2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Acquiring Task Models for Imitation Learning through Games with a Purpose

Lars Kunze*, Andrei Haidu†, Michael Beetz‡
l.kunze@cs.bham.ac.uk, andrei.haidu@tum.de, beetz@cs.uni-bremen.de

***Abstract*— Teaching robots everyday tasks like making pancakes by instructions requires interfaces that can be intuitively operated by non-experts. By performing novel manipulation tasks in a virtual environment using a data glove task-related information of the demonstrated actions can directly be accessed and extracted from the simulator. We translate low-level data structures of these simulations into meaningful first-order representations whereby we are able to select data segments and analyze them at an abstract level. Hence, the proposed system is a powerful tool for acquiring examples of manipulation actions and for analyzing them whereby robots can be informed how to perform a task.**

I. Introduction

Scaling the task repertoire of autonomous manipulation robots towards open ended sets of human-scale manipulation tasks requires novel ways of efficient programming. A promising approach that enables robots to acquire skills for everyday manipulation is imitation learning or learning by demonstration [3], [6], [15]. Imitation learning analyzes the performances of humans and estimates critical parameters. The learned behavior thereby copies and imitates the movements performed by humans. Work by Albrecht et al. [2] uses methods from optimization theory to improve learned models in a post-processing step. While these approaches have proven successful to some degree they are limited because they merely copy observed behavior (movements) without understanding the interactions between objects, effects of actions, intentions behind behavior, and variations of behavior caused by different contexts.

For example, in imitation learning it is difficult to acquire a general and flexible routine for pouring pancake mix into the pan. It is hard to learn how fast and how far to turn the container and how high to hold it because the robot does not know that the demonstrator tries to avoid spilling pancake mix. Moreover, how the container is held depends on the viscosity of the pancake mix and on the size of its opening. Although some of the contextual information such as the size of the opening is perceivable, other aspects such as the viscosity of the mix are imperceivable by the robot.

Work by Chella et al. [7] addresses this challenge by incorporating symbolic knowledge about actions and states of objects into the learning process. Similarly, this work acquires semantic descriptions of intermediate task states to structure and improve learning through imitation.

*Intelligent Robotics Lab, University of Birmingham, UK. †Technische Universität München, Germany. ‡Institute for Artificial Intelligence and the TZI (Center for Computing Technologies), University of Bremen, Germany.

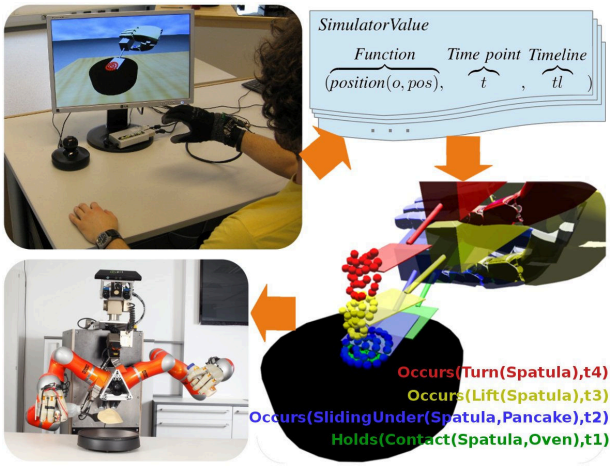

Fig. 1. Learning task models from Games with a Purpose (GWAP).

In this paper we propose a crowdsourcing approach using Games with a Purpose [1] to teach robots through imitation learning. The advantages of using games include, firstly, the ability to acquire a large number of teaching episodes without a considerable effort. Secondly, as the games are coupled with a physics simulator, physical effects and events such as contacts can directly be observed and semantically interpreted. Thirdly, the robot can actively learn by making up some situations and feeding them into the game database.

