definitions* in Figure 2). We use RCC, a representation tool in qualitative spatial reasoning (QSR), for the definitions. Users can define the *predicates* with qualitative RCC primitives, rather than quantitative values (e.g., poses). For the *activity estimator*, users need to provide a human activity model written in PDDL (i.e., *PDDL activity model* in Figure 2). PDDL has been very successful for activity modeling, thanks to its simple and intuitive structure. Using RCC and PDDL allows users to encode domain knowledge systematically and intuitively with qualitative and high-level reasoning.

LCARS constructs the *activity estimator* automatically from the *PDDL activity model* as specified in Figure 2. To be more specific, a probabilistic estimation model designed as PCCA is automatically constructed from a deterministic high-level human activity model written in PDDL. Thus, users only have to encode the human activity model in a deterministic and high-level domain, which is very intuitive. In previous HAR works, constructing estimation models required extra user effort, other than providing domain knowledge. The automatic construction minimizes user effort for LCARS. To the authors' knowledge, this is the first paper to discuss the automatic construction of probabilistic estimation models from deterministic high-level PDDL activity models.

III. Background

A. PDDL and Pick-and-Place Example

PDDL is a deterministic and high-level modeling language widely used for activity planning. [12] suggested that PDDL is good for modeling human activities for HAR as well. Table I shows an example *PDDL activity model* used in this paper (called pick-and-place example). The example has four types of activities. A human can pick or place a tool. A human can open or close a drawer. Table I shows the pick activity only.

PDDL represents activities with predicates (e.g., *(empty hand)*, *(holding hammer hand)*). In PDDL, every activity has

TABLE I: Pick-and-Place Example PDDL Model

```
(define (domain pick-and-place)
  (:requirements :strips :typing)
  (:types manipulator object drawer)
  (:predicates
    (in ?o - object ?d - drawer) // ?o is in ?d
    (clear ?o - object) // no manipulator is holding ?o
    (empty ?m - manipulator) // ?m is empty
    (holding ?o - object ?m - manipulator)) // ?m holding ?o
  (:action pick // ?m picks up ?o
    :parameters (?o - object ?m - manipulator)
    :precondition (and (clear ?o) (empty ?m))
    :effect (and (not (clear ?o)) (not (empty ?m)) (holding ?o ?m))))
```

a *precondition* (i.e, a requirement for the activity to happen) and an *effect* (i.e., the result of the activity).

Predicates and activities with unspecified parameters ($?o$, $?d$, and $?m$), such as *(holding* $?o$ $?m$*)* predicate and *(pick* $?o$ $?m$*)* activity, are called "lifted". They are referred to as being "grounded" if the parameters are specified, such as *(holding hammer hand)* and *(pick hammer hand)*. LCARS estimates over grounded activities to distinguish which object a human is picking. A detailed explanation of PDDL is in [13].

B. RCC

RCC represents the spatial relations between objects with a finite number of qualitative relations [14]. We use RCC-5, a variant of RCC, which has five possible qualitative relations between two objects (or regions) *A* and *B*. The five qualitative relations are as follows: *A* is disconnected from *B* ($DC(A,B)$) (i); *A* and *B* are partially occluding each other ($PO(A,B)$) (ii); *A* is identical to *B* ($EQ(A,B)$) (iii); and *A* is a proper part of *B* or the inverse ($PP(A,B)$ or $PPi(A,B)$) (iv and v). Figure 3 visualizes the five relations.

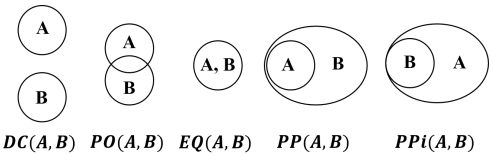


Fig. 3: RCC-5 relations.

TABLE II: Predicate Definitions Using RCC-5

Predicates	Definitions Using RCC-5
$(in\ A\ B)$	$PP(A,B)$
$(holding\ hand\ A)$	$\neg(DC\ hand\ A)$
$(empty\ hand)$	$\forall obj, (DC\ A\ obj)$
$(right\ A\ B)$	$\neg DC(A,B) \wedge (in\ A\ region_{right}(B))$ where $region_{right}(B)$ is shown in Figure 4
$(left\ A\ B)$	$\neg DC(A,B) \wedge (in\ A\ region_{left}(B))$
$(up\ A\ B)$	$\neg DC(A,B) \wedge (in\ A\ region_{up}(B))$
$(down\ A\ B)$	$\neg DC(A,B) \wedge (in\ A\ region_{down}(B))$

We can define the predicates with the RCC-5 relations. For instance, if we want to say that "a hand is empty", we can say that "the hand is DC from all the objects". Table II shows examples of the *predicate definitions* in Figure 2. As we use the qualitative relations, the definitions are much more

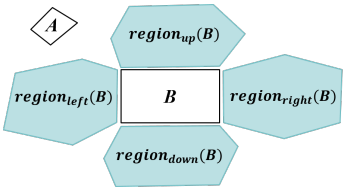


Fig. 4: Regions around B. It can be modified to user's taste.

