A Robust Online Human Activity Recognition Methodology for Human–Robot Collaboration

Sang Uk Lee and **Andreas Hofmann** and **Brian Williams**
MIT CSAIL
32 Vassar St., Cambridge, MA, 02139
{sangukbo, hofma, williams}@mit.edu

Abstract

Human activity recognition is a crucial ingredient in safe and efficient human–robot collaboration. In this paper, we present a new model-based approach for online human activity recognition. Our algorithm, called the logical activity recognition system (LCARS), has three key points and corresponding strengths. First, LCARS uses a deterministic high-level human activity model as its primary knowledge base. This makes LCARS intuitive and user-friendly because its users only have to work in a deterministic and high-level domain. Second, LCARS automatically compiles the deterministic high-level human activity model into a probabilistic model. This makes LCARS robust to sensor noise. Third, LCARS uses a qualitative map from the low-level pose (position and orientation) observations sensed by a robot to the high-level domain, as LCARS uses the high-level human activity model. Using the qualitative map makes LCARS complete in the sense that it can use low-level pose observations directly as the input, without any additional effort. Experimental results to support our claims will be provided.

Introduction

Human–robot collaboration is gaining increasing attention. There are many scenarios where humans and robots need to collaborate, such as manufacturing and household environments. Among many ingredients needed for successful human–robot collaboration, human activity recognition is crucial in ensuring safe and effective collaboration. To illustrate this, let us assume a scenario in which a human and a robot are collaborating to make a wooden chair. In this scenario, a hammer and a drill are in Toolbox A, and a box of nails is in Toolbox B. If the robot recognizes that the human is picking up the hammer, it can avoid a collision by not going to Toolbox A. Instead, it can go to Toolbox B to pick up the nails for the hammer.

In this paper, we present a new model-based online human activity recognition algorithm called the logical activity recognition system (LCARS). LCARS has three inputs: i) a deterministic high-level human activity model, ii) a qualitative map from the low-level pose observations sensed by a robot to a high-level domain, and iii) noisy online low-level pose (position and orientation) observations of all the objects (including the human hand, the hammer, and the drill). To be robust to the noisy observations, LCARS first automatically compiles the deterministic high-level human activity model into a probabilistic high-level human activity model (referred to as the "probabilistic model" from now on). After the compilation, LCARS uses the probabilistic model, the qualitative map, and the noisy online pose observations to estimate the current human activity online (e.g., "the human is picking up the hammer").

For the implementation of LCARS, we use specific frameworks for the deterministic high-level human activity model, the probabilistic model, and the qualitative map. First, to encode the deterministic high-level human activity model, we use the planning domain definition language (PDDL), a language widely used in activity planning and execution monitoring. (Yordanova 2011; Yordanova, Kruger, and Kirste 2012) have shown that PDDL is a good choice, as it encodes how human activity changes the world with deterministic *preconditions* and *effects*. Second, we design the probabilistic model as a dynamic Bayesian network (DBN). Third, for the qualitative map, we use qualitative spatial reasoning (QSR). QSR has proven to be effective in human activity recognition (Schlenoff et al. 2013; 2015).

The main contribution of our work is combining the three existing frameworks (i.e., PDDL human behavior model, QSR, and DBN) to come up with LCARS. The resulting LCARS has the following strengths. First, it is intuitive and user-friendly, thanks to its use of a deterministic high-level human activity model. Second, it is robust due to its use of a probabilistic model. Third, it is complete in the sense that it can use low-level pose observations directly as the input, without any additional effort, thanks to the use of a qualitative map.

This paper is organized as follows. The second section provides the related works. The third section provides the formal problem statement and an overview of LCARS. The fourth section provides a pick-and-place example used throughout the paper. The background is presented in the fifth section, with a brief explanations of PDDL, QSR, and DBNs. A detailed illustration of LCARS is presented in the sixth section. The seventh section presents the experimental results. Finally, the paper is concluded in the eighth section.

Related Works

Human activity recognition is a broad and complex research field that requires modeling for a wide variety of elements. Many works have tried to address some elements. To name a few, first, some have modeled a map from low-level pose input from robot sensors to abstract concepts that humans think and reason with. This allows researchers to focus on a higher level and more intuitive domain. (Schlenoff et al. 2013; 2015; Kirste 2011) extracted the high-level concepts (e.g., predicates such as "the human is holding the hammer") for various human activity recognition scenarios from the low-level pose information using QSR.

Second, some have focused on how to model the human behavior. As human activity is a high-level abstract concept, it is reasonable to describe human activity (or behavior) with high-level languages. (Yordanova 2011) used PDDL to code the human behavior with preconditions and effects. (Pynadath and Wellman 2000) models the human behavior using probabilistic state-dependent grammars (PSDGs).

