Towards Supervised Acquisition of Robot Activity Experiences: an Ontology-based Approach

Aneesh Chauhan¹, Luís Seabra Lopes¹,², Ana Maria Tomé¹,², and Armando J. Pinho¹,²

IEETA¹/DETI², Universidade de Aveiro,
Aveiro, 3810-193, Portugal
{aneesh.chauhan,lsl}@ua.pt

Abstract. This paper describes the capabilities developed, in the EU project RACE, for recording experiences of plan-based robot activities. Recorded experiences are episodic descriptions of the robot’s activities including relevant perceptions of the environment, the goals pursued, successes and failures. The paper focuses on supervised experience gathering, where a user guides the robot to perform tasks. Interaction is supported by a user interface which is tightly integrated with a central repository of information shared among foreground processes (perception, planning, interaction, execution management). A teaching action from the user triggers recording of an experience for long-term storage. The experience gathering process and the content of the recorded experiences are illustrated in a scenario where a PR2 robot is taught how to “serve a coffee” to a guest. The described capabilities are completely integrated in the implemented robot architecture.

Keywords: experience extraction; human-robot interaction; knowledge representation

1 Introduction

Service robots, designed to work autonomously in normal human environments and in close collaboration with humans, will have to deal with ever-changing scenarios. Such robots will have to be robust against changes and sufficiently versatile to adapt to novel scenarios. The robustness and versatility can be greatly enhanced if the robot is capable of learning from its past experiences and applying the learned knowledge to modify its current behavior to suit new situations.

This is the main direction being taken by RACE (Robustness by Autonomous Competence Enhancement), a project that aims to develop an integrated framework and methods for robots that learn from their experiences. Experiences are records of past happenings stored by a robot, as observed through its sensors, and interpreted according to the robot’s conceptual framework, in a coherent subset of space-time.
Towards Supervised Acquisition of Robot Activity Experiences

Gathering appropriate experiences, therefore, is one of the key tasks outlined in this project. The focus of this paper is on gathering experiences of plan-based robot activities. The project uses a PR2 robot, from Willow Garage, for experimental developments. The robot uses an HTN planner \cite{7} to generate plans to achieve given tasks.

Several projects have explored different approaches to experience gathering and subsequent conceptualization. Both, unsupervised \cite{3,4,6,12} and supervised approaches have been explored \cite{1,2,8,9}. Although unsupervised experience gathering is a key ingredient in the development of artificial cognitive agents, interaction with other agents also plays an important part. Interaction helps to focus learning on the concepts required for the tasks at hand.

Nicolescu and Mataric \cite{8} address two independent but related tasks, namely robot learning from demonstration and failure handling through human-robot interaction. A task representation is derived from each demonstration, based on correspondences between execution data and behavior models. When failing to execute a known task, the robot, behaving as a pre-linguistic child, will search for a human and induce the human to follow the robot to the location where the failure happened.

Kirsch \cite{2} presents a semi-supervised approach to experience-gathering. An experience is represented as a trace through a hybrid automaton observed during an episode. Episodes are specified, by the programmer, as LISP classes and the observed data is treated as an instance of such a class. Experiences can be raw or abstract. Raw experiences contain all data collected during execution. Each raw experience can be used for several abstract experiences, applying different abstraction steps needed for learning. The programmer defines the features to include in abstract experiences, and also defines which data to record and when it should be recorded.

Pardowitz et al. \cite{9} combine learning by demonstration with vocal comments. The robot starts with a formal definition of the structure of a task, which may allow for multiple orderings of the primitive operations. Then, at each human demonstration of the task, the robot will acquire knowledge about the best sequences. This involves segmenting the demonstration into basic actions and mapping the segmented sequence to the given task model. This knowledge is incrementally incorporated in the task model. Relevance of task features (such as manipulated objects or action pre-/post-conditions) is estimated based on vocal comments provided by the user during demonstrations.

Connell et al. \cite{1} present ELI (Extensible Language Interface), which allows a user to command a robot to perform fetch and carry tasks, as well as to teach the names of objects and actions (i.e. nouns and verbs). The action words are grounded using stepwise instructions, where each instruction corresponds to a known primitive action. A new action word is then associated to a sequence of basic actions (arm movements).