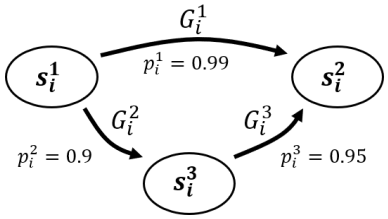


Fig. 5: The state transition model of an example PCA.

intuitive than they would be if using the pose data directly. [15] provides more defitions of frequently used predicates.

In this paper, we use RCC because we assumed the pose measurements for the low-level *observations* (see Figure 1). However, using qualitative representations for the predicate definitions can be extended to any measurements [16].

C. PCCA

PCCA is a probabilistic modeling framework for discrete-time stochastic processes [17], [18]. PCCA is a collection of probabilistic constraint automata operating concurrently. We first define a probabilistic constraint automaton (PCA).

- A PCA is a tuple $\langle \chi_i, T_i, O_i \rangle$:
  - $\chi_i = \{x_i\} \cup \chi_i^r$ is a set of variables for component i, and x_i is a state variable. χ_i^r is a set of attribute variables, including variables used to define the PCA (e.g., observation and guard variables).
  - T_i represents the state transition model for x_i. That is, T_i represents $P(x_i(t+1) \mid x_i(t), \chi_i^r(t))$. Here, $x_i(t)$ represents the state of the variable at time t.
  - O_i represents the observation model. O_i represents observation matrix $P(o_i|x_i)$, where $o_i \in \chi_i^r$.
- Then, PCCA, $\mathcal{A}$, is a set of PCAs.

The state transition model of an i^{th} PCA specifies how the state variable, x_i, changes over discrete time steps. Figure 5 shows an example of a state transition model. In Figure 5, x_i has three possible states: s_i^1, s_i^2, and s_i^3. The directed edges represent the possible state transitions. G_i^1, G_i^2, and G_i^3 along the edges represent guard variables, which are Boolean functions (*True* or *False*) whose arguments are state variables from other PCAs. The guard variable must be *True* in order for the state transition following the corresponding edge to occur. For example, $G_i^1 = (x_j = s_j^1) \vee (x_j = s_j^2)$ states that the state transition from s_i^1 to s_i^2 is possible if, and only if, the state variable x_j from the j^{th} PCA is in state s_j^1 or s_j^2. In fact, one of the strengths of PCCA is that the interconnection between concurrently operating PCAs can be modeled easily with the guard variables. We only need to define when each guard variable can be *True*. p_i^1, p_i^2, and p_i^3 represent the transition probabilities when the corresponding guard variables are *True*. Self transitions are omitted in Figure 5.

We can perform online filtering inference on PCCA. That is, we can compute the probability $P(\hat{x}(t)|\hat{o}(1:t))$. $\hat{x}(t)$ is the set of all the state variables in the PCCA at time t, and $\hat{o}$ is the set of all the observation variables from time 1 to t. An optimal constraint satisfaction problem (OCSP)-based algorithm for PCCA online filtering can be found in [18]. The details on PCCA and its online filtering is available in [18].

IV. Logical Activity Recognition System (LCARS)

LCARS has two components: i) predicate estimator and ii) activity estimator. We explain each component.

A. Predicate Estimator

The predicate estimator computes predicates online from the pose observations. It requires the predicate definitions in terms of RCC-5 primitives as domain knowledge.

Figure 6 visualizes how the predicate estimator works with an example. First, from the pose observations, we determine which of the five RCC-5 relations a pair of objects (*A* and *B*) satisfy. We can compute the RCC-5 relations for any pair of objects. This can be done quickly online using a collision detection algorithm [19]. After we obtain the RCC-5 primitives, we get the predicates online using the predicate definitions (e.g., Table II). We determine whether the logical statements in Table II are *True* or *False* using the computed RCC-5 relations.

