2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Extracting Kinematic Background Knowledge from Interactions Using Task-Sensitive Relational Learning

Sebastian Höfer Tobias Lang Oliver Brock

***Abstract*— To successfully manipulate novel objects, robots must first acquire information about the objects' kinematic structure. We present a method for learning relational kinematic background knowledge from exploratory interactions with the world. As the robot gathers experience, this background knowledge enables the acquisition of kinematic world models with increasing efficiency. Learning such background knowledge, however, proves difficult, especially in complex, feature-rich domains. We present a novel, task-sensitive relational rule learner and demonstrate that it is able to learn accurate kinematic background knowledge in domains where other approaches fail. The resulting background knowledge is more compact and generalizes better than that obtained with existing approaches.**

I. Introduction

The kinematic structure of everyday objects is generally linked to their functional use (e.g. scissors, doors, drawers, wheels, buttons, etc.). To gain access to an object placed inside a cupboard (see Fig. 1), for example, a robot needs to open the doors of the cupboard, i.e., it must possess knowledge of the cupboard's degrees of freedom (DOF) so as to be able to actuate them. If the robot does not have an *a priori* kinematic world model, as is the case in mobile manipulation applications, the robot must discover DOF of novel objects. In many realistic scenarios, this can only be achieved by physically interacting with those objects.

We present a method for the efficient extraction of kinematic background knowledge from interactions with the world. By *kinematic background knowledge* we refer to a relational description of regularities in the world that reliably predict the presence of DOF. This stands in contrast to the *kinematic model* of the world, which describes the kinematic relationship between rigid bodies and their spatial relationship. Kinematic background knowledge is necessary for the efficient acquisition of kinematic models.

We build on prior work on learning to extract kinematic models from interactions [1]. We extend that work in important regards. Whereas previously all experiences were stored and a nearest-neighbor retrieval was used to generalize experience to novel objects, we now analyze gathered experience to extract kinematic background knowledge, forming a relational description of regularities in the world. In other words, we learn a state transition model of the world that explains the experiences. As the experiments in Section VI will demonstrate, this innovation accelerates learning, improves generalization, and produces a more compact representation of kinematic background knowledge. Furthermore, the novel approach is able to address domains of higher complexity. From an application perspective, this step towards experimental domains of realistic complexity represents the first main contribution. It is achieved by applying for the first time rule-based learning to this domain.

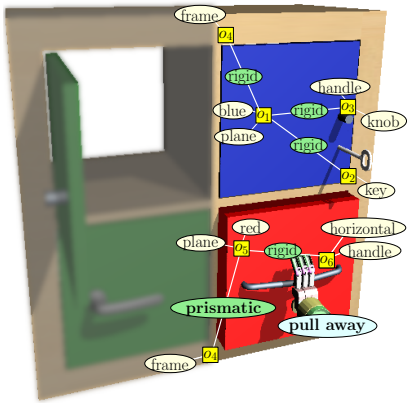


Fig. 1. Relational representation of a complex articulated object: the doors and drawers have different locking mechanisms requiring the sequential invocation of specific degrees of freedom to open the door (e.g., turn the key before the door opens). The relational representation is shown as a graph with yellow boxes denoting objects, beige ovals unary relations and green ovals binary relations. An action relation between a simulated robot hand and the red drawer is shown in blue.

Our method for extracting a compact, relational rule set from interaction experience is based on a symbolic model learner for stochastic domains [2]. The second main contribution of this paper is the extension of this learner to become task-sensitive: Experiences obtained in realistic environments can exhibit a large number of regularities, but only very few of them will be relevant to the particular task at hand. For example, the number of doors in the cupboard in Fig. 1 does in general not provide information about DOF. By making the learner task-sensitive, it is able to focus on learning rules relevant to the task at hand. As we will see in experiments in Section VI, this enables us to handle domains in which a task-ignorant learner fails.