Third, some have focused on how to be robust to noisy environment. Robots' sensing capabilities are limited in many real-world scenarios. Thus, it is helpful to resort to probabilistic approaches rather than deterministic ones. (Bui 2003) applied DBN for recognizing human activity by tracking the trajectory of a human. In (Yordanova, Kruger, and Kirste 2012), a hidden Markov model (HMM) is extracted from the human behavior encoded in PDDL.

Our work tries to integrate these separate elements under one framework. Especially, (Pynadath and Wellman 2000; Yordanova, Kruger, and Kirste 2012) lack a systematic way to model the map from low-level input to high-level concepts. In addition, in contrast to (Pynadath and Wellman 2000), we use more general PDDL to model human behavior so that LCARS can be integrated to other planning and execution monitoring works easily. In contrast to (Yordanova, Kruger, and Kirste 2012), we automatically synthesize a DBN from a PDDL model. DBN is more expressive compared to HMM, and thus it can express more complex interactions between state and action variables. The simple HMM structure in (Yordanova, Kruger, and Kirste 2012) has a collapsed state variable, without action variables defined separately. This makes modeling complex actions, such as durative actions in PDDL 2.1, difficult.

Problem Statement and Solution Overview

The task of human activity recognition is to estimate the activities that a human is performing (i.e., pick, place etc.). If the activity has a temporal duration, such as a durative action in PDDL 2.1, we also need to recognize how far we have progressed in the activity (i.e., ready, executing, finished, etc.). In this paper, we call this progression the activity stage. Then, our goal is to estimate human activity and its stage.

To state the human activity recognition problem more formally, let us assume that the *true activity model* models how human activity (and its stage) changes the world. This can be represented as a function $x_{1:t} = f(m_{1:t}, w_{1:t})$, where $x_{1:t}$ represents the positions and orientations of objects, $m_{1:t}$ represents the human activity and its stage, and $w_{1:t}$ represents all possible random aspects of human activities. We can sense $x_{1:t}$ only through a noisy sensor. The function $o_{1:t} = h(x_{1:t}, v_{1:t})$ represents the *observation model*, where $o_{1:t}$ is a noisy observation of objects' poses, and $v_{1:t}$ is a sensor noise. Then, the aim of the human activity recognition problem becomes, first, to model an *activity estimation model*, $\hat{m}_t = \hat{f}^{-1}(o_{1:t})$, which is pseudo-inverse mapping of the *true activity model*, and second, to perform an efficient online estimation over the *activity estimation model*. Here, $\hat{m}_t$ represents the recognized (or estimated) human activity and its stage. Figure 1 visualizes the above formal human activity recognition problem (Heinze 2004).

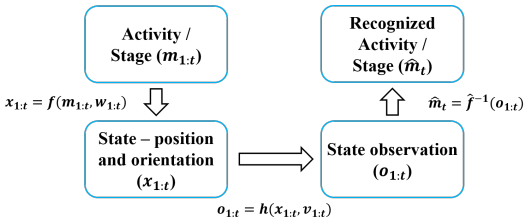


Figure 1: Human activity monitoring problem

In this paper, we provide our solution, called LCARS. To solve the above general human activity recognition problem, LCARS addresses and combines three elements: i) Spatial relations between objects are of great interest. Thus, we use QSR to map from low-level pose input to PDDL predicates. ii) Humans act according to a predefined abstract behavior model given in PDDL. iii) The PDDL model is converted into DBNs for robustness to noisy (pose) observations. Figure 2 visualizes the overview of LCARS. In the big picture, LCARS takes three inputs: online noisy pose observation of objects at each time step, user-specified definitions over PDDL predicates in terms of QSR primitives (Table 2), and PDDL human behavior model (Table 1). LCARS outputs the online estimation (probability distribution) over the current human activities and stages, as well as the predicates representing the state of the world.

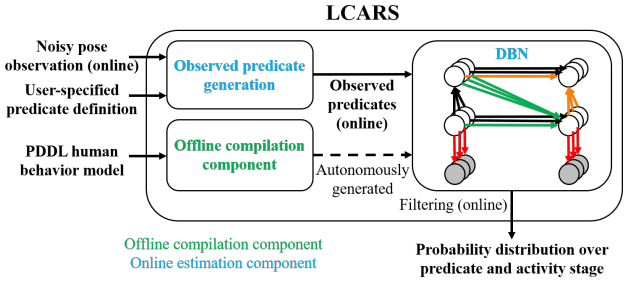


Figure 2: Graphical representation of LCARS

LCARS has two components: i) the offline compilation component and ii) the online estimation component. The offline part uses the PDDL human behavior model to generate the DBN part of the online estimation component offline. The conversion of the PDDL model into the DBN

can be automated for a new PDDL model. The online part has two subcomponents: i) the observed predicate generator, which is explained in detail in the sixth section, and ii) the DBN estimator. The observed predicate generator uses user-specified PDDL predicate definitions in terms of QSR primitives, which serve as the mapping from low-level pose data to abstract PDDL predicates, and online noisy pose observation as inputs to get the observed predicate online. The online observed predicates generated at each time step are used in the DBN estimator to estimate the current human activities and stages, as well as the predicates online.