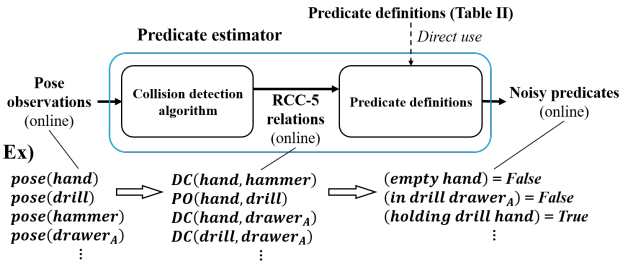


Fig. 6: Predicate estimator operation diagram

We make several remarks. First, the predicates obtained from the predicate estimator are often incorrect because pose observations are noisy. For example, if a hammer is observed to be at the wrong position, the predicate estimator might say that *(holding hammer hand)* is *False* even if a hand is holding the hammer. Thus, we call the predicates obtained from the predicate estimator "noisy predicates" (see Figure 6). The activity estimator handles the noise in the predicates. Second, we do not discuss the offline automatic construction of the predicate estimator from the predicate definitions because we use the definitions directly to compute predicates from the RCC-5 primitives. On the other hand, this is not the case for

TABLE III: The Variables for A_{act} and A_{pred}

Automaton	Variables	Implications	Possible States
A_{act}	x_{act}	State variable	{*Nil*, *Triggered*}
	G^1_{act}, G^2_{act}	Guard variables	{*True*, *False*}
A_{pred}	x_{pred}	State variable	{*True*, *False*}
	o_{pred}	Observation variables	{*True*, *False*}
	G^1_{pred}, G^2_{pred}	Guard variables	{*True*, *False*}

the activity estimator, where we need to convert the PDDL model into a PCCA estimation model (see Figure 2).

B. Activity Estimator

We design the activity estimator as PCCA. We chose PCCA over other modeling frameworks because it is suitable for modeling concurrently changing predicates and activities which heavily affect each other (using gaurd variables). In LCARS, the PCCA is first automatically constructed from the PDDL human activity model (i.e., the domain knowledge for the acticity estimator) offline. Then, we perform online filtering over the PCCA to estimate the current human activity using the noisy predicates as the online observations.

In this subsection, we first explain the structure of the LCARS' PCCA using the example PDDL model in Table I. Next, we explain how the automatic construction is done for any general PDDL models. Finally, we discuss about the online filtering over the PCCA.

1) LCARS' PCCA Design: Our PCCA design has two classes of PCAs: i) activity automata class (C_{act}) and ii) predicate automata class (C_{pred}). For each class, we describe how to get a set of variables, χ_i, the state transition model, T_i, and the observation model, O_i, from the PDDL model.

An activity automaton, $A_{act} \in C_{act}$, indicates whether an activity has happened (or has been triggered) or not. Every grounded activity in the PDDL model has one activity automaton. Every activity automaton has three variables: x_{act}, G^1_{act}, and G^2_{act}. Table III summarizes the variables. x_{act} is the state variable with two possible states: *Nil* and *Triggered*. *Nil* indicates the activity has not happened yet. *Triggered* indicates the activity has happened. G^1_{act} and G^2_{act} are the guard variables for the state transition model, T_{act}. Figure 7(a) shows T_{act}. An activity automaton does not have the observation variable and the observation model, O_{act}.

Every activity automaton has two guard variables: G^1_{act} and G^2_{act}. G^1_{act} represents the condition required for the activity to occur (i.e., the *precondition* of the activity in the PDDL model). For example, according to Table I, the predicates *(clear hammer)* and *(empty hand)* must be true for the *(pick hammer hand)* activity to occur. G^2_{act} represents the reset of the automaton (i.e., transition from *Triggered* to *Nil*) after n time steps have passed after the activity was *Triggered*. This is to reuse the automaton. Table IV summarizes G^1_{act} and G^2_{act}.

A predicate automaton, $A_{pred} \in C_{pred}$ indicates whether a predicate is *True* or *False*. Every grounded predicate in the PDDL model has one predicate automaton. Every predicate

(a) The activity automaton state transition model.

(b) The predicate automaton state transition model.

Fig. 7: State transition models.