Although RACE is exploring unsupervised and semi-supervised approaches, this paper describes the research approach and the infrastructure designed for supervised experience gathering of plan-based robot activities. A command-line
interface was developed to support human-robot interaction. This interface allows the human user to instruct the robot to perform tasks as well as teach new task concepts. The teaching action from the user triggers the module that gathers all the experiences relevant for conceptualizing the new task. These experience records will eventually be taken as input for learning and conceptualization. Experience gathering and conceptualization are background processes, since they do not directly influence the course of events. They are also meta-level processes, in the sense that they change the background knowledge used by foreground processes to make decisions.

The rest of the paper is structured as follows: the following section describes the RACE architecture, with emphasis on the modules relevant for experience gathering; Section 3 presents the ontological concepts related to experiences and user instructions; Section 4 details the command-line interface for human-robot interaction; Section 5 describes the experience extraction and recording process, followed by an example scenario, where the robot is taught how to “serve a coffee” and the relevant experience is extracted; and the final section presents some conclusions.

2 RACE implementation architecture

Rockel et al. [11] describe the RACE architecture (Figure 1) and the ontology-based knowledge representation framework. From the implementation point of view, each module is typically a ROS [10] package. Processes associated with each module run as ROS nodes on the PR2 robot. The conceptual knowledge of the robot – including physical entities, planning domain, spatial and temporal constraints, experience types, instruction types etc. – are defined in an OWL2 ontology.

All modules in the RACE architecture communicate through a central repository called Blackboard. Implemented as an RDF database, the Blackboard contents are similar to what can be found in an ABox in Description Logics. Different modules can read selected types of information from the Blackboard and write their outputs to the Blackboard. The unit of data written to the Blackboard is the so called fluent, which describes an instance of some concept in the ontology. Fluents are stamped with start and finish times. The state of the world at a given time can be inferred from currently active fluents. The Blackboard is also used to store experience records (e.g., execution of a coffee serving task), created by the Experience Extractor/Annotator. Thus, in the current state of development, the RACE Blackboard combines two memory system types well known in cognitive science, namely Working Memory and Episodic Memory [13,14]. Given that the occurrence history (working memory) and the extracted experiences (episodic memory) share the same repository, recording experiences can take, to some extent, the form of annotation of occurrences.

Background processes responsible for experience extraction and conceptualization support a long-term learning loop. The history of occurrences in the Blackboard is continuously observed by the Experience Extractor to detect and
Towards Supervised Acquisition of Robot Activity Experiences

Fig. 1. The current RACE Modules and Interfaces Architecture

extract potentially relevant experiences, based on plan structure, user feedback, pauses, similarity with stored experiences and conceptualizations, and novelty.

Experiences will then be used to improve the future performance by employing several learning methods (a work in progress). Taking the experiences from the Blackboard as input, the Conceptualizer will generate and store new concepts and/or update existing concepts, resulting in more robust and flexible future plans. Experiences and concepts are part of what can be called Long-Term Memory. Conceptualization is outside the scope of the paper and will not be addressed here.

3 Ontological Aspects

Many concepts relevant for planning and execution of robot tasks are covered in the RACE OWL2 ontology [11]. To support experience gathering, concepts to represent experiences and user instructions were also included in the ontology. This section elaborates on these ontological aspects.

3.1 Ontology of Experiences

Experiences are the main source of information used for learning how to achieve a more robust robot behavior. Thus, in this project, “experiences” play a role
similar to “cases” in Case-Based Reasoning, “episodes” in Cognitive Science or “instances” in Instance-Based Learning. They arise from robot observations and activities. The experience gathering capabilities are intended to provide direct support for conceptualization and learning capabilities.

We introduce a basic distinction between spatial and dynamic experiences. Spatial experiences can be related to the evolutionary roots of human semantic memory \[13,14\]. They are relevant for conceptualizing the spatial and relational structure of passive physical objects and scenes, in particular the shape and appearance of objects and the layout of scenes. Although spatial experiences may be extracted from dynamic scenes (or through active observation of scenes), they don’t include the time dimension. By contrast, dynamic experiences contain data that describes activities as sets of occurrences, i.e. temporally extended relations or properties.