In Figure 2, any inputs, outputs, or procedures indicated as *online* are provided, obtained, or performed at each time step. Otherwise, they are only for once.

Pick-and-Place Example

In this paper, we use a pick-and-place example throughout to help readers understand LCARS. The experimental results in the seventh section are also based on this example. Our pick-and-place example has three actions (or activities): pick, place, and pass. Although there are only three action types, they are fundamental actions for many human–robot collaboration scenarios that can be modified to express many other actions. We provide part of a pick-and-place example PDDL code (pick action only) in Table 1. This example is written in PDDL version 2.1 with durative actions. Thus, LCARS needs to estimate not only the action itself but also the activity stage, unlike in (Yordanova 2011; Yordanova, Kruger, and Kirste 2012).

All the predicates and actions in Table 1 are ungrounded, meaning that the parameters ($?o$, l, and $?m$) are not specified. Predicates and actions are called grounded when all the parameters are specified. For example a predicate *(holding* $hammer$ *hand)* and an action *(pick* $hammer$ *hand* $toolbox_A$*)* are grounded. All possible values (e.g., $hammer$, $drill$, and so on) for the parameters (e.g., $?o$) are specified in a separate format (omitted here). Note LCARS estimates over the grounded activities and predicates.

Algorithm Background

Planning Domain Definition Language (PDDL)

PDDL is a predicate-based language widely used in activity planning and execution monitoring. PDDL describes actions with preconditions and effects. An example PDDL 2.1 code is provided in Table 1. For a durative action, a *precondition* (or *condition*) is a predicate statement that must be true to perform the action, and an *effect* is a predicate statement that results in being true from applying the action. An *at start* indicates a predicate statement related to the beginning of an action, and an *at end* indicates a predicate statement related to the end. The *at start* and *at end* are sometimes referred to as snap actions. Durative actions progress in the order of *at start precondition*, *at start effect*, *at end precondition*, and *at end effect*. An *over all* indicates a predicate statement related to the duration between the start and the end. A graphical representation of a durative action is shown in Figure 5. A detailed explanation of PDDL 2.1 is in (Fox and Long 2003).

Table 1: Pick-and-Place Example in PDDL

```
(define (domain PDDL-domain)
  (:requirements :strips :typing :durative-actions)
  (:types manipulator object location)
  (:predicates
    (in ?o - object ?l - location) // ?o is in ?l
    (clear ?o - object) // no manipulator is holding ?o
    (empty ?m - manipulator) // ?m is empty
    (holding ?o - object ?m - manipulator)) // ?m holding ?o
  (:durative-action pick // ?m picks up ?o from ?l
    :parameters (?o - object ?m - manipulator ?l - location)
    :duration (= ?duration 20)
    :condition (and
      (at start (in ?o ?l)) (at start (clear ?o)) (at start (empty ?m))
      (at end (in ?o ?l)) (at end (not (clear ?o)))
      (at end (not (empty ?m))) (at end (holding ?o ?m))
      (over all (in ?o ?l)) (over all (not (clear ?o)))
      (over all (not (empty ?m))) (over all (holding ?o ?m)))
    :effect (and
      (at start (not (clear ?o))) (at start (not (empty ?m)))
      (at start (holding ?o ?m)) (at end (not (in ?o ?l))))))
```

QSR and RCC-8

QSR (Freksa 1991) abstracts the pose data (positions and orientations) of objects or regions into qualitative relations among them for intuitive reasoning. We can represent complex and high-level PDDL predicates through such reasoning. RCC (Cohn et al. 1997) is a promising tool for the abstraction. In RCC, only a finite number of qualitative relations are possible for any two given objects or regions. The number is 5 for RCC-5, 8 for RCC-8, and 23 for RCC-23. RCC-8 is used in this paper because it is rich enough.

Other human activity recognition works, such as (Schlenoff et al. 2013; 2015), used a different variant called RCC-3D to represent 3D ideas of up and down or left and right. However, RCC-8 is rich enough to represent 3D ideas as well, and using RCC-3D only complicates the QSR process. How RCC-8 can represent 3D ideas and replace RCC-3D is described in (Lee et al. 2018).

In RCC-8, the finite relations are i) *A* is disconnected from *B* ($DC(A,B)$), ii) *A* is edge-connected with *B* ($EC(A,B)$), iii) *A* is partially occluded by *B* ($PO(A,B)$), iv) *A* is identical to *B* ($EQ(A,B)$), v) and vi) *A* is a tangentially proper part of *B*, or the inverse ($TPP(A,B)$ or $TPPi(A,B)$), vii) and viii) *A* is a nontangentially proper part of *B*, or the inverse ($NTPP(A,B)$ or $NTPPi(A,B)$). These relations are visualized in Figure 3.

