Knowledge Representation for Cognition- and Learning-enabled Robot Manipulation

Daniel Beßler, Sebastian Koralewski, Michael Beetz *
Institute for Artificial Intelligence
Am Fallturm 1, 28359 Bremen, Germany

Abstract

Knowledge representation and reasoning (KR&R) systems are widely employed for the representation of abstract knowledge. Action models are usually representations of state transitions: Actions can be performed if all pre-conditions are met, and it is expected that the designated effects will take place when the action is executed. However, embodied agents need additional knowledge about how their body should be moved to achieve their goals without causing unwanted side effects. The proposed action representation is based on force dynamic events that occur when an embodied agent interacts with its world. We show how patterns of force events can be used to define semantics of action verbs. Robots use our model to acquire episodic memories which are stories of their performance coupled with sub-symbolic data, and they share their experience through the knowledge service openEASE.

Introduction

The cognition system of humans allows us to accomplish manipulation tasks very competently. This is possible through the organization of actions in terms of motion phases, and through the prediction of effects

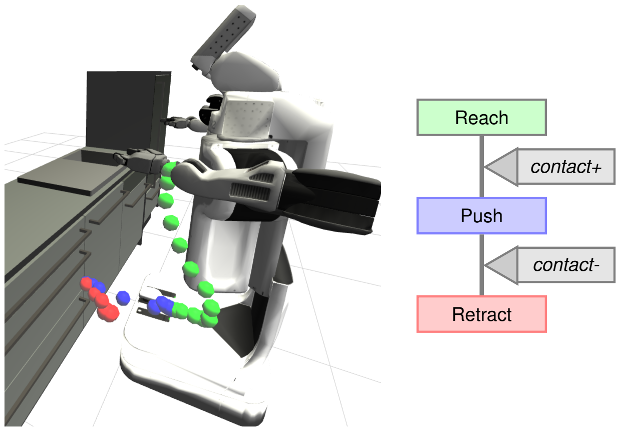

Figure 1: PR2 closing a drawer in a kitchen. The action is decomposed into different phases with distinct motion and force event pattern.

that actions might cause in terms of force events that might occur.

In this work, we investigate an action model postulated in human psychology, and make use of it in an artificial system. The model was proposed by Flannagan et *al.* [6]. Actions are decomposed into motion phases with different subgoals. The subgoals are force dynamic events that also generate distinctive sensory feedback in the nervous system.

Intentions of others can not be monitored directly. Monitoring force events, on the other hand, is at least less problematic because events may be monitored in the physics engine of virtual worlds, or observed by some agent. This is, for example, that the hand gets into contact with the milk package before grasping it from the table, or that the package looses contact to the supporting surface when the agent performs a retracting motion after the milk has been grasped.

One of the main reasons for investigating action models from human psychology in robotics is that action models in AI, such as PDDL [7], usually do not have an appropriate level of abstraction for robots. In

*The research reported in this paper has been supported by the German Research Foundation DFG, as part of Collaborative Research Center (Sonderforschungsbereich) 1320 "EASE - Everyday Activity Science and Engineering", University of Bremen (http://www.ease-crc.org/).

Copyright © by the paper's authors. Copying permitted for private and academic purposes.

In: G. Steinbauer, A. Ferrein (eds.): Proceedings of the 11th International Workshop on Cognitive Robotics, Tempe, AZ, USA, 27-Oct-2018, published at http://ceur-ws.org

CEUR-WS.org/Vol-2325/paper-04.pdf

particular, action models in AI often abstract away from body motions and only concentrate on representing action pre- and post-conditions, and sequences. Intelligent embodied agents need to bridge the gap between these representations with missing information and the actual execution of an action in the physical world. Bridging this gap is non trivial and a problem which is widely unsolved on the abstract level (i.e., by re-usable general knowledge). It is further expected that conditions and effects of actions are pre-defined – a hard to meet requirement with the diversity of effects actions may cause in the physical world.

The central question for successful embodied action execution is how agents should move their bodies to achieve certain effects while avoiding unwanted side-effects. This is, for example, how a robot should move its arm such that the pancake mix contained in the bottle it holds is poured on top of the pancake maker, and forms a pancake with 10cm diameter. In the area of AI there are only few approaches that address this problem despite the semantic nature of this reasoning problem.