II. Related Work

a) Perception and Manipulation: To guide its discovery of DOF in the world, the robot has to select interaction points in the environment. This can be achieved with special-purpose object detectors [3] or machine learning-based ap-

All authors are with the Robotics and Biology Laboratory, Technische Universität Berlin, Germany

We gratefully acknowledge financial support by the Alexander von Humboldt foundation through an Alexander von Humboldt professorship (funded by the German Federal Ministry of Education and Research) and through the First-MM project (European Commission,FP7-ICT-248258). We would like to thank Sebastian Koch and Dennis Loeben for their work on the simulation framework.

proaches on RGB or 3-D range data [4], [5]. However, such "single-point" approaches cannot handle kinematic structures requiring sequential actions. If the specific object category is known, policies for these sequential actions can be designed by hand, for example, in the context of doors [6]. We address the problem when not all object categories can be known precisely in advance.

A number of approaches permit the perception of DOF from visual or RGB-D data [7]–[9]. In these approaches, the interaction points are given a priori or specified heuristically, i.e. they do not address the selection of interaction points for exploration. A promising solution to this problem is to estimate the information gain of actions for minimizing uncertainty has been proposed [10]. Since this kind of methods has not yet fully matured in our context of relational learning, we use a simpler method to generate efficient exploration by rewarding the agent for discovering new knowledge [1], resembling the idea of intrinsic motivation [11].

b) Learning World Models: Planning of interactions requires information about the world. This information can be acquired from experience. It can be gathered through physical simulations and then represented in graphical models [12]. A different approach formalizes symbolic models of sensorimotor experience and behavior as Object-Action-Complexes (OAC) [13]. Similar to our approach, OAC use STRIPS-like relational rules for world modeling, but they do not address the problem of learning compact and general models from experience.

c) Reinforcement Learning: The idea of learning models from interaction is tightly linked to reinforcement learning (RL). In RL, the choice of state, action and transition function is crucial for learning success. In this work, we focus on learning a transition function from experience. One core idea is to take the task-specific reward structure of the domain into account for learning the transition function. Similar ideas have been applied in RL in the context of temporal difference learning [14]. However, these approaches only work when the optimal state representation can be expressed by a vector of fixed dimensionality. The domains we address in this paper are not amenable to this representation.

d) Relational Learning: Relational reinforcement learning (RRL) overcomes the state dimensionality problem discussed above by using first-order logic to represent states and actions [15]. Researchers have developed approaches for model-free RRL, e.g. using relational regression to the Q-function [16], as well as model-based RRL, by learning a relational transition model and using it for planning [2], [17]. The disadvantage of the model-free approach is its task-specificity: since the resulting Q-function merges domain (state transitions) and task (reward) knowledge, the solution is only applicable to one task. The model-based approach does not have this problem, but the learned transition function might fail to capture task-relevant aspects of the state space. We therefore suggest to learn a *task-sensitive relational transition model* to balance the task-sensitivity of model-free and the generality of model-based RRL.

III. Background On Relational Learning

A. Relational Markov Decision Process

We will now introduce the relational Markov decision process which allows us to formalize the problem of extracting kinematic background knowledge. A relational Markov decision process (RMDP) is defined as (S, A, T, ρ) with states S, actions A, transition probabilities T, and rewards ρ. Relational MDPs differ from normal MDPs by employing a *relational representation* of states and actions.

1) Relational States: Every state corresponds to a set of positive and negated literals, which each take a number of arguments. An example for a binary literal is $\texttt{attached}(o_1, o_2)$ (objects o_1 and o_2 are attached to each other). By combining literals, we can describe complex scenes. For example, we describe the upper right part of the cupboard in Fig. 1, consisting of a wooden frame o_4, a door o_1, a key o_2, and a knob o_3, as follows:

$$\begin{aligned}&\texttt{blue}(o_1), \texttt{plane}(o_1), \texttt{key}(o_2), \texttt{attached}(o_1, o_2), \texttt{rigid}(o_1, o_2)\\ &\texttt{handle}(o_3), \texttt{knob}(o_3), \texttt{attached}(o_1, o_3), \texttt{rigid}(o_1, o_3),\\ &\texttt{frame}(o_4), \texttt{attached}(o_1, o_4), \texttt{rigid}(o_1, o_4), \ldots\end{aligned} \tag{1}$$

All other literals are implicitly false.

2) Relational Actions: A relational action corresponds to a positive literal, for example $\texttt{pushUp}(\texttt{o}_1)$ (push up object o_1) or $\texttt{pullAway}(\texttt{o}_1, \texttt{o}_2)$ (pull object o_1 away from o_2).