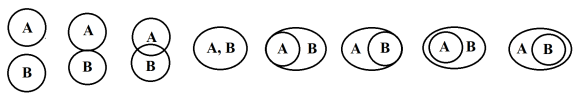


Figure 3: RCC-8 primitives

Given two closed regular regions *A* and *B* in $\mathbb{R}^3$ space, we can acquire an RCC-8 relation using a collision detection algorithm. Collision detection has been a well-studied

field with many efficient algorithms (Ericson 2004). Thus, we can find the RCC-8 primitive statement very efficiently. How to obtain the RCC-8 primitives using a collision detection algorithm is described in (Lee et al. 2018).

Dynamic Bayesian Networks (DBNs)

A DBN is a directed probabilistic graphical model (Murphy 2002). It is very similar to a Bayesian network (BN) except that the DBN is used to model a discrete-time stochastic process. An HMM, a popular modeling method for discrete-time stochastic processes, is an instance of a DBN. Let us assume Z_t is a collection of random variables of a DBN at time t. Then, a DBN is defined to be a pair, $(B_1, B_{\rightarrow})$, where B_1 is a BN that defines the prior $P(Z_1)$, and $B_{\rightarrow}$ is a two-slice temporal BN (2TBN) that defines $P(Z_t \mid Z_{t-1})$ by means of a directed acyclic graph (DAG). The nodes of the DAG represent random variables, and the edges of the DAG represent the conditional probability distribution (CPD). Note Z_t can be partitioned into $Z_t = (X_t, O_t)$, where X_t is the collection of hidden random variables, and O_t is the collection of observation variables. An example DBN is shown in Figure 4. In Figure 4, the hidden variables are colored white, and the observation variables are colored gray. Directed edges represent the CPD by means of DAG.

The inference on a DBN is very similar to the inference on a BN. In this paper, we perform online filtering over the generated DBN using online observations. There are several exact inference algorithms, such as frontier algorithm, junction tree algorithm, and so on. For an approximate inference, the particle filter algorithm is widely used. A detailed explanation of the DBN and its inference is omitted here and can be found in (Murphy 2002).

Logical Activity Recognition System (LCARS)

LCARS has two components: i) an offline compilation component and ii) an online estimation component. In this section, we explain the two components.

Offline Compilation Component

In the offline compilation, the high-level human behavior model in PDDL is taken in as the input to generate the DBN offline. The DBN is used in the online estimation. In this subsection, our unique DBN modeling methodology for converting the PDDL human behavior model is presented. To be more specific, we specify how to obtain all the random variables (represented as nodes) and CPDs (represented as edges) for the DBN from the PDDL human behavior model. Our DBN modeling has i) three types of random variables and ii) three types of CPDs. We explain our DBN design with the pick-and-place example. The DBN for the pick-and-place example is provided in Figure 4.

Let us first go over the three types of random variables, corresponding to three levels of nodes in the LCARS' DBN design in Figure 4. The level 1 layer is for observed predicate variables ($pred_t^{obs}$), which are formed in parallel to each other. An observed predicate variable ($pred_{t,i}^{obs}$) is formed for every grounded predicate in PDDL. $pred_t^{obs}$ represents

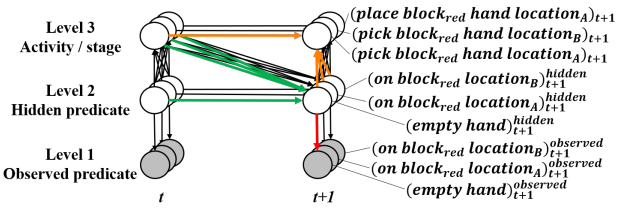


Figure 4: DBN modeling for the pick-and-place experiment in the seventh section. Only a small portion is shown here.

the collection of all the parallel observed predicate variables, whereas $pred_{t,i}^{obs}$ represents the ith observed predicate variable in the collection. The observed predicate variables represent the noisy predicates obtained from the observed predicate generator using the noisy pose observation. The observed predicate generator is explained in the next subsection. The observed predicate variables are the observation variables (O_t) in the DBN and colored gray in Figure 4. The level 2 layer is for hidden predicate variables ($pred_t^h$), which are formed in parallel to each other. A hidden predicate variable ($pred_{t,i}^h$) is formed for every grounded predicate as well. The hidden predicate variables represent the world truth predicates, whereas the observed predicate variables in level 1 do not. The observed predicate variables and the hidden predicate variables have two possible states, *True* and *False*. The level 3 layer is for activity variables (act_t), which are also formed in parallel to each other. An activity variable ($act_{t,i}$) is generated for every grounded (durative) action in PDDL. The activity variables are to tell the human activity and its stage. Activity variables have six possible states, representing the activity stages. They are *Nil*, *Ready*, *Executing*, *Almost*, *Finished*, and *Failed*. The *Nil* stage means the *at start precondition* of a grounded action has not been satisfied yet. This means the action not only has not been initiated but also has not even been prepared for initiation. The *Ready* stage means the *at start precondition* has been satisfied. This means the action is ready to be initiated. *Executing* means the *at start effect* has been achieved. This means the action has been initiated. *Almost* means the *at end precondition* has been satisfied. This means the precondition to finish the action has been satisfied. *Finished* means the *at end effect* has been achieved. This means the action has been finished. *Failed* means the action has been failed. The six stages of an action variable are graphically represented in Figure 5.