One of the peculiarities of our KR&R system is that it runs inside the perception-action loop of a robotic agent. Symbols correspond to data structures of the robot control system, and as such they have a rather simple grounding. The representations in our system are inspired by the role that episodic memories play in the acquisition of generalized knowledge in the human memory system [18].

The proposed representation of episodic memories consists of two parts. One part stores experiences and events as symbolic data. Those events and experiences can be e.g. perceived objects, or performed actions, their duration, and possible failures. The second part stores sensor data from the robot in a database. We define this unstructured data as sub-symbolic data. In the first section, we will describe the symbolic knowledge representation. An overview about the sub-symbolic data will be given afterwards. Then, we will show how those memories can be used to improve the robot's action models by getting insights about manipulation activities. This will be achieved by using a combination of query answering and visual analytic tools.

Our KR&R system is made available as part of the knowledge web service openEASE [1] [4]. The web service gives the KR community the opportunity to do research in the context of real robot experiments. Researchers in the field of KR-based robot control can further extend the knowledge base of the web service by providing additional episodic memories of their robots performing manipulation activities.

[1]`http://www.open-ease.org/`

We use the openEASE platform for storing and managing the episodic memories represented with our model. It also allows to ask queries about it, such as how the robot was moving when an action was performed, and to visualize snapshots of the activity with visual annotations. Figure 1 shows such an example where the robot was closing a drawer in a kitchen environment. The action is properly segmented into the different motion phases, which is also visible in the Figure. The vision is to collect a large data set of episodic memories, and utilize them for learning tasks to gain a better understanding about manipulation activities.

Related Work

There are several projects with efforts to provide symbolic knowledge about manipulation activities to robots. The most notable one is the IEEE-RAS working group ORA (Ontologies for Robotics and Automation) [13], which aims at defining standards for knowledge representation in robotics. Schlenoff [12] also presented a related approach for detecting intentions in cooperative human-robot environments based on states which are more easily recognizable by sensor systems than actions. In his work, intentions are also used for the prediction of the next action. For this work, we extend the KnowRob system [17], which, among others, defines concepts for actions, and their effects [16]. KnowRob also has a notion of motion phases, but these are not defined using force dynamics.

Another related branch of research is task and motion planning. In this work, we present an action model that can be used to yield higher level activities from observations of force events. Such force events can also be detected through haptic feedback, and be used to minimize uncertainty during manipulation activity planning [19]. The relation of our system to general planning systems is that planning domains can be represented using our model and that plan parameters can be inferred from knowledge represented in our system. Action models in traditional planning systems (such as PDDL) often only consider action preconditions and their effects, and do not incorporate more detailed information about motions and forces. More recently, systems emerged that enable robots to perform planning on both task and motion level by introducing an interface layer between task and motion planner [14, 5]. Our action model could be used by such systems to represent tasks, and to define action pre-conditions which are occurrences of force events.

Another aspect is that our system can yield partial boundaries of motion phases given some observation. Motion segmentation methods typically apply some form of clustering to build stochastic representations of primitive motions and motion sequences. These methods include self-similarity [8], k-means [10] or hierarchical [20] clustering. Primitive motions are often represented as Hidden Markov Models [9, 11, 15] and sequences as stochastic motion graphs [15]. This research has mainly focused on body motions with some exceptions that also consider object movement [9, 11]. Contrary to our approach, the listed motion segmentation approaches do either not consider manipulated object movement or only consider its trajectory. Instead, we define motion boundaries according to interactions of objects with the physical world through force dynamics. These contact states seem particularly important for control strategies employed by humans [6].

Narrative of Episodic Memories

This section introduces an action model for robots inspired by the Flanagan model. The basis of it are force events that occur when an agent moves its body, and the different motion phases of actions. Our ontology is organized along these areas. It has 4 levels: Force events, situations, motion phases, and intentional activities. In addition, we use rules to declare identity constraints. In this section we provide a description of how this information is organized and represented.

In this work, we build upon the KnowRob ontology, and (manually) extend it with concepts of our action model such as *ForceEvent* and *PouringMotion*. We have chosen KnowRob because it provides the necessary infrastructure for interfacing with robot control systems, and to record episodic memories from task execution. It defines concepts such as *Event* and *Situation*, and also specific ones to describe e.g. robots and their parts.

Force Events

At the lowest level of our action representation there are events that physical objects cause in a (simulated) physical world. They are described independently from intentions. This is to allow detecting them fully automated, without taking into account previous events and higher level knowledge about task or embodiment.