The type of a recorded experience is determined by the extracted experience data, and not by the original occurrence data. For instance, the activity of creating a scene (e.g. preparing a table for dinner) can be used to conceptualize the activity itself (thus a DynamicExperience is extracted and stored) or the produced scene structure (in this case a spatial experience is stored), or both. Keeping in mind the scope of the paper, the ontological concepts related to robot activity experiences are described in Listing 1.1.

```
Class: Experience
SubClassOf: Thing
  and hasCategoryLabel some String
  and hasLabelSource exactly 1 LabelSource
  and hasExperienceStartTimeLowerBound some time
  and hasExperienceStartTimeUpperBound some time
  and hasExperienceEndTimeLowerBound some time
  and hasExperienceEndTimeUpperBound some time

Class: DynamicExperience
SubClassOf: Experience
  and hasSceneObject some SceneObject

Class: RobotActivityExperience
SubClassOf: DynamicExperience
  and hasPlanObject some PlanObject
  and hasEndStatus some EndStatus
```

Listing 1.1. Ontology of robot activity experiences in OWL2 Manchester Syntax

The following is a textual definition of the different types of experiences:

**Experience** [Thing] This concept represents an experience record containing data describing an instance of a category to be learned and possibly the name of the category as provided by an instructor or an unsupervised internal process (LabelSource). The description of the instance is extracted from the occurrence history in a coherent subset of space-time.

**DynamicExperience** [Experience] An experience relevant for conceptualizing activities in terms of sets of occurrences in a coherent subset of space-time.

**RobotActivityExperience** [DynamicExperience] A dynamic experience containing data about an activity of the robot, extracted from the occurrence history recorded during the execution of that activity. The experience data
Towards Supervised Acquisition of Robot Activity Experiences

includes the underlying goals, executed plan, a set of occurrences and success information. Robot activity experiences are relevant for conceptualizing a given category of robot activities.

3.2 Ontology of User Instructions

In RACE, human-robot interaction is essentially used for supervised experience gathering. It will be used in situations where the robot fails to deal with exceptional situations autonomously as well as for teaching knowledge relevant for new tasks. The project is developing and integrating basic mechanisms for verbal and gestural instructions. The main concern is to identify the range of teaching actions and then develop simplified implementations for each of them, allowing us to explore how learning can benefit from interaction. In Listing 1.2, we present basic ontological concepts which allow to internally represent user instructions.

```owl
Class: ElementaryInstructorActivity
SubClassOf: InstructorActivity
and hasInstructor exactly 1 Instructor

Class: InstructorGesture
SubClassOf: ElementaryInstructorActivity

Class: InstructorPoint
SubClassOf: InstructorGesture
and hasReferent exactly 1 PhysicalEntity

Class: InstructorDialogMove
SubClassOf: ElementaryInstructorActivity
and hasSentence some String

Class: InstructorTell
SubClassOf: InstructorDialogMove
and hasConstrainedState exactly 1 ConstrainedState
and hasRelationSubject max 1 (SceneObject or PlanObject)
and hasRelationObject max 1 SceneObject

Class: InstructorTeach
SubClassOf: InstructorTell
and hasRelationSubjectCategoryName some String

Class: InstructorAchieve
SubClassOf: InstructorTell
and hasTask exactly 1 Task

Class: InstructorAsk
SubClassOf: InstructorDialogMove
and hasTargetType max 1 TeachTargetType

Class: InstructorCorrect
SubClassOf: InstructorTell
and hasInstructorAsk exactly 1 InstructorAsk
and hasRobotReply some String
and hasCorrectLabel some String
```

Listing 1.2. Ontology of user instructions in OWL2 Manchester Syntax

An ElementaryInstructorActivity always involves an Instructor. We distinguish between two main types of instructor activities, namely gestures (InstructorGesture), particularly pointing, and verbal dialog moves
(InstructorDialogMove). Among the later, four types are considered: InstructorTell is a dialog move of general-purpose declarative nature. A special case is InstructorTeach which is useful for providing informations in terms of assertions about the scene, for example “This is a mug” or “You just served coffee to guest1”. This dialog move is relevant for identifying instances of concepts to be learned. The InstructorAchieve dialog move is of imperative nature, i.e. it can be used to request the robot to achieve some goal or to perform some activity. Dialog moves of interrogative nature (also useful in teaching situations) are represented by the InstructorAsk. The dialog move InstructorCorrect, another special case of InstructorTell, also relevant for teaching, can be used to correct inferences previously made by robot.