Figure 1 shows the overall idea of the crowdsourcing approach. A user performs a manipulation task in an interactive physics simulation. The computer game is started by placing all ingredients and tools specified in the instructions on the table. It then prints the individual instruction steps on the screen and asks the player to perform them. The physics simulator provides the information about object interactions, cause-effect relationships for action effects, and forces. We extended the simulator with specific physical processes such as mixing, baking, etc. The data structures of the simulator, that is, the dynamics of objects, are monitored and logged throughout the game. Afterwards the logs are translated into temporal first-order representations, called timelines. The set of acquired timelines are turned into virtual knowledge base from which the learning robot can query abstract information about the game episode in a PROLOG-based language. Eventually, comprehensive task models based on timelines can provide valuable information for robots in imitation learning. Using the query language the robot can answer queries such as:

978-1-4673-6357-0/13/$31.00 ©2013 IEEE

- What is the effect of an action?
- Which action lead to the desired outcome?
- Which action caused the current circumstances?
- Why should an action be performed?
- How should an action be parameterized to yield a different effect?

The remainder of the paper is structured as follows. We first consider the problem of making pancakes in Section II. In Section III present the overall framework for the acquisition and representation of task knowledge from games. We describe two realized games were users have to perform a pouring task and evaluate the thereby acquired data in Section IV. Finally, we present and discuss related work in Section V before we conclude in Section VI.

II. Making a Pancake

In their daily routines personal robot assistants are supposed to accomplish novel tasks for which they have not been pre-programmed. In this work, we take the task of making a pancake as our running example. Tenorth et al. [16] demonstrated how robots can extend their task repertoire by extracting natural language step-by-step descriptions from the Web and translating them into well-defined executable plans. In the work of Beetz et al. [5], we describe an experiment where our robot Rosie actually performed the task of making pancakes on the basis of such plans.

Natural language instructions are descriptive enough for humans to understand a task. However, for robots such instructions are highly underspecified. That is, within the experiment many parameters of the plan were determined and specified by programmers. But in principle, a robot has to infer the appropriate parameters of these actions by other means. By observing humans performing the task the robot can estimate some of the missing parameters. For example, the robot could estimate parameters like height and angle of the container while the pouring action is performed. Also the duration of this action could be estimated. Such information could be extracted from instruction videos retrieved from the Web or from a human tracking system [4]. Since our goal is to acquire a deep understanding of the physical effects of such manipulation actions, we propose to acquire such task-related knowledge based on games played in a physics-based simulator. When considering, for example, the pouring action we would like to answer questions such as

- how much mix was spilled during the game?
- how much was poured onto the pancake maker?
- did it form a proper pancake?
- how much mix was left in the original container?
- how long did the user hold the container above a target?
- at what height? at what angle?

III. Acquiring Task Knowledge through Games

After a brief overview of the overall framework, we explain the virtual manipulation environment, the underlying representations and reasoning mechanisms and finally we illustrate all processing steps by an example.

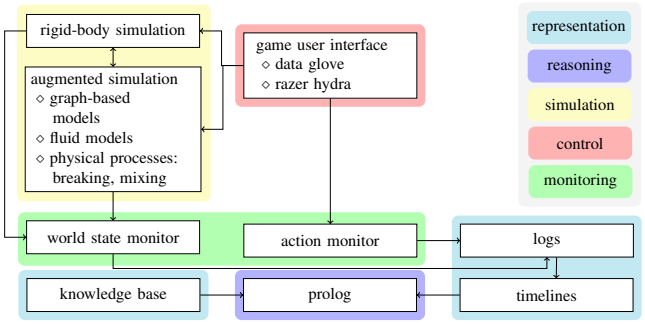

Fig. 2. Framework of the virtual manipulation environment.

A. Overview

Figure 2 gives an overview of the framework for the acquisition of task models. The framework consists of two parts: a virtual manipulation environment[1] and a knowledge processing module for the extraction and analysis of data.