PhysicalEvent $\sqsubseteq$ *Event* is the most general concept in this ontology. It implies that physical events occur at a particular time instant (derived from *Event*), and that at least one object is *involved.* Involved means that one of their physical properties is salient during the event. This is the case if the object involved is created or destroyed, touched or untouched, transformed into something else, etc.

The most essential events are the contact events (*ContactEvent*) that occur whenever an object moves in the world such that it touches (*contact+* $\sqsubseteq$ *involved*) another object within a spatial region (*contactRegion*). The property *contact+* is further decomposed into functional properties *contact+*$_1$ and *contact+*$_2$ denoting the two salient objects during the contact event (the two objects can be randomly assigned). The objects remain touched until they separate again which is indicated by a *LeavingContactEvent.* The contact is either caused by an agent moving objects into contact, or through a physical process such as gravity, for example, pulling an object such that it falls onto the floor.

Creation (*CreationEvent*) and destruction (*DestructionEvent*) events are also distinctive subgoals of activities that we use for activity representation at a higher level of our ontology (e.g., cutting a bread creates a slice of bread).

The last category of physical events we consider in the scope of this work are fluid flow events (*FluidFlowEvent*). These are events in which some liquid or gaseous substance moves, for example, milk flowing from a package to a glass, or water flowing in a river. Such events may be intended as in “pouring milk in a glass”, or unintended as in “spilling milk on the floor during navigating”. The primary involved object is the liquid or gaseous substance, linked to the event via the functional property *fluid* $\sqsubseteq$ *involved.*

Force Situations

At the next level of our ontology there are situations during which force events occurred (*ForceSituation* $\sqsubseteq$ *Situation*). Force events occur at time instants, for example, in the moment the hand touches some object, and when it leaves contact again. We use such temporal patterns of force events to expand them to distinctive situations.

Sub-events are linked to situations via the inverse functional *event* object property. With inverse functional we imply that each event can only be the sub-event of a single situation. For detecting situations, we use two dedicated events: One indicating the start and the other indicating the end of the situation. These are represented using the functional properties *starter* $\sqsubseteq$ *event* for the event starting the situation, and *stopper* $\sqsubseteq$ *event* for the one stopping it.

Surely, starter event should occur before stopper event. Situations during which the object is not in contact could else be classified as contact situations. We use predicates from Allen's interval algebra [1] and an identity constraint to assert this relation between starter and stopper event. As illustration, this constraint can be written as:

$$\begin{array}{c}\forall\ \mathit{instance\_of}(x, \mathit{ForceSituation}) :\\ \exists(\mathit{stopper} \circ \mathit{after} \circ \mathit{starter}^{-})(x, x)\end{array} \tag{1}$$

Note that the fact that some event occurred *after* another one is inferred on demand by our reasoner and does not need to be asserted. The begin time of the situation is further defined as time of occurrence of the starter, and the end time as the time of occurrence of the stopper.

The starter event of contact situations is the contact event and the stopper event is the leaving contact event. Both have exactly the same involved objects. We represent this type of information using identity constraint rules using a property chain starting from the starter event via involved objects, stopper event, and back to the starter event.

Fluid flow situations are a bit different because there are no distinct starter and stopper event types. At some time instant the first and at a later time the last fluid flow event of a situation occurs. However, not every sequence of fluid flow events referring to the same fluid makes a situation. If the container is put aside for a while, for example, one would rather say that the situation ended then, and that a new situation starts when the container is used later on. This can be enforced by asserting that, during fluid flow situations, the container may only be salient for fluid flow events.

Motion Phases

Motions can be detected by monitoring the joint configuration of an agent. Movements are either reflexive or intentional. But at this level of our ontology, without knowing intentions of agents, we can not distinguish between reflexive and intentional motions and represent motions solely in terms of expected events and body parts used.

The different body parts are defined in the KnowRob ontology. Here, we define a general "body part moved" concept for each of these body parts. We define the functional relation *partMoved* to represent which body part moved during a motion, and restrict the range of this property to the corresponding body part type. For *ArmMovement*'s, for example, we assert: $\forall partMoved.Arm$ and $= 1 partMoved.Arm$. Force events salient for a motion are denoted by the inverse functional *event* relation. Temporal ordering constraints are asserted by temporal properties *before*, *after*, and *during*.

Here, we only investigate arm movements. Hand movements are also represented, but only at a coarse level using a boolean state: Opened or closed. We also ignore gaze motions in this work. However, it would be interesting to look into gaze contact events and to compare gaze patterns for different expert levels in future work.

