On the Use of Probabilistic Relational Affordance Models for Sequential Manipulation Tasks in Robotics

Bogdan Moldovan Plinio Moreno Martijn van Otterlo

***Abstract*— In this paper we employ probabilistic relational affordance models in a robotic manipulation task. Such affordance models capture the interdependencies between properties of multiple objects, executed actions, and effects of those actions on objects. Recently it was shown how to learn such models from observed video demonstrations of actions manipulating several objects. This paper extends that work and employs those models for sequential tasks. Our approach consists of two parts. First, we employ affordance models sequentially in order to recognize the individual actions making up a demonstrated sequential skill or high level concept. Second, we utilize the models of concepts to plan a suitable course of action to replicate the observed consequences of a demonstration. For this we adopt the framework of relational Markov decision processes. Empirical results show the viability of the affordance models for sequential manipulation skills for object placement.**

I. Introduction

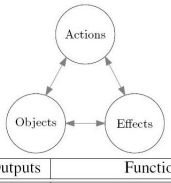
Robotics develops mobile, physical agents that reason, learn and manipulate their environment. The seminal logical STRIPS representation [8] shares its logical foundations with contemporary approaches, e.g., [11], [23]. Logic is effective at dealing with higher knowledge needed for planning and reasoning, but the physical aspect of robots requires dealing with uncertainty from noisy sensors and uncertainty about the world [3]. The use of probabilistic techniques is widespread in robotics [24], yet mostly without employing rich logical representations. We need *statistical relational learning* (SRL) [5] or more generally a *probabilistic programming language* (PPL) [6] which combines logical representations, probabilistic reasoning and machine learning. Recent examples [12] show the power of such techniques in robotic settings.

Recent work [19] showed that probabilistic logic-based techniques are well-suited for the induction of *object affordances* in multi-object settings. Before [17], [20], [15], affordance models for single objects could be learned using the probabilistic formalism of Bayesian networks (BN) to provide a way to couple object *properties*, *actions* and *effects*. However, propositional BN models cannot handle interactions between multiple objects in a generic way. In contrast, the relational techniques of [19] can explicitly represent abstract definitions of spatial relations between objects and generically model rich interactions between multiple objects, e.g. for shelf sorting or packing items in a bag.

Relational affordance models can be used for imitation learning [20], where a robot computes the most likely demonstrated action based on observed effects. However, a typical skill consists of multiple steps, e.g. inserting a new object on a shelf may first require actions to make room for it. Imitation of such sequences of actions requires an extension of the method for exploiting the affordance models. That is, first one needs to compute the sequence of actions most likely demonstrated to the robot, and then devise a sequential plan that can imitate the (effects of the) demonstration.

A. Affordance-based Models

Affordances are a concept introduced by J. J. Gibson [9] that can be used to model the robot-world interaction by capturing *action opportunities* to structure the environment. They define the relationships between the robot and the environment through the robot's available sensing and motor capabilities [17], [20]. They model the correlations between the set of *object properties* detected by the sensors: $O = \{o_1, \ldots, o_n\}$, the repertoire of *actions* available to the robot, $A = \{a_1, \ldots, a_n\}$, and the *effects* of performing those actions on the objects: $E = \{e_1, \ldots, e_n\}$ as detected by the sensors as changes in object features. The model is typically learned by experimenting with motor actions on the available objects [17], [20], [15]. Using the relationships between O, A, and E, one unknown variable can be inferred given the others. We will use the affordance model as in [19] for action prediction, i.e. computing the *maximum a posteriori probability* (MAP) estimate: $\arg\max_A P(A|O,E) = \arg\max_A \frac{P(A,O,E)}{P(O,E)}$, for the observed values of O and E. Fig. 1 shows a generic affordance model.



Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

Fig. 1. Affordances as relations between (A)ctions, (O)bjects and (E)ffects, used for solving different tasks: prediction; action or object selection [17].