In the virtual environment objects can be manipulated using a data glove and a 3D position sensor where the sensor information is directly translated into a pose and articulation of the simulated hand model. Since we have complete knowledge about the simulated world state we are able to extract different kinds of information of the task-related objects. This information include, for example, an object's position, orientation, linear and angular velocities as well as its bounding box. Also contacts between objects are reported in each time step. In contrast to vision-based systems we do not have to deal with occlusions and other typical problems like the recognition of transparent objects.

The framework, that we have designed and implemented, can be used as a tool for the acquisition of task-related information by logging the internal states of the simulator. The logged simulations are then translated into interval-based first-order representations, called timelines, as described in [13]. By formulating logical queries we can extract task-related information from these timelines semantically. For example, we can query the pose of the container while it was hold by the hand. Then, methods form statistics and machine learning can be applied on the selected data to analyze the manipulation actions with respect to various aspects.

B. Virtual Manipulation Environment

The virtual environment is based on Gazebo[2]; a 3D multi-robot simulator with rigid-body physics. In the environment a user wearing a data glove controls a robot hand which allows him to interact with objects. Figure 3 shows the hardware equipment and a user controlling the virtual robot hand.

The virtual hand is a simulated version of the DLR-HIT robot hand, described using the Unified Robot Description Format (URDF), which is an XML format for representing a robot model, and then loaded in the simulator. The hand consists of four identical fingers, each having four joints

[1] http://ias.cs.tum.edu/~kunzel/videos/virtual_env.mp4

[2] http://gazebosim.org

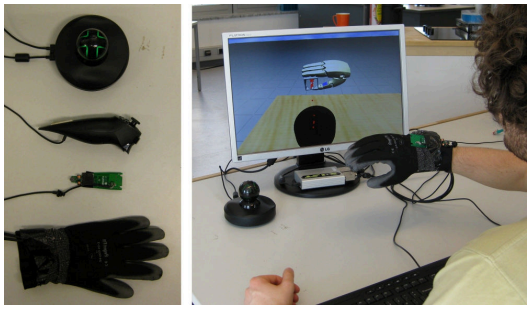

Fig. 3. Virtual Manipulation Environment.

except the thumb which has an additional degree of freedom for mimicking the human opposable thumb.

The data glove we use for detecting the positions of the finger joints is the X-IST Dataglove. It is equipped with 15 bend sensors (three per finger, one for each joint).

For detecting the absolute position and orientation of the hand, we use the Razer Hydra gaming controller. It has a base station that emits a weak magnetic field, and with the help of the sensors, that were initially integrated in the controllers and afterwards attached to the data glove, we can have a true six degree-of-freedom motion tracking.

The position of the virtual hand is controlled by calculating at each simulation time step (1000 hz) the required linear/rotational velocities that need to be applied on it in order to move to the desired position. Having the difference between the simulated position and the real one a proportional-integral-derivative (PID) controller returns the required velocities in order to have smooth hand movements. For simplifying the controller, the gravity force acting on the hand was disabled, which does not influences its behavior as the inertial forces are still present. The fingers are controlled in a similar manner.

C. Representation and Reasoning about Manipulation Tasks

Within the framework, monitors track the state evolution of the simulator and log information whenever there is a difference between two succeeding states. Hence, logged simulations are a sequence of states over time. We basically distinguish between two kinds of states, namely *World states* and *Contact states*. World states comprise the position, orientation, linear and angular velocities, as well as the bounding box of an object at a certain point in time:

World state : $\langle$*time, obj, pos, orient, lin_vel, ang_vel, bbox*$\rangle$.

A contact state holds information about the number of contacts (*num*) between two objects at a point in time. Additionally, it includes the respective forces, torques, and normals for each of these contact points:

Contact state : $\langle$*time, o1, o2, num, force, torque, normal*$\rangle$.

Data structures of the logged simulations are accessed using a predicate, called *SimulatorValue*, as follows:

$$SimulatorValue(\overbrace{position(o, pos)}^{Function}, \overbrace{t}^{Time\ point}, \overbrace{tl}^{Timeline}),$$

whereby different functions are available for accessing the time-stamped information of the world and contact states.