Arm Movement

Arm movements are fundamental for object manipulation. The repertoire of different arm motions of humans is rich: reaching, lifting, throwing, cutting, pouring, etc. Some of which have distinct patterns of force dynamic events, such as cutting, that we use for representing them.

We use force events as delimiters of motion phases. In particular important are contact situations between body parts and other objects. Motions during which lifetime the contact between body part and object is continuously salient are called carrying motions (*CarryingMotion*). The body part in contact with the object must be part of the body part (denoted by *partOf*) which is moved during the motion. This is to allow, for example, that the contact occurs between hand and tool while the body part referred to by the motion is the arm (which in turn has a hand part).

Objects held by agents may also touch other objects or liquids during the motion, causing distinct force events during that interaction. We use this pattern of force events for the representation of tool motions. A cutting motion, for example, is a carrying motion, performed with a cutting tool, during which some object was cut into pieces. Cutting events may also be destruction events in case the object cut into pieces entirely disappeared. We further assert that the tool used in the cutting event (*cutter*) is also salient during the carrying situation.

Another challenging manipulation task is pouring. It can be performed in many ways, and on many different expert levels. The motion profiles of different expert levels are drastically different, but they all generate fluid flow events when particles are leaving the source container. We represent pouring motions as contact situations with a subgoal which is a fluid flow situation. First, we state that pouring motions are carrying situations where a container that contains some fluid is a salient object, and that at least one fluid flow situation is a subgoal of this situation. We further state that the fluid transported in fluid flow events of subgoals is exactly the fluid inside of (*contains*) the contacted container.

Activities

At the highest level of our ontology there are activities composed of motions with expected event patterns. At this level of the ontology, the intention of agents is implied by action concepts. The standard example quoted in the work of Flanagan et *al.* is a fetch-and-place activity. During fetch-and-place tasks, there is a contact situation between agent and fetched object, and also distinct events indicating that the carried object first leaves contact to a supporting surface, and later gets into contact with a supporting surface again.

We state that fetch-and-place activities have a *submotion* which is a carrying motion. And that there are two additional force events linked to the action via the *subevent* relation. We further state that there is a subevent in which the carried object looses contact to a supporting surface.

At this level, we can distinguish between colliding, supporting, and intentionally touching. Unexpected contacts during an activity are classified as collisions. This makes it very easy to detect them. With expected we mean that the activity concept asserts their occurrence during the activity.

We use the same scheme to distinguish between pouring and spilling: Pouring actions have intended fluid flow subgoals while spillage events are exactly the unintended fluid flow events occurring during an action. More concretely, pouring actions have a *target* location where the fluid should be poured into or onto. We classify all fluid flow events where the fluid is transported to somewhere else then the target location as spillage events.

Experience of Episodic Memories

Experience data captures low-level information about experienced activities represented as time series data streams. This data has often no or only unfeasible lossless representation as facts in a knowledge base. To make this data *knowledgable*, procedural hooks are defined in the ontology to compute relations from the experience data, and to embed this information in logic-based reasoning.

The data is stored in a NoSQL database using JSON documents. Each individual type of data is stored in a collection named according to the type of data stored in it. When imported, the knowledge system stores the data in a MongoDB [2] server, for which the knowledge system implements a client for querying the data during question answering.

Pose Data

A robotic system typically has many mobile components arranged in a kinematic chain. Each component in a kinematic chain has an associated named coordinate frame such as world frame, base frame, gripper frame, head frame, etc. 6 DOF relative poses are assigned to frames. These are usually updated with about 10 Hz during movements, and expressed relative to the parent in the kinematic chain to avoid updates when only the parent frame moves. The transformation tree is rooted in the dedicated world frame node (also often called map frame).

[2]https://www.mongodb.com/

The data is used by our knowledge system to answer questions such as: *"Where was the base relative to the object, 5 seconds ago"*.

Reasoning with Episodic Memories

The knowledge represented in acquired experiences is very comprehensive. It not only contains narrations of activities but also raw experience data. Competent robot behavior needs both: Experience data encodes particularities of motions such as forces and velocities, and the narrative is required to make sense of the data at higher cognitive levels.

Here, we provide reasoning examples with our action representation. We first describe how activities can be obtained from force events, and also how an agent can make sense of action concepts. We finally outline some analytical reasoning tasks that can be performed on episodic memories.