4 User Interface

An interface is required to allow the user to provide instructions to the robot, thus facilitating experience gathering. Several efforts took place in this direction. In an early stage, a textual natural language interface, developed for interactive teaching of object categories, was integrated with a simulated PR2 in a mixed-reality environment. After the RACE architecture was defined, a command-line interface was developed to support communication between the robot and human user. The command-line interface is an intermediate approach currently being used in the project, but it will be replaced by a more friendly and flexible Android-based interface in the near future. The new interface will reuse a significant part of the command-line interface functionalities, on which the current section focuses.

The current interface allows the user to provide instructions for the successive steps in a task, such as moving to places, and picking and placing objects. The user can also teach the name (CategoryLabel) of the performed activity. More specifically, the interface currently supports two types of instructions:

- **achieve**: this instruction corresponds to the InstructorAchieve concept and takes as input a task to be completed by the robot:
  
  Format: achieve <task name> <task arguments>

- **teach_task**: this instruction corresponds to the TeachActivityCategory concept, which is a sub-concept of InstructorTeach, and takes as input the name of the task that the user wishes to teach and the respective parameters:
  
  Format: teach_task <task name> <task arguments>

This interface is tightly integrated with the Blackboard. The set of tasks that an achieve instruction can take as input (tasks that the robot can carry out) is queried from the blackboard. These tasks are already included as leaf concepts in the Task ontology (see Figure 2).

Each task takes zero or more arguments for instantiation. For example, as shown in Figure 2, the grasp_object_w_arm_Task takes an instance of PassiveObject as the first argument and an instance of Arm as the second argument. For every task, the argument types and possible values are also queried
Towards Supervised Acquisition of Robot Activity Experiences

Fig. 2. Fragment of the ontology of tasks defined for the project. The leaf concepts are the possible inputs to the achieve instruction.

from the Blackboard. This information is exploited to enhance the usability of the interface in two ways:

– Tab completion: While an instruction is being typed by the user, the user interface tracks the current state of the typed input as well as the knowledge of the set of next possible inputs. E.g. achieve is always followed by a <task name>, which in turn is followed by its first argument, and so on. Tab completion facilitates user interaction by displaying all possible choices for the next term to be typed, as well as automatically fills in partially complete input.

– Instruction validation: An incorrect input from the user, e.g. an incorrect/unknown task name and/or wrong arguments, can be easily validated. In this case, a warning is displayed, highlighting the incorrect terms in the instruction.

These features make the interface more user-friendly and less prone to erroneous input.

For the teach_task instruction, a new or an existing <task name> can be given as input. The arguments to this task are instances (fluents) already present in the Blackboard.

The interface was also designed keeping in mind that new concepts and instances of existing concepts will be added to the Blackboard. After each user instruction, the interface queries the blackboard for the leaf concepts of Task as well as the types and instances of the corresponding arguments. This enables to maintain an up-to-date set of possible tasks and corresponding arguments.

Once the user provides the instruction, the user interface module writes one or more fluents to the Blackboard. For example, to instruct the robot to move
to a location near the counter (more specifically, the pre-manipulation area east of the counter, \texttt{pmae3}), the user will enter the following instruction:

```
achieve drive_robot_Task pmae3
```

This will lead to the creation of two fluents by the interface (shown in Listing 1.3). The first fluent represents the instruction. The contents of this fluent can be summarized as follows: a new instance of \texttt{InstructorAchieve}, named \texttt{instructorAchieve1}, was created; this instruction contains an instance of \texttt{Task}, named \texttt{drive_robot_Task_ui}; the Fluent was created at Unix time-point 1363704909.988. The second fluent contains the instance of the \texttt{drive_robot_Task} (a leaf concept of \texttt{Task}) to be completed. This task takes one argument, \texttt{pmae3}, as input, which is an instance of the \texttt{Area} concept.

```
Class_Instance: [InstructorAchieve, instructorAchieve1]
StartTime: [1363704909.988, 1363704909.988]
FinishTime: [1363704909.988, 1363704909.988]
Properties: 
- [hasTask, Task, drive_robot_Task_ui]
```

```
Class_Instance: [drive_robot_Task, drive_robot_Task_ui]
StartTime: [1363704909.988, 1363704909.988]
FinishTime: [1363704909.988, 1363704909.988]
Properties: 
- [hasFirstArgument, Area, pmae3]
```

Listing 1.3. Fluents corresponding to an achieve instruction in YAML format

5 From Instructions to Robot Activity Experiences

During execution, different foreground modules (proprio/perception, planner, execution manager, user-interface etc.) continuously add and update fluents to the Blackboard. Robot activity experiences are extracted from Blackboard contents in a time interval in which some interesting robot activity took place. In the case of supervised gathering of experiences, the teach instructions are taken as cues for identifying this time interval. The experience extraction is triggered when the user interface adds an instance of \texttt{TeachActivityCategory} (a \texttt{teach_task} instruction in the command-line interface).