By using the above predicate, logged simulations are translated into interval-based first-order representation, called timelines. We access and evaluate the timelines from PROLOG by using predicates similar to those in the Event Calculus [12]. The notation is based on two concepts, namely fluents and events. Fluents are conditions that change over time, e.g., a mug contains a pancake mix: *contains(mug,mix)*. Events (or actions) are temporal entities that have effects and occur at specific points in time, e.g., consider the action of pouring the mix from the mug onto the pancake maker: *occurs(pour(mug,pan))*. Logical statements about both fluents and events are expressed by using the predicate *Holds(f,t,tl)* where f denotes a fluent or event, t simply denotes a point in time, and *tl* a timeline. The following logical formulas show how the fluent *on* is based on two other fluents, namely *contacts* and *above*, which in turn are grounded in the data structures of the simulator:

$$\begin{aligned} &Holds(on(o_1, o_2), t_i, tl) \Leftrightarrow \\ &\quad Holds(contacts(o_1, o_2), t_i, tl) \wedge \\ &\quad Holds(above(o_1, o_2), t_i, tl) \end{aligned}$$

$$\begin{aligned} &Holds(contacts(o_1, o_2), t_i, tl) \Leftrightarrow \\ &\quad SimulatorValue(contacts(o_1, o_2), t_i, tl) \end{aligned}$$

$$\begin{aligned} &Holds(above(o_1, o_2), t_i, tl) \Leftrightarrow \\ &\quad SimulatorValue(bbox(o_1, bbox_1), t_i, tl) \wedge \\ &\quad SimulatorValue(bbox(o_2, bbox_2), t_i, tl) \wedge \\ &\quad Above(bbox_1, bbox_2). \end{aligned}$$

Using the predicates *Holds_tt(f,ti,tl)* and *SimulatorValue_tt(f,ti,tl))* where *ti* denotes time interval we can even query for a complete interval throughout a fluent holds or an event occurs.

The following simplified excerpt shows how a pouring action can be defined using the above mentioned language elements. The *pour* predicate is true if there is some mix X inside the *Mug* at the beginning of the timeline and if there is a subset of X, namely Y, inside the *Pan* at the end of the timeline. Note that this predicate does not determine when the action has happened and whether there has been mix spilled onto the table.

```
occurs(pour(Mug,Pan)) :- hasType(Mix,liquid),
                         partOf(X,Mix), subsetOf(Y,X),
                         holds_tt(in(X,Mug),I1,TL),
                         holds_tt(in(Y,Pan),I2,TL),
                         begin(TL,Begin), end(TL,End),
                         starts(I1,[Begin,End]),
                         finishes(I2,[Begin,End]).
```

Similarly, we can formulate first-order queries to retrieve the answers to questions as listed at the end of Section II.

D. An Example: Acquiring Knowledge of a Pouring Task

In this section we describe how task knowledge can be acquired from execution traces of the virtual environment.

A user performed a task related to the *making pancakes* scenario, namely pouring pancake mix onto a pancake maker. Figure 4 illustrates how the task was performed in the virtual manipulation environment.

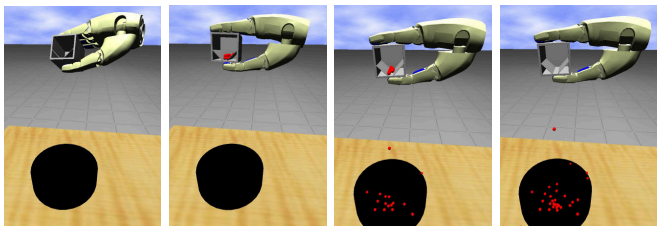

Fig. 4. Virtual Manipulation Task: Pouring pancake mix onto a pancake maker.