Activity Parsing

In virtual worlds, force dynamic events can be monitored perfectly. These can be asserted to the knowledge base as they occur. Given the occurrence of force events, we can infer new knowledge using descriptions from higher levels of our ontology. In the first step, the events are expanded to situations. The situations are then refined to motions with distinct force event patterns. Finally, high level activities are detected based on patterns of force events and motions.

Expanding Force Events

The expansion process exploits representations of situation concepts to identify events that determine the situation. Situations are determined by so called starter and stopper events. The events are processed from earliest to latest. A situation symbol is created when a starter event was detected, and a triple that specifies the *starter* relation is asserted. The procedure stores a list of situations without stopper events. For each new event, this list is first iterated to test whether the event is a stopper event of the situation, and a triple that specifies the *stopper* relation is asserted if this is the case. Finally, it is also tested if new events are sub-event of one of the situations without stopper.

Classifying Motions

We assume that arm motions are only segmented by zero velocity segmentation in advance. We use force events as delimiters for coarse-grained segmentation. We think that this segmentation is sufficient because it captures the force events which are the essential subgoals of manipulation activities. Here, we only consider arm motions. For each situation during which an arm motion occurred, we iterate through the different subclasses of *ArmMotion* which are also contact situations, and we test if classifying the situation with that type would yield a contradiction. The motion type is asserted if this is not the case. The motion classes are disjoint such that situations can only be classified as being instance of one of the motion classes.

Parsing Activities

Motions and force events are then used as building blocks for activities. Activities can be parsed using rules that detect temporal patterns of events and motions that are distinctive for them. Force events and motions that are subgoals of activities are denoted by the *subevent* and *submotion* properties. Patterns with partial ordering constraints can be inferred from this model. The output of the parser is an ontology, describing instances of detected actions. Here, we provide one hand-written rule that is used to detect pick-and-place activities shown in Algorithm 1.

Algorithm 1 Detect Pick-and-Place

```
 1: procedure DETECT-PICK-AND-PLACE
 2:     $CarryingMotion(?s)$, $contact{+}_S(?s, ?obj)$,
 3:     $LeavingContactEvent(?ev1)$,
 4:     $loose\text{-}support\text{-}event(?s, ?ev1, ?obj)$,
 5:     $ContactEvent(?ev2)$,
 6:     $gain\text{-}support\text{-}event(?s, ?ev2, ?obj)$,
 7:     $before(?ev1, ?ev2)$.
 8: procedure LOOSE-SUPPORT-EVENT(?s,?ev,?obj)
 9:     $contact\text{-}(?ev, ?obj)$, $contact\text{-}(?ev, ?t)$,
10:     $SupportingSurface(?t)$,
11:     $stopper(?s, ?x)$, $before(?ev, ?x)$.
12: procedure GAIN-SUPPORT-EVENT(?s,?ev,?obj)
13:     $contact{+}(?ev, ?obj)$, $contact{+}(?ev, ?t)$,
14:     $SupportingSurface(?t)$,
15:     $starter(?s, ?x)$, $after(?ev, ?x)$.
```

Activity Interpretation

Our ultimate goal is to enhance the performance of robots by supplying them with knowledge about everyday activities, and in particular with high-level stories about what happened combined with experience data. In this section, we provide a description of how robots may use the information represented in episodic memories.

A typical query first asks for a particular semantic action that fulfills certain constraints such as being successful, being performed by a particular agent, etc. The inferred action symbol is bound to a variable which is used as index to sub-symbolic data in the experience part of episodic memories. This is done to access data slices corresponding to the semantic activity for which the symbol was inferred earlier. An example of such a query is shown in the following:

```
entity(Act, [an, action, [type, putting_down]]),
occurs(Act, [_,End]),
holds(pose(pr2:'pr2_base_link',Pose), End).
```

Which corresponds to the question *"Where did the robot stand at the end of put-down actions?"*.

Based on our model, we can also ask questions about the goals of an action, for example, *"What motion phases are the subgoals of an action"*. For our introductory example of a robot closing a drawer (see Figure 1), the motion phases can be queried with a query such as:

```
entity(Act, [an, action, [type, closing_a_drawer],
     [part_moved, [an, object, [base_link, HandBase]]]]),
findall(M, entity(Act, [sub_motion, M]), Motions).
```

For a more detailed description of the question answering system used here, please consult the system paper written by Beetz et *al.* [2].