TABLE IV: Definitions on Guard Variables

Automaton	Variables	Definitions (Conditions to be *True*)
A_{act}	G^1_{act}	*precondition* satisfied
	G^2_{act}	n time steps have passed after in *Triggered* state
A_{pred}	G^1_{pred}	$\bigvee_{a \in A_{neg}}$ (a in *Triggered* state)
	G^2_{pred}	$\bigvee_{a \in A_{sup}}$ (a in *Triggered* state)

automaton has four variables, which are summarized in Table III: x_{pred}, o_{pred}, G^1_{pred}, and G^2_{pred}. x_{pred} is the state variable with two possible states, indicating whether the predicate is *True* or *False*. o_{pred} is the observation variable that can also be either *True* or *False*. x_{pred} represents the hidden predicate of the ground truth (e.g., a hand is actually holding a hammer), whereas o_{pred} represents the noisy predicate computed from the predicate estimator (e.g., a hand is observed to be not holding a hammer due to noisy pose measurements). G^1_{pred} and G^2_{pred} are the guard variables for the state transition model, T_{pred} (see Figure 7(b)). The observation model for a predicate automaton, written as O_{pred}, is a two-by-two observation matrix representing $P(o_{pred}|x_{pred})$.

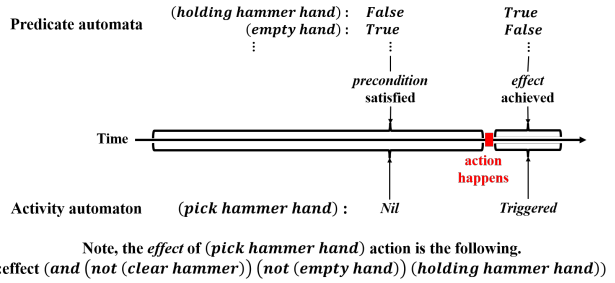


Fig. 8: Graphical representation of how predicates change as the *effect* of an action.

Every predicate automaton has two guard variables: G^1_{pred} and G^2_{pred}. They represent the conditions required for the predicate to change between being *True* and *False*. Based on the PDDL model, the predicate changes from *True* to *False* if it is negated by the *effect* of an activity (e.g., *(not (empty hand))* in the *effect* of *(pick hammer hand)* action in Table I). Likewise, the predicate changes from *False* to *True* if it is supported by the *effect* of an activity (e.g., *(holding hammer hand)* in the *effect* of *(pick hammer hand)*). Figure 8 visualizes how two example predicates (*(holding hammer hand)* and *(empty hand)*) change as the *effect* of

Algorithm 1: Automatic construction of the PCCA

Data: PDDL model
Result: PCCA= $\mathcal{A}$
1 initialization: $\mathcal{A} = \{\}$;
2 **for** $act_i \in$ {*all PDDL activities*} **do**
3 $A_{act}(act_i)$ = Gen_Act_Automaton(*PDDL model*, act_i);
4 $\mathcal{A} \leftarrow \mathcal{A} \cup \{A_{act}(act_i)\}$;
5 **for** $pred_i \in$ {*all PDDL predicates*} **do**
6 $A_{pred}(pred_i)$ = Gen_Pred_Automaton(*PDDL model*, $pred_i$);
7 $\mathcal{A} \leftarrow \mathcal{A} \cup \{A_{pred}(pred_i)\}$;

Algorithm 2: Gen_Act_Automaton

Data: PDDL model, act_i
Result: PCA= $A_{act}(act_i)$
1 initialization: $A_{act}(act_i) = \langle \chi_i, T_i, O_i \rangle$;
2 $\chi_i \leftarrow$ Use Table III and Table IV;
3 $T_i \leftarrow$ Use Figure 7(a);
4 $O_i \leftarrow$ *null*;

the action *(pick hammer hand)* happening (*(clear hammer)* is omitted). Thus, G^1_{pred} of a predicate automaton should be *True* when an activity automaton that negates the predicate in the *effect* has been *Triggered*. G^2_{pred} should be *True* when an activity automaton that supports the predicate in the *effect* has been *Triggered*. There can be multiple actions whose *effects* negate or support the predicate. In this case, the gaurd variables should be *True* when any one of the activities have been *Triggered*. Table IV summarizes G^1_{pred} and G^2_{pred}. In Table IV, A_{neg} and A_{sup} are the sets of activity automata that negate and support the predicate in the *effect*, respectively.