3) Experiences: By interacting with the world, the robot collects experiences. We define an experience as a quadruplet $(s_{pre}, a, s_{post}, \rho)$ consisting of predecessor state $s_{pre} \in S$, action $a \in A$, successor state $s_{post} \in S$, and reward $\rho \in \mathbb{R}$. In the following, we denote a set of experiences by $\mathcal{E}$.

B. Relational Rules

Model-based reinforcement learning attempts to deduce the transition function T from experiences in order to compute a policy. We learn a *relational* transition function—a rule set $\mathcal{R}$—from a set of experiences $\mathcal{E}$ [2].

The relational rule learner infers a set of *noisy indeterministic deictic (NID) rules* $\mathcal{R}$ from $\mathcal{E}$. NID rules are a more expressive, probabilistic version of STRIPS rules. Their probabilistic nature explicitly accounts for uncertainty and failures in unstructured environments. A NID rule consists of a context, an action, and a set of outcomes, each associated with a probability:

$$\begin{aligned}&\textit{CONTEXT}: && \texttt{handle}(X), \texttt{frame}(Z), \texttt{attached}(X, Y), \texttt{attached}(Y, Z)\\ &\textit{ACTION}: && \texttt{pullAway}(X, Y)\\ &\textit{OUTCOMES}: && \begin{cases} 0.59 & \texttt{revolute}(Y, Z), \neg\texttt{rigid}(Y, Z)\\ 0.40 & \emptyset\\ 0.01 & \textit{noise}\end{cases}\end{aligned} \tag{2}$$

Note that NID rules can have *noise* outcomes in order to ignore experiences with unexpected effects (e.g., failures in perception or action). Also note that an NID rule removes reference to concrete objects $o_1, o_2, \ldots$ by substituting variables $X, Y, \ldots$

How do we apply NID rules to states and experiences? Intuitively, we say an NID rule $r \in \mathcal{R}$ *covers* a state s and an action a if we can substitute the variables in r such that

its context and action match with s and a. For example, the NID rule in Eq. 2 covers example state 1 and action $\texttt{pullAway}(o_3, o_1)$.

C. Rule Learning

The exact solution to the rule induction problem was shown to be NP-complete, rendering the computation of a globally optimal set of rules practically infeasible for large experience sets. Therefore, Pasula et al. [2] restated the problem as an optimization problem and solved it (non-optimally) using a greedy hill-climbing procedure.

The learner starts with a rule set containing only one rule with an empty and a noise outcome. It then iteratively generates new candidate rule sets by applying different operators which add, alter, and remove rules from the rule set. Every candidate rule set is then evaluated with respect to a score function and the best candidate is repeatedly selected until a local maximum of the score function is reached. The *default score function* for a rule set $\mathcal{R}$ is defined as follows:

$$S(\mathcal{R}) = \sum_{(s,a,s') \in \mathcal{E}} \log P(s'|s, r_{s,a}) - \alpha \sum_{r \in \mathcal{R}} \text{PEN}(r), \qquad (3)$$

where $r_{s,a}$ is the uniquely covering rule for (s,a). The first term maximizes the log-likelihood of the experiences given the rule set; it increases as the rule set explains the experiences more exactly. The second term is a regularization term penalizing complex rule sets. The overall penalty is obtained by summing the penalty $\text{PEN}(r)$ of each rule r, defined as the number of literals appearing in a rule. The parameter α scales the influence of the regularization term and trades off how accurate the rule set is versus how compact it is. The rule learner has an additional parameter $p_{\min}$ to control the probability of subsuming an experience not explained by the rule set under the noise outcome of some rule, rather than creating a new rule to explain it.

IV. Task-Sensitive Learning from Interaction

We now extend the relational rule learner to allow task-sensitive rule learning. At the same time, we instantiate the general learning algorithm to our specific task: extracting kinematic background knowledge from interaction.

A. Relational Representation

In our experiments, we present different articulated objects to the robot. The robot's goal is to detect all joints of presented objects with the least possible number of actions. It achieves this by extracting task-specific regularities from its experience and by applying this knowledge to select subsequent actions.

To model the learning problem as a relational Markov decision process, we now define states, actions and the state transition function.