Activity Analytics

Episodic memories are very comprehensive and additional tools for inspection are required. For illustration, we pick one simple pick-and-place task performed by a robot and show how our visual analytics tools are used to get insights about manipulation activities and reasoning processes. Our goal is to provide tools for gathering data for learning algorithms, and to learn about the requirements for robots performing everyday activities. Clustering methods may be used, for example, to group actions based on their parameterizations, and to identify e.g. what kind of actions require two arms to be performed successfully, or what kind of actions require additional tools. Different components of our analytics framework will be described below.

Action Hierarchy Visualization

Cognition-enabled plan frameworks, such as CRAM [3], generate action hierarchies instead of sequences of actions. This is because, in cognition-enabled plans, most actions are abstract and require reasoning which results in action hierarchies. For instance, a pick and place action requires a "pick" sub-action to be performed followed by a "place" sub-action. Action hierarchies are stored in our episodic memory as symbolic data. To get a better understanding of an experiment, openEASE contains a component to visual the whole action hierarchy. This visualization gives an overview

	collision	*not found*	*unreachable*	*manipulation*	*not found*
MovingToLocation	13				
VisualPerception		5			
FetchAndDeliver			2	1	1
PickingUpAnObject			2		
MovingToOperate				1	
LookingForSomething					1

Figure 2: Co-occurrence matrix between actions and errors. The cell values indicate how many times the error occurred for each action type.

about what actions were executed by the robot, the relationship between those actions and which tasks were successful and which not.

With our visual analytics framework we want to go beyond showing just hierarchies and statistics. Each visual component is linked to the knowledge base which allows us to perform queries on the displayed data. To be specific, the nodes in the action hierarchy can be selected by the user, and the user can ask queries about them such as getting the error type of an unsuccessful task, the time duration, etc. In addition, trajectories during actions can be queried and visualized. Having the experience data linked to the narrative of an activity further allows to correlate success of an action with e.g. the goal pose relative to the base.

Visualization of Errors

For every episodic memory we can request a co-occurrence matrix between actions and errors which occurred during an activity. Figure 2 shows an error matrix for a pick and place activity. The rows and columns can be sorted by frequency to get quickly an overview which actions failed the most or which error type occurs the most. Referring to Figure 2, the matrix shows that most failed action was *MovingToLocation* due to collision.

We are also using the error matrix to extract action preconditions which were not considered during plan design. Currently we are extracting the preconditions manually. In the future we are planning to automatize this extraction so the robot can extend its action model by itself.

The matrix is also linked to the knowledge base, this allows us to query detailed information about the errors. For instance, for perception errors we can query which objects could not be perceived. Those queries can give us an overview e.g. for which objects the perception system might need to be improved.

	location	*action*	*motion*	*object*	*grasping*
MovingToLocation	417	35	35		
BaseMovement	2	24	24		
LookingAt	18	12	12		
VisualPerception		9	29		
LookingFor	15	14			
Reaching	14	3	10		
MovingToOperate	7	14		2	
PickingUpAnObject	7	4	1	2	1
Retracting	6	2	5		
PuttingDown	4	2		2	1
LiftingAnArm	4	1	3		
OpeningAGripper		4	4		
LoweringAnArm	2	1	2		
Pulling	2	1	2		
FetchAndDeliver		5			
AcquireGrasp	2	1	2		
ClosingAGripper		2	2		
SettingAGripper		2	2		

Figure 3: Co-occurrence matrix between actions and reasoning tasks. The cell values indicate how many times a reasoning task was performed during each action type.

Visualization of Reasoning Tasks

Cognition-enabled plans require a significant amount of reasoning. We provide multiple visualization tools available to get insights about reasoning processes. Figure 3 shows a co-occurrence matrix with the action types (rows) and the reasoning questions (columns) which are asked during a pick and place action. This matrix gives an overview which reasoning tasks were performed the most and which tasks required the most reasoning. In our example, a significant amount of spatial and perception reasoning tasks were performed.

Our analytics framework serves additional statistics, such as depicted in Figure 4. The left pie chart shows the ratio between the frequency of reasoning tasks compared to actions. A high number of reasoning tasks indicates the robot performed a very abstract plan since it required a lot of reasoning to be able to execute it. The right pie chart in Figure 4 depicts an overall time usage between reasoning and action execution. Note that even though the general amount of reasoning tasks is significantly higher than the number of actions, the action execution requires the most time. This insight gives us the the opportunity to let the robot do more expensive reasoning in the future without extending the overall experiment runtime because we could run the reasoning in parallel during the action execution.