The heuristic for segmenting the history of occurrences to extract a robot activity experience is the following:

- The experience starts either at the first \texttt{achieve} instruction after the previous \texttt{teach} instruction (if it exists) or at the beginning of the session. The \texttt{StartTime} of this \texttt{achieve} instruction sets the lower-bound on the time interval from which the experience will be extracted;
- The experience ends with the current \texttt{teach} instruction. The \texttt{FinishTime} of this instruction sets the upper bound on the time interval from which the experience will be extracted.

All the fluents active within this interval become part of the experience record pertaining to the new activity taught by the user.
Towards Supervised Acquisition of Robot Activity Experiences

5.1 An Example

Consider the problem of teaching \textit{trixi}, the PR2 robot used in our project, how to serve a coffee to a guest in a restaurant. The restaurant floor plan is as shown in Figure 3, where the robot knows the position of \texttt{mug1} on the counter, the position of \texttt{table1}, the position of \texttt{guest1} west of \texttt{table1}, and the regions for manipulation, sitting and placing [11]. The user successively instructs the robot to move to \texttt{counter1}, grasp \texttt{mug1}, move to the manipulation region south of \texttt{table1} (\texttt{mas1}), and place \texttt{mug1} at the western placement region of \texttt{table1} (\texttt{pawr1}). At the end, the robot is told that a \texttt{ServeACoffee} activity was carried out to \texttt{guest1}:

\begin{verbatim}
achieve drive_robot_Task pmae3
achieve grasp_object_w_arm_Task mug1 rightArm1
achieve drive_robot_Task pmas1
achieve put_object_Task mug1 pawr1

teach_task ServeACoffee guest1
\end{verbatim}

Listing 1.4. Step-by-step instructions for teaching ServeACoffee

The \texttt{teach_task} instruction triggers the \textit{Experience Extractor} module to extract and record the experience. All fluents active within the time interval between the first \texttt{achieve} and the \texttt{teach_task} instructions, are recorded. In the current implementation, this record is the robot activity experience. Figure 4 presents a summary of the concepts for which instances could be found in the Blackboard at the end of the \texttt{ServeACoffee} task, along with the number of instances found. A total of 345 concept instances were found, the most common ones being poses, bounding boxes, arm postures and torso postures.

As mentioned earlier, currently an experience record contains all active fluents. These experience records are already being exploited for conceptualization. However, most of the stored fluents are not necessary for learning a new activity. For example, fluents related to the robot’s posture, many bounding boxes and the poses related to objects in the environment, may be not relevant to learn-
Towards Supervised Acquisition of Robot Activity Experiences

6 Conclusions

This paper describes the infrastructure and the capabilities developed, in the RACE project, for supervised acquisition of plan-based robot activity experiences. Gathering experiences is seen as a background process, since it does not directly influence the course of events. The experience gathering process is also a meta-level process, since it observes the trace of foreground cognitive activity and changes the background knowledge to be used by foreground processes to make future decisions. The ontological concepts describing experiences and user instructions were presented.

A command-line interface, which allows the user to instruct the robot to perform tasks, as well as teach new activities, was developed. This interface is tightly integrated with the Blackboard and supports instruction validation and tab completion. Using teaching instructions as cues and a history segmentation heuristic, the Experience Extractor finds the time interval in which the new activity has occurred. All the active fluents in this interval are identified and associated to a new experience in the Blackboard. Currently, all the active fluents within the experience interval are stored. In future work, filtering and abstraction mechanisms will be developed to identify the relevant fluents.
Acknowledgments. This work is funded by the EC Seventh Framework Program, under the grant agreement no. 287752, RACE (http://project-race.eu). The authors thank Tony Cohn, Sandeep Dubba, Martin Guenther, Stephie von Riegen, Lothar Hotz and Bernd Neumann for fruitful discussions.

References