1) State: The relational state includes two pieces of information: (i) visual information about the object and (ii) the robot's current belief about the kinematic model of the objects present in the scene.

We assume that visual information is generated by a perception system which gives a segmentation of the scene into objects (or rigid bodies) o_j and annotates each part with different features, such as their color, size, and spatial relationships. Every feature is encoded using a relational symbol.

Initially, the robot assumes rigid bodies to be rigidly connected, captured by commutative literal $\texttt{rigid}(\texttt{o}_\texttt{i}, \texttt{o}_\texttt{j})$ for parts o_i and o_j. Similarly, the robot denotes a belief about a prismatic joint using the binary literal $\texttt{prismatic}(\texttt{o}_\texttt{i}, \texttt{o}_\texttt{j})$ and the unary literals $\texttt{isPrismatic}(\texttt{o}_\texttt{i})$, $\texttt{isPrismatic}(\texttt{o}_\texttt{j})$. Revolute joints are encoded using the literals `revolute` and `isRevolute`.

A partial example of a state representation is shown as a graph in Fig. 1 and as a logic formula in Eq. 1.

2) Action: The robot can act upon an object by pushing, pulling, or rotating a rigid body. We define a set of unary actions with respect to a global reference frame ($\texttt{pushUp}, \texttt{pushDown}, \ldots$) and a set of binary actions with respect to a reference frame specified by the two rigid bodies (`pushTowards`, `pullAway`, `pushPerpendicularForward`, `pushPerpendicularBackward`, `rotateClockwise`, `rotateCounterClockwise`, $\ldots$). The local reference frame is defined by gravity and the line connecting the center of mass of the first rigid body to the closest point on the second rigid body.

3) State Transition: After executing an action, the state of the world and the robot's belief state change. The robot updates its own state by incorporating visual changes (taken care of by the perception system) as well as updating the robot's belief about the kinematic model. To update this model, the robot observes how object parts move relative to each other during an interaction. If two parts move together, the robot assumes a rigid connection. If they move in a prismatic or revolute motion, the robot inserts the relation indicating the corresponding joint to its kinematic model. The new kinematic model corresponds to the new state in the relational MDP which is the belief state of the robot. For example, if the robot detects a new prismatic joint between parts o_i and o_j, it removes the $\texttt{rigid}(\texttt{o}_\texttt{i}, \texttt{o}_\texttt{j})$ literal and adds the literals $\texttt{prismatic}(\texttt{o}_\texttt{i}, \texttt{o}_\texttt{j})$, $\texttt{isPrismatic}(\texttt{o}_\texttt{i})$ and $\texttt{isPrismatic}(\texttt{o}_\texttt{j})$. Details of how to analyze relative motion can be found in [7].

4) Rewards: The robot detects joints when relative motion between objects exceeding a threshold occurs. The threshold imitates limitations of real-world sensing. When a joint is detected for the first time, the robot receives a reward of $\rho = 1$, otherwise $\rho = -0.1$ (cost of an action). Hence, the robot maximizes its rewards by executing actions that expose unseen joints.

B. Task-Sensitive Score Function

We now describe how we can enhance the rule learner described in III-C to learn rules sets in a task-sensitive manner. The learner's default score function (Eq. 3) does not consider rewards. This makes it difficult for an autonomous robot to focus on learning task relevant knowledge. Consider again the object from Fig. 1. The blue door on the upper right can only be opened after the key (part o_2) has been turned.

However, the default score function strives for compactness and comes up with the rule in Eq. 2. Unfortunately, this rule does not capture the key opening mechanism, and leaves the robot clueless about how to reliably open the door.

Our main insight is that the rule in Eq. 2 is task-ignorant: the rule does not make an effort to distinguish whether the task is achieved and not. We therefore want to modify the score function used in relational rule set learning so that rules are favored for which similar rewards are associated with all outcomes (i.e. split Eq. 2 in two rules with more precise preconditions). This ensures that the selection of a rule can help to maximize reward.