For example, let us assume $(empty\ hand)_t^h = True$ and $(holding\ hammer\ hand)_t^h = False$ for the two hidden predicate variables, and $(pick\ hammer\ hand\ toolbox_A)_t = Ready$ and $(place\ hammer\ hand\ toolbox_A)_t = Nil$ for the two activity variables. This means the hand is empty and not holding $hammer$ at time t. In addition, the hand has satisfied the *at start precondition* for the pick action and is ready for picking up $hammer$ from $toolbox_A$, while the hand has not satisfied the precondition for the place action and is not ready for placing $hammer$ on $toolbox_A$ at time t. Note, we implicitly assumed that multiple actions can be in the *Executing* stage in the DBN model through a set of par-

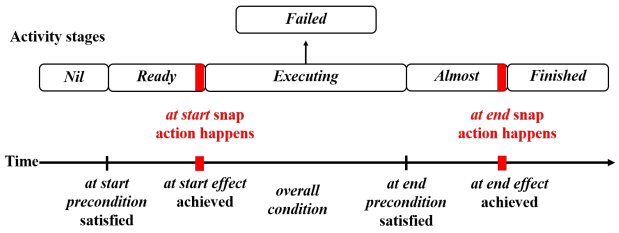
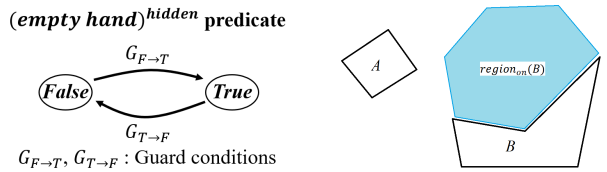


Figure 5: A diagram of temporal progression of a PDDL durative action (bottom) and a diagram of how the temporal progression corresponds to the activity stages (top)

allel activity variables. This means that humans can perform multiple actions at the same time, and LCARS can estimate all those multiple actions.

Next, let us go over the three types of CPDs in the LCARS' DBN. Three types of CPDs are represented in Figure 4 with colored directed edges (red, green, and orange). A red edge represents an observation model between an observed predicate variable and a hidden predicate variable ($P(pred^{obs}_{t,i} \mid pred^{h}_{t,i})$). The observation models are two-by-two matrices. A set of green edges represents a state transition model for a hidden predicate variable (a CPD of form $P(pred^{h}_{t+1,i} \mid pred^{h}_{t,i}, act_t)$). It is formed for every hidden predicate variable. A set of orange edges represents a state transition model for an activity variable (a CPD $P(act_{t+1,i} \mid act_{t,i}, pred^{h}_{t+1})$). It is formed for every activity variable.

Let us go over the state transition models for green edges and orange edges in more detail. For the green edges, let us consider the state transition model of an example hidden predicate *(empty hand)*. Figure 6(a) shows the state transition model. The state transition is possible only when a specific condition has been met. For example, the *(empty hand)* experiences $True \rightarrow False$ when *(pick hammer hand* $toolbox_A$*)* action's *at start* snap action happens. This, in turn, means that the predicate's state transition can happen when the action is in the *Ready* stage. This is because the *at start* snap action can happen when the action is in the *Ready* stage, as shown in Figure 5. This condition can be modeled as a guard condition $G_{T \rightarrow F}$, indicated along the transition arrow in Figure 6(a). The guard condition in this example would be "*(pick hammer hand* $toolbox_A$*)* in the *Ready* stage." Because there can be many other grounded actions that can make the *(empty hand)* predicate change, these actions should also be included in the guard conditions through disjunctive statements. This justifies the CPD of green edges being given as $P(pred^{h}_{t+1,i} \mid pred^{h}_{t,i}, act_t)$, because the state transition of a hidden predicate variable should depend on the activity variables. For the orange edges, let us consider the state transition model of an example action *(pick hammer hand* $toolbox_A$*)*. Figure 7 shows the state transition model. The state transition model in Figure 7 is modified from the one in (Wang and Williams 2015), where the concept of temporal stages for a grounded durative action in PDDL was introduced. By the definition, the pick action experiences the stage transition *Nil* $\rightarrow$ *Ready* when the *at start precondition* is satisfied. That is, ($act_{t,i}$ in *Nil*) $\wedge$ (*at start precondition* satisfied at time $t+1$) $\rightarrow$ ($act_{t+1,i}$ in *Ready*). This is represented as a guard condition between *Nil* and *Ready* in Figure 7. Figure 7 shows the guard conditions for all possible activity stage transitions.