2) Automatic Construction: A PCCA estimation model can be constructed from a PDDL model automatically. The process is summarized in Algorithm 1. Algorithm 1 takes a PDDL model as the input. Line 1 initializes the PCCA, $\mathcal{A}$. In lines 2–4, one activity automaton, $A_{act}(act_i)$, is added to $\mathcal{A}$ for each grounded activity, act_i. In line 3, Gen_Act_Automaton function (Algorithm 2) generates the activity automaton. In lines 5–7, one predicate automaton, $A_{pred}(pred_i)$, is added to $\mathcal{A}$ for each grounded predicate, $pred_i$. In line 6, Gen_Pred_Automaton function (Algorithm 3) generates the predicate automaton.

Each automaton is defined as a tuple $\langle \chi_i, T_i, O_i \rangle$ as described in Section III. Thus, in Algorithm 2 and 3, we formulate χ_i, T_i and O_i for every automaton. We use Table III, Table IV, and Figure 7 in formulating χ_i, T_i and O_i.

We have not discussed how to obtain the transition and observation probabilities for the state transition and observation models, respectively. The probabilities can be learned from a training dataset by applying parameter learning algorithms. To be more specific, as PCCA can be viewed as a special class of dynamic Bayesian networks (DBNs), we can apply DBN parameter learning algorithms such as expectation-maximization (EM) algorithm. We refer to [20] for parameter learning algorithms because our focus is on the structural design of the PCCA rather than the paramter learning. We emphasize that the amount of training dataset required for LCARS is significantly less than that for CNNs, as shown in Section V.

Algorithm 3: Gen_Pred_Automaton

Data: PDDL model, $pred_i$
Result: PCA= $A_{pred}(pred_i)$
1 initialization: $A_{pred}(pred_i) = \langle \chi_i, T_i, O_i \rangle$;
2 $\chi_i \leftarrow$ Use Table III and Table IV;
3 $T_i \leftarrow$ Use Figure 7(b);
4 $O_i \leftarrow$ two-by-two observation matrix ($P(o_{pred_i} | x_{pred_i})$);

3) Online Estimation: We perform online filtering on the automatically constructed PCCA model. We use the noisy predicates computed from the predicate estimator as the online observations for the filtering. That is, the noisy predicates become the online observations for the predicate automata's observation variables, o_{pred}. We perform the online filtering to estimate the current human activity. This corresponds to estimating the activity automata's state variables, x_{act}. We use the OCSP-based PCCA filtering algorithm in [18]. We omit the details on the online filtering using the OCSP-based algorithm, as our main focus is in the modeling of LCARS.

V. Evaluations

In the evaluation, we attempt to prove two hypotheses. First, LCARS requires much less training data compared to learning-based approaches. We compare LCARS with a CNN-based work in [1]. Second, LCARS requires minimal domain-specific modeling compared to other model-based works. We evaluate this by applying LCARS to four different scenarios (two experimental and two simulated scenarios). This is something other model-based works have not done, because each scenario would require a considerable amount of domain-specific modeling effort.

A. Evaluation Scenarios

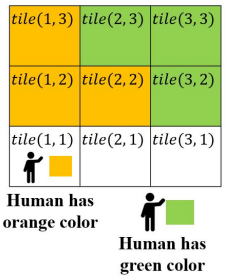
1) Pick-and-Place Example: We performed an experiment using the pick-and-place example in Table I. The experiment was performed in the lab environment shown in Figure 9(a). In the pick-and-place example, there are four types of activities; pick, place, open, and close. A human picks up (places down) an object in pick (place) activity. A human opens (closes) a drawer in open (close) activity. The four types of activities are not yet grounded (i.e., the four activities are lifted), as we have not specified a particular object or drawer to pick, place, open, or close yet.

In the experiment, a subject (a human using a *hand*) could pick or place three objects (*hammer*, *drill*, and *nails*) and open or close two drawers ($drawer_A$ and $drawer_B$). Thus, there were ten grounded activities to estimate (e.g., *(pick hammer hand)*, *(pick drill hand)*, *(place hammer hand)*, *(open $drawer_A$ hand)* and so on). To be more specific, there were three grounded activities for picking up one of the three objects, three grounded activities for placing down one of the three objects, two grounded activities for opening one of the two drawers, and two grounded activities for closing one of the two drawers. Table V summarizes the four lifted actions

TABLE V: Lifted Actions and Domains of the Parameters for the Pick-and-Place Example

Lifted Actions	*(pick ?o ?m)* - $?m$ picks up $?o$	
	(place ?o ?m) - $?m$ places down $?o$	
	(open ?d ?m) - $?m$ opens $?d$	
	(close ?d ?m) - $?m$ closes $?d$	
Parameters and Domains	$?m$ (manipulator)	Domain: $\{hand\}$
	$?o$ (object)	Domain: $\{hammer, drill, nails\}$
	$?d$ (drawer)	Domain: $\{drawer_A, drawer_B\}$

(a) Pick-and-place example environment



(b) Three-by-three tile environment

Fig. 9: Experimental environments

and the domains of three parameters ($?o$, $?d$, and $?m$) in the lifted actions.