We define a new *task-sensitive score function* as follows:

$$S_\rho(\mathcal{R}) = S(\mathcal{R}) - \beta \sum_{r \in \mathcal{R}} \mathrm{Var}_{e \in \mathcal{E}|_r}(\mathrm{reward}(e)), \qquad (4)$$

where Var denotes the sample variance and $\mathcal{E}|_r$ denotes all experiences in $\mathcal{E}$ covered by rule r. Intuitively speaking, we introduce a new term to penalize rules that cover a subset of experiences $\mathcal{E}|_r$ with very different rewards. This term counteracts the compactness objective and allows the learner to model task-relevant properties in more detail. Although the exact choice of the term is heuristic, it resembles a feature selection criterion previously considered in the context of temporal difference learning [14]. As we will see in Section VI the modification of the scoring function has tremendous impact on learning performance.

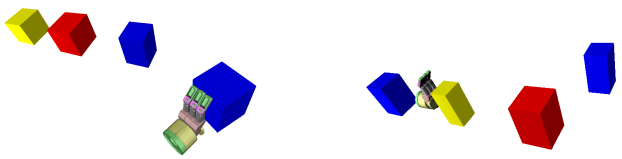
Note that the score function can be easily adapted to cope with several different tasks (reward functions) by adding one summation over the reward variances per reward function.

V. Simulated Relational Worlds

To evaluate our approach, we performed experiments with two different types of worlds, each exhibiting regularities pertaining to kinematic DOF. Experiments are conducted in simulation using a physical simulation engine based on NVIDIA PhysX™ and OpenSceneGraph. Performing simulation experiments allows us to conduct the large number of experiments required to evaluate the different learning algorithms. The simulator implements the necessary perceptual and action capabilities to perform the experiments. The physical realization of these capabilities is outside of the scope of this paper. However, we believe these capabilities have been demonstrated in real-world scenarios as part of the state of the art.

A. Kinematic Chain World

The first world consists of kinematic chains of different lengths [1] as shown in Fig. 2. Every chain exhibits the same regularity: it has two coupled joints mimicking a door locking mechanism. The first joint is revolute and is located at one end of the chain. The second joint is prismatic and is located at a random location in the chain, but always between a yellow and a red link. To unlock (and therefore be able to discover) the prismatic joint, the robot has to first actuate the revolute joint (i.e., push the door handle). All other connections are rigid, size information does not matter and was randomly assigned to the parts as a distraction.



(a) (b)

Fig. 2. Robot interacting with kinematic chains (seeing what the robot sees): A locked prismatic joint is located between the yellow and the red block. It is unlocked when the revolute joint is moved by 45 degrees. In this picture the actual joints are invisible as they are for the robot.

We give the robot a set of primitive relations to conceptualize the objects: unary relations encoding color and size (`yellow`, `red`, `white`, `small`, `middle`, `big`), the binary spatial relation `right` indicating the direct right neighbor of a box ("right" is defined with respect to the robot's coordinate frame); additionally, we provide unary relations denoting the number of left/right neighbors (`numLeft` and `numRight`) and the right and left most boxes (`leftMost` and `rightMost`). We also provide conjunctive relations for all pairs of unary relations such as `rightMostBig`, `leftMostBig`, `yellowBig`, etc. This method is similar to what is achieved by feature expansion in machine learning, as it increases the amount of information that the learner can use. Additionally, the robot uses the joint type relations described in Section IV-A.1.

B. Furniture World

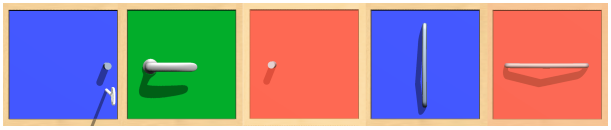


Fig. 3. Different opening mechanisms in the furniture world, from left to right: knob with key (doors only), revolute handle (doors only), knob (doors and drawers), vertical and horizontal handle (drawers only).

In the second world, the agent interacts with realistic objects of higher complexity (see Fig. 1), inspired by modular furniture. Each world consists of a cupboard with one to four elements: drawers or doors of four different colors. Each element exhibits a different opening mechanism (Fig. 3). A drawer can be opened by simply pulling the attached handle. A door exhibits one of three mechanisms: either a pullable knob, a handle which has to be pushed down before opening the door, or a knob with a key which has to be turned to unlock the door. Moreover, the door joint is either located at the left or the right. All these variations result in a set of more than 1 million possible cupboards. In our experiments, we randomly sample a set of such cupboards and let the robot interact with them to extract the regularities for different opening methods. The relations for these objects are listed in Table I.