Conclusion

In this paper, we have introduced an approach for representing episodic memories of embodied agents performing manipulation tasks. The action model is inspired by a model from human psychology. Its representations are based on force dynamic events which are used to define semantics of action verbs. We have

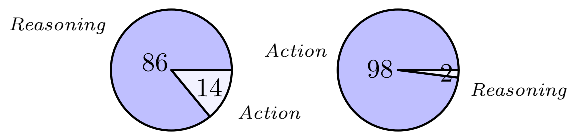

Figure 4: The left chart shows the frequency of reasoning tasks (791) compared to the number of performed actions (124). The right chart shows how much time was spend during action execution (180.61 sec) and resoning (3.68 sec).

shown that patterns of force events can be used to detect intentions, and what actions an embodied agent performed. The action model is coupled with experience data that stores control level information. We believe that collections of episodic memories are key for understanding how experiential knowledge about manipulation tasks can be generalized.

References

[1] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[2] M. Beetz, D. Beßler, A. Haidu, M. Pomarlan, A. K. Bozcuoglu, and G. Bartels. Knowrob 2.0 – a 2nd generation knowledge processing framework for cognition-enabled robotic agents. In *International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, 2018.

[3] M. Beetz, L. Mösenlechner, and M. Tenorth. Cram—a cognitive robot abstract machine for everyday manipulation in human environments. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1012–1017. IEEE, 2010.

[4] M. Beetz, M. Tenorth, and J. Winkler. Open-EASE – a knowledge processing service for robots and robotics/ai researchers. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, USA, 2015. Finalist for the Best Cognitive Robotics Paper Award.

[5] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki. Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and Systems*, 2016.

[6] J. R. Flanagan, M. C. Bowman, and R. S. Johansson. Control strategies in object manipulation tasks. *Curr. Opin. Neurobiol.*, 16(6):650–659, Dec 2006.

[7] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL–the planning domain definition language. *AIPS-98 planning committee*, 1998.

[8] B. Krüger, A. Vögele, T. Willig, A. Yao, R. Klein, and A. Weber. Efficient unsupervised temporal segmentation of motion data. *IEEE Transactions on Multimedia*, 19(4):797–812, 2017.

[9] V. Kruger, D. L. Herzog, S. Baby, A. Ude, and D. Kragic. Learning actions from observations. *IEEE Robotics Automation Magazine*, 17(2):30–43, 2010.

[10] K. Kulkarni, E. Boyer, R. Horaud, and A. Kale. An unsupervised framework for action recognition using actemes. In R. Kimmel, R. Klette, and A. Sugimoto, editors, *Computer Vision – ACCV 2010*. Springer Berlin Heidelberg, 2011.

[11] Sanmohan, V. Krüger, and D. Kragic. Unsupervised learning of action primitives. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, 2010.

[12] C. Schlenoff. *Inferring intentions through state representations in cooperative human-robot environments. (Déduction d'intentions au travers de la représentation d'états au sein des milieux coopératifs entre homme et robot).* PhD thesis, University of Burgundy, Dijon, France, 2014.

[13] C. Schlenoff, E. Prestes, R. Madhavan, P. Goncalves, H. Li, S. Balakirsky, T. Kramer, and E. Miguelanez. An IEEE standard ontology for robotics and automation. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1337–1342. IEEE, 2012.

[14] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[15] W. Takano, H. Imagawa, and Y. Nakamura. Spatio-temporal structure of human motion primitives and its application to motion prediction. *Robotics and Autonomous Systems*, 75:288 – 296, 2016.

[16] M. Tenorth and M. Beetz. A unified representation for reasoning about robot actions, processes, and their effects on objects. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October, 7–12 2012.

[17] M. Tenorth and M. Beetz. KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. *Int. Journal of Robotics Research*, 32(5):566 – 590, April 2013.

[18] E. Tulving. Episodic and semantic memory 1. *Organization of Memory. London: Academic*, 381(e402):4, 1972.

[19] N. A. Vien and M. Toussaint. Touch based POMDP manipulation via sequential submodular optimization. In *15th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2015, Seoul, South Korea, November 3-5, 2015*, pages 407–413, 2015.

[20] F. Zhou, F. D. l. Torre, and J. K. Hodgins. Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):582–596, 2013.