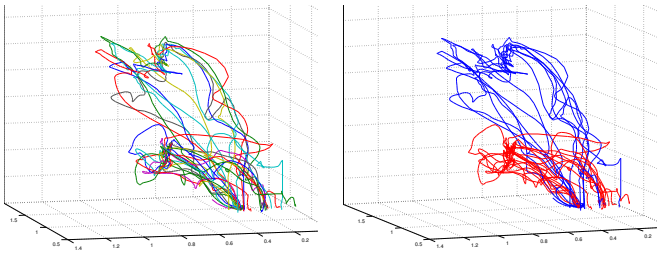

Fig. 5. Trajectories of the mug in Euclidean space when it was in contact with the hand. Raw (left) and clustered (right) trajectories after aligning them using dynamic time warping.

By translating the data structures of the simulator into timelines we can use first-order logic to query task-related data semantically. For example, we can ask for the poses of the mug in a time interval where there was a contact between mug and the robot hand as follows:

```
?- holds_tt(contacts(mug,hand),I,TL),
   simulator_value_tt(position(mug,Ps),I,TL),
   simulator_value_tt(orientation(mug,Os),I,TL).
```

where I denotes a time interval and the other variables denote lists of their respective data types. Similarly, we can get the last position of the mug in that interval for analyzing where the user has placed the mug after the pouring.

In the experiments liquid was poured from different heights which can be seen by clustering the trajectories (Figure 5). We first applied dynamic time warping to align the trajectories of different length in time and then we clustered the trajectories as in [2].

Logical queries allow us to select data segments of the logged simulations on an abstract level. For example, we can select only data when the mug is above the pancake maker or when it is tilted at an angle in a certain range.

IV. Experimental Results

We set up two games within the virtual environment for acquiring task knowledge. In both games the task is to pour pancake mix onto a pancake maker. However, the conditions and contexts in the games were varied. For each game we first explain its initial conditions and the task the user has to achieve, and second, we describe how we analyze and evaluate the extracted data.

A. Pouring without spilling

1) Overview: Within the first game, the task of the user is to pour pancake mix onto a pancake maker without spilling[3]. The simulation is initialized with a pancake maker and a mug on a table. The virtual robot hand of the user is floating around in the environment. The user has to grasp the mug, move it to a position over the pancake maker, tilt it so that the mix flows out of it onto the pancake maker, and eventually put it back onto the table. Since the user should not spill anything, he/she has to be careful while performing the task.

To analyze the behavior of users with respect to the viscosity of liquids we changed the fluidity of the pancake mix within the game.

The model of the liquid has a controller attached that sets a given damping factor to each of its particles. The damping is realized by multiplying the current angular velocity of each particle with one minus the damping factor value (1-8) multiplied by 0.05 at every time step (1000 hz). Hence we have a controllable level of viscosity for the liquid.

We used eight different levels of fluidity. For each level the user has to perform ten trials, i.e, 80 trials in total. However, the user does not know the level of fluidity in advance. So, he/she has to experience it in the first round(s) of each game.

Our hypothesis is that a user would lower the position of the mug while pouring if the fluidity of the mix is increased in order to prevent the liquid to be spilled onto the table. On average we would expect also an increased angle while tilting the mug if the mix has a higher viscosity to increase the flow velocity.

2) Results: In total, we analyzed 80 trials of eight different levels of fluidity with respect to various aspects. The logged data of the individual trials is represented using the timeline data structures which allows us to make queries using a first-order language. For the analysis we basically selected the data from the interval where the user hold the mug above the pancake maker and tilted it by more than 30 degrees. The following query shows how the data was selected

```
?- holds_tt(tilted-X(mug,pi/6),I1,TL),
     holds_tt(above(mug,pancake_maker),I2,TL),
     cooccur(I1,I2),
     simulator_value_tt(data(mug,Data),I1,TL).
```

where $I1$ and $I2$ denote time intervals on a timeline TL, and $Data$ includes all information about the mug within a given interval, e.g., its position, orientation, linear and angular velocities, and contacts.