(a) Hidden predicate transition model (self transitions omitted)

(b) *A*, *B*, and $region_{on}(B)$

Figure 6: The predicate transition model and $region_{on}(B)$

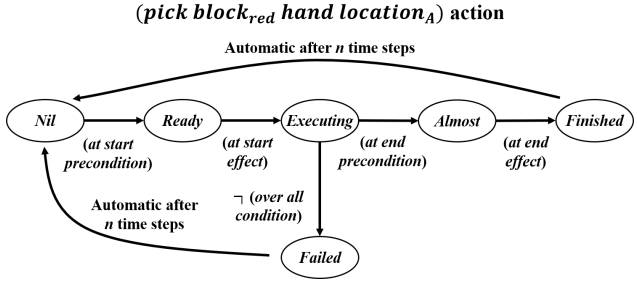


Figure 7: State transition model for activity stages (self transitions omitted)

The state transition models for other activity variables have the same structure as in Figure 7, except for the guard conditions. We need to write in the appropriate guard conditions that correspond to *preconditions* and *effects* in PDDL code. As they are explicit in the PDDL code, the process is very simple. The state transition models for other hidden predicate variables also have the same structure as in Figure 6(a), except for the guard conditions. The guard conditions, $G_{T \rightarrow F}$ and $G_{F \rightarrow T}$, are not explicit in this case, however. How to obtain the guard conditions is provided in Algorithm 1, for the jth hidden predicate variable $pred^{h}_{j}$.

We would like to reemphasize that the generation of the DBN part of LCARS from a PDDL model can be automated. For a new PDDL, we first generate all the variables for grounded predicates and actions. Second, we formulate the observation models between the observed predicate variables and the hidden predicate variables. Third, we formulate the state transition models for the hidden predicate variables and the activity variables. Here, we need to modify the guard conditions appropriately in Figure 6(a) and Figure 7 for different state transition models. This process is repetitive and can be automated.

As a final remark, we have not discussed how to obtain the numeric probability values for the observation models and the state transition models, after we found all the guard conditions. The observation probabilities and transition probabilities should be learned from a training dataset. This is

Algorithm 1: How to obtain the guard conditions for a hidden predicate variable $pred_j^h$ from PDDL

Data: PDDL human behavior model
Result: Guard conditions for $pred_j^h$ ($G_{T\to F}$ and $G_{F\to T}$)
initialization: $G_{T\to F} = nil$ and $G_{F\to T} = nil$;
for $act_i \in \{all\ PDDL\ grounded\ actions\}$ **do**
  switch do
    case $pred_j^h$ *becomes* $True$ *in at start effect* **do**
      $G_{F\to T} \leftarrow G_{F\to T} \vee (act_i\ in\ Ready\ stage)$;
    case $pred_j^h$ *becomes* $False$ *in at start effect* **do**
      $G_{T\to F} \leftarrow G_{T\to F} \vee (act_i\ in\ Ready\ stage)$;
    case $pred_j^h$ *becomes* $True$ *in at end effect* **do**
      $G_{F\to T} \leftarrow G_{F\to T} \vee (act_i\ in\ Almost\ stage)$;
    case $pred_j^h$ *becomes* $False$ *in at end effect* **do**
      $G_{T\to F} \leftarrow G_{T\to F} \vee (act_i\ in\ Almost\ stage)$;

omitted here, since it is not our focus. We can apply various existing parameter learning algorithms to obtain the numeric probabilities. Note, learning the parameters is going to be much faster with guard conditions precomputed, compared to learning the parameters without precomputing guard conditions. This is because the gaurd conditions set transition probability parameters to 0 for many impossible cases.

Online Estimation Component

In the online estimation, LCARS estimates the probability distributions over the activity stages and predicates. The inputs for the LCARS' online estimation are i) user-specified PDDL predicate definitions and ii) online noisy pose observation of all the objects. The online estimation component has two subcomponents: i) the observed predicate generator and ii) the DBN estimator.

Observed Predicate Generator The observed predicate generator serves as a bridge between the online noisy pose observation from a robot's low-level sensors and the observed predicate variables, which are observation variables of the LCARS' DBN. The DBN generated from the offline compilation does not include the online noisy pose observation directly. The online pose observations are included indirectly through the observed predicate variables, which represent PDDL predicates. Thus, we convert the pose observation into PDDL predicates online using the observed predicate generator. The generated PDDL predicates are then used for the observation variables in the online DBN filtering. Because the online pose observation is noisy, the acquired PDDL predicates from the observed predicate generator are noisy as well, meaning they might not be correct compared to the ground truth. This is why we named them *observed predicates* and distinguished them from *hidden predicates*. We resort to the DBN estimator to handle the noisy observed predicates.

In summary, the observed predicate generator takes in the user-specified PDDL predicate definitions (once) and the online noisy pose observation of objects (at each time step) as inputs to generate the observed predicates online. The online observed predicates (at each time step) are then used in the DBN estimator. The operation diagram for the observed predicate generator is provided in Figure 8 with an example.