VI. Experimental Results

We conducted experiments in the two simulated worlds to show that task sensitivity is indispensable for learning kinematic background knowledge for manipulating articulated objects. We show that (i) the task-sensitive score function

TABLE I

RELATIONS AND FUNCTIONS IN THE 3D EXPERIMENT

Relation/Function	Arity	Explanation
Object properties		
`frame, handle, plane, knob, key`	1	object type
`black, red, blue, green`	1	color
`horizontal, vertical`	1	orientation of the object
`singleAttached` `doubleAttached`	1 1	handle connected to the plane at one or two points
`keyAttached`	1	true if key attached to plane
Spatial properties		
`attached`	2	is attached to (commutative)

extracts better rule sets than the default score function, (ii) the task-sensitive rule learner generalizes faster to unseen objects of the same class than nearest-neighbor learning, and (iii) the resulting representation is more compact. We compare the following learning strategies:

(i) Nearest-neighbor strategy based on subgraph isomorphism (NN) (for a detailed description see [1])
(ii) Default rule learner (see Section III)
(iii) Task-sensitive (TS) rule learner (see Section IV)
(iv) Random (selects actions uniformly at random)

1) Experiments: Every experiment consists of a set of N episodes. During an episode we present a new object to the agent, selected uniformly at random from the set of all possible objects. An episode ends when the robot has discovered all of the object's degrees of freedom. All learners use the same object in a particular episode. The NN learner and the two rule learners update their experience after every interaction. Additionally, the rule learners compute a new rule set every time they get an experience which is not covered by their current rule set.

2) Action Selection: The NN learner and the rule learners use an ε-greedy policy with simulated annealing for action selection. They select a random action with probability $0 < \varepsilon < 1$ and a greedy action otherwise. We decrease ε in each episode n according to the following function: $\varepsilon(n) = \exp(3\frac{n}{N})$. The rule learners select a greedy action by searching for all rules which cover the current state and selecting the rule/action with the highest value. We define a rule's value as the average reward of all covered experiences, weighed by the outcome probabilities. The NN learner selects a greedy action by matching the current state with all experiences using subgraph isomorphism and applying the experience associated with the best match [1].

3) Rewards: The robot receives rewards as explained in Section IV-A.4. In the kinematic world domain the robot receives an additional reward of $\rho = 5$ when it finds both joints at the end of an episode. The reward for an episode is the average reward per action.

A. Task-Sensitivity Extracts Better Rule Sets

We now show that the task-sensitive score function allows the robot to infer better rule sets, i.e., kinematic background knowledge that captures the regularities of world better.

1) Kinematic Chain World: We compared the learners in experiments consisting of 30 episodes with objects generated uniformly at random and averaged the episode reward over five independent runs. For the rule learning strategies, we selected the hyperparameters as $\alpha = 10^{-5}, p_{min} = 10^{-7}$ and $\beta = 30$. Fig. 4 shows the results. The task-sensitive rule learner (green) and the nearest-neighbor strategy (black) perform best, as their average rewards approaches the optimal reward of 3.5 towards the end of the experiment (two actions with a reward of 1, each to discover one DOF, the additional reward of 5, normalized by two, the number of joints). This means that the task-sensitive rule-learner has "understood" the domain and exploits the regularity without explorative actions. The original rule learner performs poorly. It is sidetracked by task-irrelevant regularities and, for example, does not obtain rules describing the locking mechanism. Instead, it produces a rule to predict that prismatic joints can be found in $\approx 50\%$ of all cases.

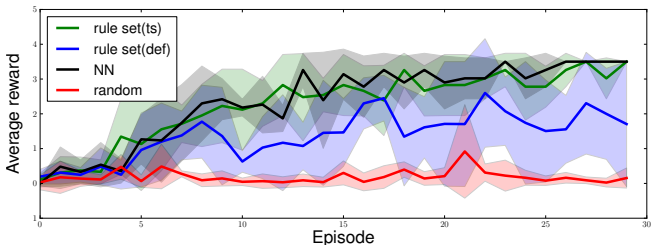


Fig. 4. Comparing different learning strategies in the kinematic chain world: We see that the task-sensitive rule learner (green) and the nearest-neighbor strategy (black) perform best as their average rewards approach the optimal reward of 3.5 towards the end of the experiment.