In the experiment, non-expert human subjects performed the grounded activities in a random sequence. Meanwhile, we measured the poses of the objects, drawers, and human hand using a vision-based pose estimator [8]. LCARS estimated the human activities using the measured poses.

The four types of activities (or lifted activities)—pick, place, open, and close—are fundamental activities in many HAR works. For example, [1] used the Opportunity Activity Recognition dataset to estimate 18 grounded activities, 14 of which involved opening and closing 7 different drawers and doors.

2) Floor Tile Domain: We performed an experiment using the Floor Tile domain from the international planning competition (IPC). In the Floor Tile domain, a human needs to paint a three-by-three tile environment (see Figure 9(b)). Initially, the human stands on one of the tiles (e.g., *tile(1, 1)*). He/she can move from one tile to another. The human can paint a tile upper or lower from where he/she is standing. There are two paint colors ($orange$ and $green$). The human can perform seven types of activities (i.e., seven lifted activities): i) move-right (i.e., move to the right tile), ii) move-left, iii) move-up, iv) move-down, v) paint upper cell, vi) paint lower cell, and vii) change paint color. There are 50 grounded activities to estimate in total. There are six grounded activities each for moving right, left, up, and down (6×4). There are twelve grounded activities each for painting an upper cell and a lower cell (12×2). There are two grounded activities for changing paint colors (2×1). The human can perform these activities sequentially until he/she finishes painting all the tiles. We refer to [21] for the PDDL model of the Floor Tile domain. We measured the poses of the human, human hands, and the two colors of paint. We used Table II for the predicate definitions.

This domain is similar to the experimental scenario used in [4]. [4] estimated the human activity by tracking human trajectory in a grid cell environment.

3) Simulation Scenarios: We performed simulations on two scenarios from the IPC with some modifications: i) Storage domain and ii) Transport domain [21]. The simulations were performed in a virtual environment where we could generate simulated pose observations. The Storage domain is about a human storing items in containers in a large storage. Some containers are located high and the human needs a ladder to access the container. We used five items and five containers for the simulation. We recognized over four types of activities: i) store an item in a container, ii) climb up the ladder, iii) climb down the ladder, and iv) move between the containers. We simulated the human performing these activities in a sequence. We generated the poses of the human, human hand, the items, and the ladder. The Transport domain is about a human delivering packages from one cell to another. We used five packages and five cells. We recognized over five types of activities: i) pick up an item from a cell, ii) drop an item in a cell, iii) get into a cell, iv) get out of a cell, v) move between cells. We generated the poses of the human, human hand, and the packages.

B. Evaluation Results

We compare LCARS with the CNN in [1]. The CNN is composed of five sections. The first two sections have convolution layers and subsamplig layers. The third section is designed as a convolution layer. The fourth section is designed to unify all sensor measurements. The fifth section is designed as a fully-connected network for the final classification. We refer to [1] for details on the CNN.

For each scenario, we ran LCARS and the CNN online using the pose measurements as online observations (or input features). Table VI summarizes the results.

TABLE VI: Evaluation Results on the Four Scenarios

-		Experiments		Simulations	
		Pick and place	**Floor Tile**	**Storage**	**Transport**
AC	**LCARS**	90.93%	**90.87%**	92.21%	**93.12%**
	CNN	**91.38%**	90.16%	**92.89%**	91.68%
	HMM	90.05%	-	-	-
AF	**LCARS**	58.26	**52.29**	86.13	**85.39**
	CNN	**60.87**	51.48	**89.52**	78.95
	HMM	56.21	-	-	-
NF	**LCARS**	73.94	**67.84**	97.72	**92.65**
	CNN	**78.47**	63.20	**99.32**	86.83
	HMM	72.80	-	-	-
TD	**LCARS**	4000	**4000**	**3000**	**2000**
	CNN	28000	21000	13000	8000
	HMM	**3000**	-	-	-

Following [22], the accuracy (AC), average F-measure (AF), and normalized F-measure (NF) are used to evaluate the preformance of LCARS and the CNN. The accuracy is

calculated as a ratio between the total number of correct recognitions and the total number of samples (i.e., total number of online time steps). For example, if we got correct recognition for 8,000 out of 10,000 samples (i.e., time steps), the accuracy would be 8,000/10,000$\times$100 = 80%. The labels over the correct activities were collected by a human supervisor. We refer to [22] for details on the AF and the NF. Table VI also compares the amount of training data (TD) used to train LCARS and the CNN. The unit of training data used is the number of samples. The training samples were collected in a stream while humans were performing activities sequentially. The samples were collected in a constant rate (10 Hz), thus the number of samples indicates how long it took to collect the training data. The best performance for each evaluation metric is highlighted in bold.