First of all, we evaluated whether the user spilled liquid onto the table for the different levels of fluidity. Figure 6 show that liquid was spilled in almost all trials when the fluidity was high. The number of trials in which the user spilled something decreased when the viscosity increased.

Generally the acquired data indicates how the user became acquainted with the task and optimized his behavior during the ten trials across all damping factors. Both duration and height decreased when the game advances to the next round.

[3]Video: https://ias.cs.tum.edu/~kunzel/videos/exp1-viscosity.mp4

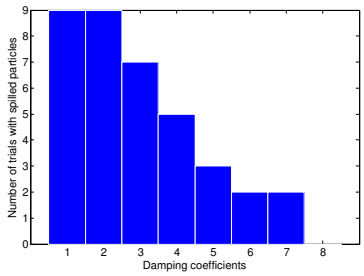

Fig. 6. Number of games with spilled particles for different damping coefficients.

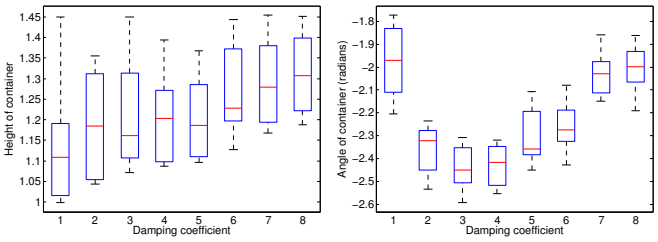

Fig. 7. Left: Height of container for different damping coefficients. Right: Angle of container for different damping coefficients.

However, more interesting is how the observed behavior changes for the different levels of viscosity. We analyzed the pose of the mug in the interval when it was tilted above the pancake maker. For each of the ten trials we calculated the arithmetic mean of height and angle during that interval. Figure 7 shows that both height and angle generally increase if the viscosity increases. An exception can be seen for the angle with the lowest damping value. Maybe this is due to the fact that this was the first game played by the user.

We also calculated the Pearson product-moment correlation coefficient for the height of the container and the level of viscosity. The *p-values* indicate that the correlation between the variables is significant (*p-value* of 0.0001). However, this result should be verified by a larger user group.

B. Pouring the Right Amount

1) Overview: The task of the user in this game is to pour a certain amount of pancake mix onto the pancake maker[4]. Similar to the other game, the simulation is set up with a pancake maker and a container containing some pancake mix. The user should grasp the container with the robot hand, pour a certain amount of its contents onto the pancake maker and place it back onto the table.

Within the game we varied two conditions. First, the user is presented either a mug or a bottle of pancake mix. The opening of the latter is smaller, i.e., less liquid flows out of the container while tilting it. Second, we varied the filling level of the container. In total, we looked at four filling levels of the respective containers. The amount of pancake mix the user is asked to pour corresponds to the lowest filling level. Figure 8 shows different types of containers and fill levels.

[4]Video: https://ias.cs.tum.edu/~kunzel/videos/exp2-container-level.mp4

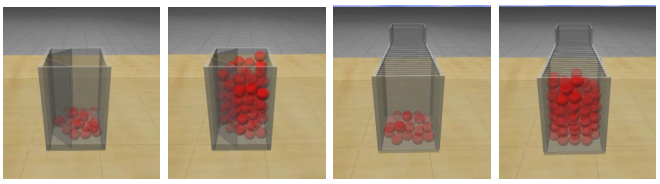

Fig. 8. Different types of containers and different filling levels.

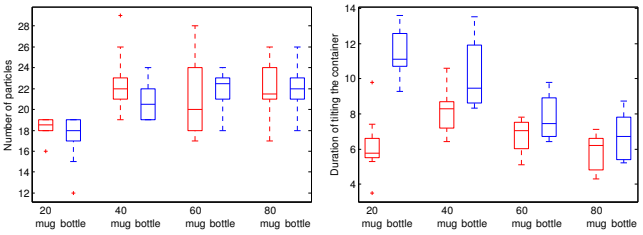

Fig. 9. Left: Number of particles on pancake maker for different types of containers and filling levels. Right: Duration of tilting the container during the pouring action.