2) Furniture World: In the furniture world, we again conducted experiments with 30 episodes, a different random object for each episode, and averaged the reward over five independent runs. Fig. 5 shows that the task-sensitive rule learner performs better than the default rule learner.

The task-sensitive rule learner performs best, achieving an average reward of close to 1.0. An average reward of 1.0 would indicate that the robot has "understood" the domain and identifies DOF without exploratory actions. There are two reasons why our learner does not quite achieve the optimal average reward. First, the ε-greedy strategy forces the robot to select random exploratory actions, sometimes resulting in suboptimal behavior. Secondly, the stochastic nature of the object generation, action selection, and rule learning can by chance generate wrong regularities. For example, in one experiments the rule learner maintained rules until the end which take into account the (irrelevant) color of the doors. This was simply because it had actually observed this regularity, albeit by chance. However, as soon as an experience contradicts such a rule, the learner relearns a consistent set of rules. In one experiment, the learner acquired an optimal rule set after only eleven episodes.

B. Rule Learning Generalizes Faster

We now show that the task-sensitive rule learner not only outperforms the other strategies, but also generalizes faster from the data to learn kinematic background knowledge about the given object classes than the NN strategy.

1) Kinematic Chain World: Since the task-sensitive rule learner and the nearest-neighbor strategy generated comparable results on kinematic chains with four links (Fig. 4), we evaluated their performance on chains of random lengths

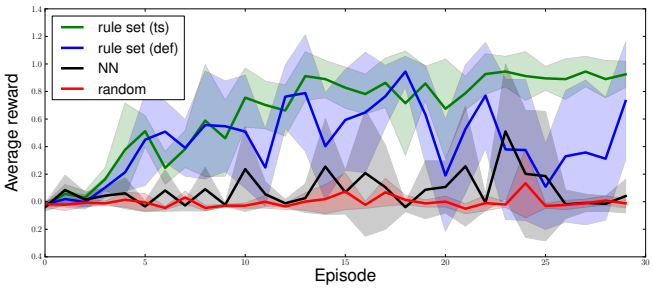


Fig. 5. Comparing different learning strategies in the furniture world: The task-sensitive rule learner is the only strategy that succeeds at learning generalizable background knowledge about the world.

between seven and ten, exhibiting the same regularity as before. We then ran all strategies on these longer chains while using the kinematic background knowledge obtained with chains of length four in the previous experiment.

Averaged over 100 random chains of length seven to ten, the task-sensitive rule learner achieves close to optimal performance (average number of actions to discover both joints: 2.1, $\sigma = 0.3$, optimal reward: 2.0). This again implies that the task-sensitive rule-learner has "understood" the domain, exploits the regularities without explorative actions, and is able to transfer the background knowledge to different objects in the same domain. In contrast, the NN strategy fails if the prismatic joint is located too far apart from the unlocking mechanism, resulting in an average of 5.8 actions ($\sigma = 3.8$). In these cases, the matching fails as NN cannot at the same time match the revolute joint at one end and the prismatic joint close to the opposite end.

In addition, the task-sensitive rule learner significantly outperforms the NN strategy with respect to compactness. The NN maintains a set of 210.8 ($\sigma = 34.9$) unique experiences on average whereas the task-sensitive rule set learner results in an average of 25.0 ($\sigma = 9.6$) rules, corresponding to a compression ratio of 8.4. (Old experiences must be retained to learn a new rule set as new experiences come in, though.)

2) Furniture World: Fig. 5 shows that the task-sensitive rule learner outperforms the NN strategy in the furniture world. Indeed, the NN strategy performs much worse than in the kinematic chain experiment mainly because of the complexity of the world. The NN strategy fails to generalize its experiences because of the many different possible combinations of cupboards. Again, the task-sensitive rule set is much more compact than the set of unique experiences: 567.8 ($\sigma = 94.4$) unique experiences for NN versus 34.0 ($\sigma = 3.8$) task-sensitive rules, resulting in an average compression ratio of 16.7.