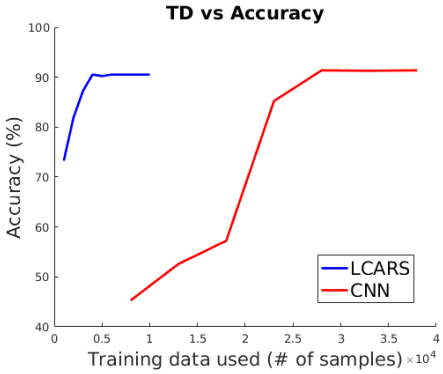
As shown in Table VI, LCARS have comparable accuracy to the CNN. The CNN have higher accuracy (AF and NF as well) in the pick-and-place example and the Storage domain, while LCARS have higher accuracy in the Floor Tile domain and the Transport domain. On the other hand, LCARS requires much less training data. Figure 10 shows how the online estimation accuracies (ACs) change when varying the amount of training data used. We trained both LCARS and the CNN with varying amount of training data. Then, we performed online activity estimation with (partly-trained) LCARS and CNN. Note, LCARS reaches its maximum accuracy with much less training data.

In Table VI, we included the result with another model-based work [12] for the pick-and-place example. This work used a hidden Markov model (HMM) to model human activities (thus we refer to this work as "HMM work"). The experimental scenario in [12] is similar to the pick-and-place example. LCARS maintains good accuracy compared to the HMM work. For the HMM work, we only include the results for the pick-and-place example because much domain-specific modeling effort is required for it to be applied to other scenarios. This is because the HMM work requires complex integration step using Gaussian assumption for sensor measurements. We refer to [12] for the details on the HMM work.

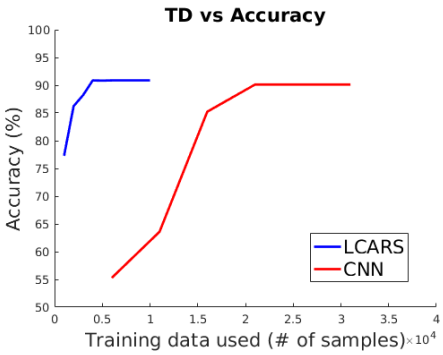
On average, LCARS performed online filtering within 0.1 seconds for a single sample (or time step). Thus, LCARS is computationally efficient enough for online operation. Having more grounded predicates and activities would increase the computational burden, but the additional burden would not grow fast. This is because the automata in the LCARS PCCA model are sparsely connected. For instance, in the pick-and-place example in Section V-*A-1)*, the automaton for *(pick hammer hand)* activity is only connected to three automata for the following predicates: *(clear hammer)*, *(empty hand)*, and *(holding hammer hand)*.

VI. Conclusions

This paper presents LCARS, a model-based HAR system. As we use a model-based approach, LCARS requires much less training data compared to learning-based works. LCARS has two components: i) predicate estimator and ii) activity estimator. Each requires domain knowledge that is intuitive to encode using RCC and PDDL. The activity estimator designed as PCCA is constructed automatically from the PDDL human activity model. Compared to other model-based works, LCARS requires minimal domain-specific modeling effort, thanks to the intuitive domain knowledge and the automatic construction. Requiring minimal training data and modeling effort allows LCARS to be easily applicable to various scenarios.



(a) Pick-and-place example



(b) Floor tile domain

Fig. 10: Training data used vs. accuracy

LCARS uses RCC to capture the qualitative spatial relations between objects effectively. However, RCC is insufficient at capturing the qualitative relations between moving objects. For example, a qualitative relation such as "a hand is moving toward a hammer" is not captured with RCC. This information can be valuable for HAR and even further for human intent recognition. Future efforts could focus on applying other qualitative representations such as qualitative trajectory calculus, which is designed to capture the qualitative movements between objects.