By performing the task in various contexts, i.e., with different types of containers and filling levels, we want to analyze whether the behavior of users is context dependent. If our hypothesis is true, we should be able to extract parameters like pouring angle, height and time that depend on the task context. Overall the user performed 80 trials, 40 for each container and the different filling levels.

2) Results: Firstly, Figure 9 (left) shows that the user was able pour an amount reasonable close to the target amount of 20 particles onto the pancake maker with both containers at all filling levels. Figure 9 (right) indicates that the user generally poured longer when the container was a bottle. This makes sense since the bottle has an opening that is smaller than that of the mug.

Figure 10 (left) illustrates that the height for both containers decrease on average when the filling level increases. Figure 10 (right) shows opposing results with respect to the tilting angle for the different container types. The greater the angle, the steeper the inclination of the container. We will investigate this behavior by performing more trials and an in-depth analysis of the data.

V. Related Work

Physics-based simulators have successfully been used to teach surgeons how to perform surgical procedures [14].

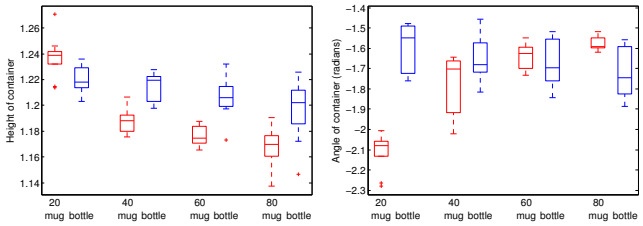

Fig. 10. Left: Height of container. Right: Angle of the tilted container in the main pouring direction.

Within robotics, they have been deployed in the context of planning [17] and navigation [8].

Games with a Purpose have been used to acquire commonsense knowledge from Internet users [1]. However, most of these games focus on image and natural language tasks.

Work by [4] describes an approach of markerless tracking of human motions. In this work we used a data glove and a Razer Hydra sensor to reliably control a virtual hand.

Work on imitation learning often focuses only on the imitation ([11]) and/or optimization ([2]) of observed trajectories. That is, information about the context, effects, and the user's intentions are neglected. Only recently, there are approaches that started to consider effects of actions in imitation learning [10]. Work by [9] uses zero-velocity movements to provide more structured information about the task. The present work extracts high-level information about task states and its structure from logs of physical simulations.

VI. Discussion and Conclusions

In this paper we have presented an approach for acquiring knowledge about manipulation tasks through Games with a Purpose. Within a game a user is instructed to perform a manipulation action in a virtual environment. In the simulated environment the user manipulates objects by controlling a robot hand with a data glove. The data structures of the simulator are monitored, logged, and translated into timelines. Logical queries on timelines can be used to answer questions such as "Which action lead to a desired outcome?" As an example we presented a definition of a pouring action (Section III-C). These query results are further analyzed and interpreted using methods from statistics and machine learning in order to generate informative models for manipulation tasks.

In future work, we will explore how robots can learn actively missing task knowledge. To this end, we will investigate how the initial configurations of games can be automatically generated based on perceived situations.

The acquired task knowledge from games goes beyond the information that is usually extracted in imitation learning. Whereby imitation learning often learns and optimizes stereotypical trajectories we also consider information about the task context as well as the relationship between objects with respect to spatial and physical aspects during the task execution. Thereby we build models that reflect, for example, the situated context, physical phenomena, and intended goals.

Acquiring data from games has also the advantage that large amounts of information can be obtained from multiple users simultaneously. However, currently our system is limited to a workplace given the equipment requirements.

As a downside of our approach, we would see the problem that current state-of-the-art robot simulators are not as robust, flexible and easy to use as one would desire. However, we believe that issues regarding performance and usability will be solved by the game and animation industry which heavily employs physics engines in the game development.