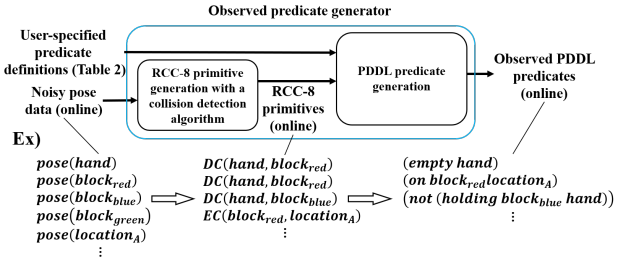


Figure 8: Observed predicate generator operation diagram

From the online noisy pose observation of objects, we can get the RCC-8 primitives online using a collision detection algorithm, as explained in the fifth section. The resulting online RCC-8 primitives can then be used to get PDDL predicates online based on the provided user-specified PDDL predicate definitions. The user-specified PDDL predicate definitions define all the PDDL predicates in terms of RCC-8 primitives. The exemplary definitions are provided in Table 2. We directly apply these logical statements to the online RCC-8 primitives to get PDDL predicates. As we are using qualitative relations, the definitions are much more intuitive compared to using the pose data directly. More definitions can be found in (Vieu 1993).

Table 2: Some PDDL Predicate Definitions Using RCC-8

Predicates	Primitive representation
$(in\ A\ B)$	$NTPP(A,B) \vee TPP(A,B)$
$(holding\ hand\ A)$	$\neg(DC\ hand\ A)$
$(empty\ hand)$	$\forall obj, (DC\ A\ obj)$
$(on\ A\ B)$	$EC(A,B) \wedge (in\ A\ region_{on}(B))$ where $region_{on}(B))$ is shown in Figure 6(b)

DBN Estimator The DBN estimator performs the online filtering estimation over the DBN model constructed in the offline compilation. The estimator uses the online observed predicates (at each time step), the online output from the observed predicate generator, as the observations for the online DBN filtering. The estimator outputs the probability distributions over activity stages and predicates (at each time step). To be more specific, the probability distributions of the hidden predicate variables and the activity variables in the DBN are the outputs of the DBN estimator. Many DBN filtering algorithms can be applied. We applied particle filter for the experimental results for the computational purposes.

Experimental Results

We performed an experiment using the PDDL in Table 1. Figure 9 shows the experiment environment. For a practical purpose, we used three blocks instead of three tools ($block_{red}$ for the *hammer*, $block_{blue}$ for the *drill*, and $block_{green}$ for the *nails*). We used two locations ($toolbox_A$ and $toolbox_B$), and two manipulators (a human hand and a

robot manipulator). We estimated over 13 grounded predicates (6 for on, 3 for clear, 1 for empty, and 3 for holding) and 15 grounded activities (6 for pick, 6 for pick, and 3 for passing).

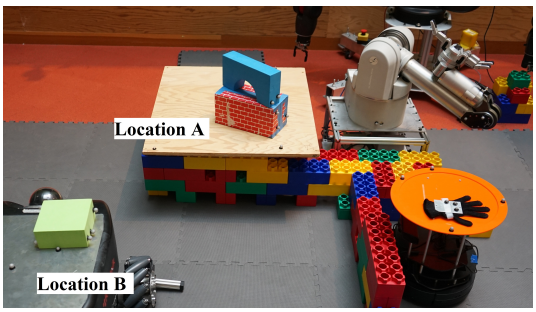


Figure 9: Experimental environment

We used a Vicon system to measure the position (global x, y, and z coordinates for the center of mass) and the orientation (helical x, y, and z coordinates) of each object online, while a person performed a series of actions. As the Vicon system is very accurate, we included random white Gaussian noise. For position, we added zero mean noise with the covariance of $400 \times I_{3\times3}$ (mm^2). For orientation, the covariance was $0.01 \times I_{3\times3}$ (rad^2). Here, $I_{3\times3}$ is the 3×3 identity matrix. Note that the length of the blocks' sides ranged from 70 mm to 200 mm, so the added noise was relatively significant. Then, LCARS estimated over the human activities and predicates using the online noisy pose observations. The pose measurement was performed for about 100 seconds, receiving 9,848 sequential observations.

At each time step, we calculated the most likely states for all the activities and predicates (hidden predicates) online. If we got all correctly compared to the ground truth, we assumed we got correct recognition at that time step. Ground truth was collected separately while collecting the noisy pose observations. We calculated the accuracy rate of our correct recognition out of the total number of time steps. For instance, if we got correct recognition for 8,000 time steps out of the total of 9,848 time steps, the accuracy rate would be 8,000/9,848$\times$100 $\approx$ 81.23%. When we applied LCARS, the accuracy rate was 83.37%. To be more specific, it was 92.28% for correctly recognizing all the predicates only and 85.83% for correctly recognizing all the activity stages only. The results are summarized in Table 3.