The results show that the task-sensitive learner is the only strategy that generalizes effectively over a limited set of observations by balancing compactness and accuracy, and learns suitable kinematic background knowledge for exploring articulated objects.

VII. Conclusions

This paper presented a method for extracting kinematic background knowledge from interactions with the world. The learned relational model captures task-relevant regularities in the world, enabling predictions about the presence of DOF. The ability to acquire such kinematic background knowledge is an important prerequisite for autonomous manipulation of unknown objects, as it enables robots to efficiently discover DOF. We demonstrated this ability in two simulated domains: articulated kinematic chains and modular furniture, including doors and drawers with different locking mechanisms. In the latter domain, only solvable by our method, accurate and yet compact kinematic background knowledge was extracted after experiencing only 30 out of over one million possible pieces of furniture. The key to the effectiveness of the proposed method is a novel, task-sensitive relational rule learner.

References

[1] D. Katz, Y. Pyuro, and O. Brock, "Learning to manipulate articulated objects in unstructured environments using a grounded relational representation," in *Robotics: Science and Systems IV*, 2008, pp. 254–261.

[2] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning symbolic models of stochastic domains," *Journal of Artificial Intelligence Research*, vol. 29, no. 1, pp. 309–352, 2007.

[3] R. B. Rusu, W. Meeussen, S. Chitta, and M. Beetz, "Laser-based perception for door and handle identification," in *International Conference on Advanced Robotics (ICAR)*, 2009, pp. 1–8.

[4] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," in *Robotics: Science and Systems IX*, 2013.

[5] H. Nguyen and C. Kemp, "Autonomously learning to visually detect where manipulation will succeed," *Auton. Robots*, pp. 1–16, 2013.

[6] E. Klingbeil, A. Saxena, and A. Ng, "Learning to open new doors," in *IEEE/RSJ Int. Conf. On Intelligent Robots and Systems (IROS)*, 2010, pp. 2751–2757.

[7] D. Katz and O. Brock, "Manipulating articulated objects with interactive perception," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008, pp. 272–277.

[8] D. Katz, A. Orthey, and O. Brock, "Interactive perception of articulated objects," in *Proc. of the 12th Int. Symp. on Experimental Robotics (ISER), 2010*, ser. Springer Tracts in Advanced Robotics, O. Khatib and G. S. V. Kumar, Eds. Springer, 2014, vol. 79, pp. 301–315.

[9] J. Sturm, C. Stachniss, and W. Burgard, "A probabilistic framework for learning kinematic models of articulated objects," *Journal of Artificial Intelligence Research*, no. 41, pp. 477–526, 2011.

[10] H. van Hoof, O. Kroemer, H. B. Amor, and J. Peters, "Maximally informative interaction learning for scene exploration," in *IEEE/RSJ Int. Conf. On Intelligent Robots and Systems*, 2012, pp. 5152–5158.

[11] P.-Y. Oudeyer, F. Kaplan, and V. Hafner, "Intrinsic motivation systems for autonomous mental development," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 2, pp. 265–286, Apr. 2007.

[12] J. Mugan and B. Kuipers, "Autonomous learning of high-level states and actions in continuous environments," *IEEE Transactions on Autonomous Mental Development*, vol. 4, no. 1, pp. 70–86, 2012.

[13] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omren, A. Agostini, and R. Dillmann, "Object-Action Complexes: Grounded abstractions of sensory-motor processes," *Robotics and Autonomous Systems*, vol. 59, no. 10, pp. 740–757, 2011.

[14] J. Piater, S. Jodogne, R. Detry, D. Kraft, N. Krüger, O. Kroemer, and J. Peters, "Learning visual representations for perception-action systems," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 294–307, Jan. 2011.

[15] S. Dzeroski, L. De Raedt, and K. Driessens, "Relational reinforcement learning," *Machine Learning*, vol. 43, no. 1-2, pp. 7–52, 2001.

[16] K. Driessens and J. Ramon, "Relational instance based regression for relational reinforcement learning," in *Int. Conf. on Machine Learning*, vol. 20, Washington, USA, 2003, pp. 123–130.

[17] T. Lang, M. Toussaint, and K. Kersting, "Exploration in relational domains for model-based reinforcement learning," *Journal of Machine Learning Research*, vol. 13, pp. 3691–3734, 2012.