Overall we believe that the proposed framework for extracting information from interactive simulations can be a useful tool for the acquisition of task-related knowledge in many areas of robotics research including monitoring, planning, diagnosis, question answering, and learning.

Acknowledgments: This work has been supported by the EU FP7 projects *STRANDS* (grant number 600623), *RoboHow* (grant number 288533) and *SAPHARI* (grant number 287513).

References

[1] L. v. Ahn. Games with a purpose. *IEEE Computer*, 39(6):92–94, June 2006.

[2] S. Albrecht, K. Ramirez-Amaro, F. Ruiz-Ugalde, D. Weikersdorfer, M. Leibold, M. Ulbrich, and M. Beetz. Imitating human reaching motions using physically inspired optimization principles. In *11th IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, October, 26–28 2011.

[3] P. Azad, T. Asfour, and R. Dillmann. Toward an Unified Representation for Imitation of Human Motion on Humanoids. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[4] M. Beetz, J. Bandouch, D. Jain, and M. Tenorth. Towards Automated Models of Activities of Daily Life. In *First International Symposium on Quality of Life Technology – Intelligent Systems for Better Living*, Pittsburgh, Pennsylvania USA, 2009.

[5] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth. Robotic Roommates Making Pancakes. In *11th IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, October, 26–28 2011.

[6] A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal. Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, 47(2-3):69–77, 2004.

[7] A. Chella, H. Dindo, and I. Infantino. A cognitive framework for imitation learning. *Robotics and Autonomous Systems*, 54(5):403–408, 2006.

[8] B. Frank, C. Stachniss, R. Schmedding, M. Teschner, and W. Burgard. Real-world robot navigation amongst deformable obstacles. In *ICRA'09: Proc. of the 2009 IEEE int. conf. on Robotics and Automation*, 2009.

[9] R. Jkel, S. Schmidt-Rohr, M. Lsch, and R. Dillmann. Representation and constrained planning of manipulation strategies in the context of programming by demonstration. In *IEEE International Conference on Robotics and Automation (ICRA 10)*, 2010.

[10] S. M. Khansari-Zadeh, K. Kronander, and A. Billard. Learning to play minigolf: A dynamical system-based approach. *Advanced Robotics*, 2012.

[11] J. Kober and J. Peters. Imitation and reinforcement learning — practical algorithms for motor primitive learning in robotics. *IEEE Robotics and Automation Magazine, 17(2), pp. 55-62*, 2010. Intelligent Autonomous Systems.

[12] R. Kowalski and M. Sergot. A logic-based calculus of events. *New generation computing*, 4(1):67–95, 1986.

[13] L. Kunze, M. E. Dolha, E. Guzman, and M. Beetz. Simulation-based temporal projection of everyday robot object manipulation. In Yolum, Tumer, Stone, and Sonenberg, editors, *Proc. of the 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Taipei, Taiwan, May, 2–6 2011. IFAAMAS.

[14] A. Maciel, G. Sankaranarayanan, T. Halic, V. Arikatla, Z. Lu, and S. De. Surgical model-view-controller simulation software framework for local and collaborative applications. *International Journal of Computer Assisted Radiology and Surgery*, 6(4):457–471, 2011.

[15] S. Schaal, A.-J. Ijspeert, and A. Billard. *The neuroscience of social interaction*, chapter Computational approaches to motor learning by imitation, pages 199–218. Oxford University Press, 2004.

[16] M. Tenorth, D. Nyga, and M. Beetz. Understanding and Executing Instructions for Everyday Manipulation Tasks from the World Wide Web. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1486–1491, Anchorage, AK, USA, May 3–8 2010.

[17] S. Zickler and M. Veloso. Efficient physics-based planning: sampling search via non-deterministic tactics and skills. In *AAMAS '09: Proc. of The 8th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2009.