Table 3: LCARS Accuracy Rates

Case	LCARS accuracy rate	HMM accuracy rate
All predicates only	92.28%	62.37%
All activity stages only	85.83%	63.33%
All predicates and activity stages	83.37%	55.92%

To illustrate the robustness of the LCARS' DBN estimator, we also calculated the predicate accuracy rate using the observed predicates as the estimate. The observed predicates are the outputs from the observed predicate generator and also the observations for the DBN filtering, as it was illustrated above. The observation predicates are full of noise. On the contrary, the hidden predicates that the LCARS' DBN estimator outputs are much more accurate. By comparing the accuracy rates with the observed predicates and the hidden predicates, we can show the robustness of our method. With the observed predicates, the accuracy rate of correctly recognizing all the predicates was 21.66%. From Table 3, it was 92.28% with the hidden predicates. The comparison is summarized in Table 4. Table 4 also compares the accuracy rate of correctly recognizing each individual predicate.

Table 4: Comparison of Accuracy Rates for Predicates

Predicates	The observed predicates	The hidden predicates (LCARS)	HMM accuracy rate
$(empty\ hand)$	76.45%	98.20%	92.52%
$(in\ block_{red}\ toolbox_A)$	51.26%	95.83%	90.65%
$(holding\ block_{red}\ hand)$	69.48%	97.75%	89.17%
$(clear\ block_{red})$	71.36%	97.26%	92.40%
All predicates altogether	21.66%	92.28%	62.37%

We also compared LCARS with an HMM-based method. We designed an independent HMM for each grounded predicate to get the HMM estimates of all the grounded predicates and actions, using the observed predicates as the online observations. The results for the HMMs are shown in Table 3 and Table 4. We can see that the accuracy rate from LCARS is higher. This means that LCARS captures the complex interactions between predicates and actions well, compared to simple HMM-based methods.

Conclusions

This paper presents LCARS algorithm for robust online human activity recognition. Our solution combines three elements: i) smart mapping from low-level pose data to abstract PDDL predicates using QSR, ii) efficient modeling of abstract human behavior using PDDL, and iii) a robust probabilistic model using DBN. LCARS has two components: i) the offline compilation component and ii) the online estimation component. The offline part automatically generates the online DBN part using a PDDL human behavior model. The online part has two subcomponents: i) the observed predicate generator and ii) the DBN estimator. The observed predicate generator takes in the user-specified PDDL predicate definition in QSR primitives and online noisy pose observation as inputs and outputs the observed predicates online. The observed predicates are used in the DBN estimator's online filtering to find the estimate over the human activity stages and predicates.

Future efforts will focus on applying LCARS to various environments with noisy sensors. For example, cameras with neural network-based object detection algorithms are widely used today. In such cases, an interesting scenario is when the online pose observations of some objects are blocked for some amount of time. We expect LCARS to be robust to such cases as well.

References

Bui, H. H. 2003. A General Model for Online Probabilistic Plan Recognition. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, volume 3, 1309–1315. Citeseer.

Cohn, A. G.; Bennett, B.; Gooday, J.; and Gotts, N. M. 1997. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *GeoInformatica* 1(3):275–316.

Ericson, C. 2004. *Real-Time Collision Detection*. CRC press.

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

Freksa, C. 1991. Qualitative Spatial Reasoning. *Cognitive and Linguistic Aspects of Geographic Space* 63:361–372.

Heinze, C. 2004. Modelling Intention Recognition for Intelligent Agent Systems. Technical Report DSTO-RR-0286, Defence Science and Technology Organisation Salisbury (Australia) Systems Sciences Lab.

Kirste, T. 2011. Making Use of Intentions. *Informatik Preprint CS-01-11, Institut für Informatik, Universität Rostock*.

Lee, S. U.; Jasour, A.; Hofmann, A.; and Williams, B. 2018. 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)*.

Murphy, K. P. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Phd dissertation, University of California, Berkeley.

Pynadath, D. V., and Wellman, M. P. 2000. Probabilistic State-Dependent Grammars for Plan Recognition. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 507–514. Morgan Kaufmann Publishers Inc.

Schlenoff, C.; Pietromartire, A.; Kootbally, Z.; Balakirsky, S.; and Foufou, S. 2013. Ontology-based State Representations for Intention Recognition in Human–robot Collaborative Environments. *Robotics and Autonomous Systems* 61(11):1224–1234.

Schlenoff, C.; Kootbally, Z.; Pietromartire, A.; Franaszek, M.; and Foufou, S. 2015. Intent Recognition in Manufacturing Applications. *Robotics and Computer-Integrated Manufacturing* 33:29–41.

Vieu, M. A. 1993. A Three-level Approach to the Semantics of Space. *The Semantics of Prepositions: from Mental Processing to Natural Language Processing 3* 393–440.

Wang, D., and Williams, B. C. 2015. tBurton: A Divide and Conquer Temporal Planner. In *In Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 3409–3417.

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

Yordanova, K. 2011. Modelling Human Behaviour Using Partial Order Planning Based on Atomic Action Templates. In *Intelligent Environments (IE), 2011 7th International Conference on*, 338–341. IEEE.