References

[1] J. B. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition," in *International Joint Conferences on Artificial Intelligence (IJCAI)*, vol. 15, 2015, pp. 3995–4001.

[2] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, "Convolutional Neural Networks for Human Activity Recognition Using Mobile Sensors," in *Mobile Computing, Applications and*

Services (MobiCASE), 2014 6th International Conference on. IEEE, 2014, pp. 197–205.

[3] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler *et al.*, "Collecting Complex Activity Datasets in Highly Rich Networked Sensor Environments," in *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*. IEEE, 2010, pp. 233–240.

[4] N. T. Nguyen, D. Q. Phung, S. Venkatesh, and H. Bui, "Learning and Detecting Activities from Movement Trajectories Using the Hierarchical Hidden Markov Model," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 955–960.

[5] M. Awais and D. Henrich, "Human-robot Collaboration by Intention Recognition Using Probabilistic State Machines," in *Robotics in Alpe-Adria-Danube Region (RAAD), 2010 IEEE 19th international Workshop on*, Balantonfured, Hungary, Jun. 2010, pp. 75–80.

[6] R. Kelley, A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis, "Understanding Human Intentions via Hidden Markov Models in Autonomous Mobile Robots," in *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*. ACM, 2008, pp. 367–374.

[7] C. Heinze, "Modelling Intention Recognition for Intelligent Agent Systems," Defence Science and Technology Organisation Salisbury (Australia) Systems Sciences Lab, Tech. Rep. DSTO-RR-0286, Nov. 2004.

[8] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects," *arXiv preprint arXiv:1809.10790*, 2018.

[9] J. Wu, E. Lu, P. Kohli, B. Freeman, and J. Tenenbaum, "Learning to See Physics via Visual De-Animation," in *Advances in Neural Information Processing Systems*, 2017, pp. 153–164.

[10] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6D Object Pose Estimation Using 3D Object Coordinates," in *European conference on computer vision*. Springer, 2014, pp. 536–551.

[11] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1521–1529.

[12] K. Yordanova, F. Kruger, and T. Kirste, "Context Aware Approach for Activity Recognition Based on Precondition-Effect Rules," 2012.

[13] M. Fox and D. Long, "PDDL 2.1: An Extension to PDDL for Expressing Remporal Planning Domains," *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, Dec. 2003.

[14] A. G. Cohn, B. Bennett, J. Gooday, and N. M. Gotts, "Qualitative Spatial Representation and Reasoning with the Region Connection Calculus," *GeoInformatica*, vol. 1, no. 3, pp. 275–316, Oct. 1997.

[15] C. Schlenoff, A. Pietromartire, Z. Kootbally, S. Balakirsky, and S. Foufou, "Ontology-based State Representations for Intention Recognition in Human–robot Collaborative Environments," *Robotics and Autonomous Systems*, vol. 61, no. 11, pp. 1224–1234, Nov. 2013.

[16] J. De Kleer and J. S. Brown, "A Qualitative Physics Based on Confluences," *Artificial intelligence*, vol. 24, no. 1-3, pp. 7–83, 1984.

[17] B. C. Williams, M. D. Ingham, S. H. Chung, and P. H. Elliott, "Model-Based Programming of Intelligent Embedded Systems and Robotic Space Explorers," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 212–237, 2003.

[18] O. Martin, B. C. Williams, and M. D. Ingham, "Diagnosis as Approximate Belief State Enumeration for Probabilistic Concurrent Constraint Automata," in *Proceedings of the Twentieth National Conference on Artificial Intelligence*, Pittsburgh, PA, Jul. 2005, pp. 321–326.

[19] S. U. Lee, A. Jasour, A. Hofmann, and B. Williams, "Robust Human Activity Monitoring Using Qualitative Spatial Representation and Reasoning," in *2018 International Conference on Automated Planning and Scheduling (ICAPS) Workshop on Planning and Robotics (PlanRob)*, Delft, Netherlands, Jun. 2018.

[20] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[21] M. Vallati, L. Chrpa, M. Grześ, T. L. McCluskey, M. Roberts, S. Sanner *et al.*, "The 2014 International Planning Competition: Progress and Trends," *Ai Magazine*, vol. 36, no. 3, pp. 90–98, 2015.

[22] H. Cao, M. N. Nguyen, C. Phua, S. Krishnaswamy, and X. Li, "An Integrated Framework for Human Activity Classification," in *UbiComp*, 2012, pp. 331–340.