B. Problem Statement and Approach

Our previous work on relational affordance models [19] tackled the task of single action prediction. This paper tackles

Bogdan Moldovan is with the Department of Computer Science, Katholieke Universiteit Leuven, Belgium

Plinio Moreno is with the Electrical & Computer Engineering Department, Instituto Superior Técnico, Portugal

Martijn van Otterlo is with the AI Department, Radboud University Nijmegen, The Netherlands

Bogdan Moldovan is supported by IWT (agentschap voor Innovatie door Wetenschap en Technologie). This work is supported by the European Community's 7th Framework Programme, grant agreement First-MM-248258, and POETICON++, STREP Project ICT-288382.

multi-step skills, referred to as *high level concepts*. For example, a robot opens a cupboard by executing a sequence of basic actions: grasping, then pulling, then releasing the handle. This amounts to an incremental (or, developmental) way of acquiring new skills on top of old ones; in this case building high-level concepts on top of single-step affordance models. We use our previous work [19] as a building block for recognizing demonstrated actions, and extend our model to multi-step planning for replicating desired effects.

Our problem setting is: *given* i) demonstrated high-level concepts (sequences of basic actions), implicitly observed through observed object (interaction) properties and observed changes ii) a previously learned probabilistic relational affordance model for a pre-defined set of actions, and iii) an initial object configuration, and a final configuration obtained through demonstration and given as a set of effects, *find* iv) the most likely sequence of basic actions composing each concept and v) a sequence of concepts (possibly different than in iii) to replicate the final effects of the demonstration.

Our problem domain is a *table-top scenario* (see Fig. 2), with some initial objects, a desired final configuration, and demonstrations of high-level concepts. For example, in Fig. 2, to accomplish the task of placing the magenta object between the green and yellow objects the robot first needs to make space (concept composed of pushing the yellow object to the left and green to the right), then to place the magenta object in the space created (composed of two basic actions). Here probabilistic models are needed to deal with uncertainty of the vision and manipulation parts, relational models to deal with the (spatial) interactions between all objects and the robot, and sequential models to deal with the multiple actions needed to achieve arbitrary object placement goals.

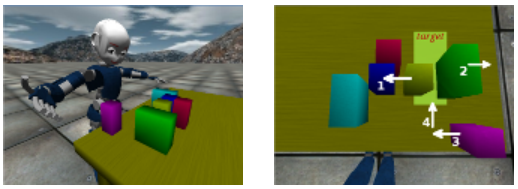


Fig. 2. Scenario with action sequence for object placement.

C. Contributions and Outline

The main contribution of this paper is extending the relational affordance model of [19] for an object placement scenario. This includes the extension of the relational domain for the robot to learn and model high-level concepts. SRL is a key element for the generalization over different object types and configurations, and action effects. For planning, we employ Markov decision processes (MDP) [29], and we describe an extension of affordance models towards relational MDPs [27] which facilitates replicating a demonstrated behavior in terms of desired effects. All aspects would be effectively impossible in the usual setting of affordances learned using BNs as in [17], [20], [15]. The experiments show encouraging results for the object placement task.

Section II presents related work and Section III the robot setting. Section IV reviews our previous work on relational affordance models. Section V describes our main contribution: the extension of relational affordance models for high-level concepts, and the extension towards a relational MDP model for our object ordering task. Section VI presents experimental results and we conclude in Section VII.

II. Related Work

Affordances, introduced by J. J. Gibson [9], are used to model world-robot interaction, especially in the context of imitation learning [17], [20], [15]. Relational affordance models for a multiple object setting were introduced [19] based on probabilistic relational models [5], [6]. Related work on planning object manipulation in a relational domain is [16], [25], based on noisy indeterministic deictic rules. Tackling a similar task, planning push actions for object placement on a cluttered table surface, is performed in simulation [4] and with a PR-2 robot [7]. However, the object interactions are determined by a dynamics simulator based on the object physical properties, not taking uncertainty into account. Other related work in decision-theoretic planning tackles similar problem domains [27], but usually works with predefined and full action models, whereas we work with learned affordance models. Related work into detecting [14] and manipulating [1], [10], [13] objects in cluttered environments is usually concerned with detecting the objects and planning the hand and arm trajectory [10], [13] for a grasp of the objects without collision with other objects.

III. Basic Skills of the Robot

We use a simulated *iCub robot* [18] (see accompanying video) with two head cameras (results transfer to the real platform similar to [19]). We only use one arm. Software modules provide: (i) motion control to reach a target position [22], (ii) image segmentation [2] and (iii) stereo triangulation. Basic skills of the robot include motor skills for actions and perceptual skills to measure object features and effects.

The robot has three basic core motor actions: `tap` (right-to-left hand movement), `push` (away movement) and `grasp` (pick-up, move to right and away, then release), with parameters adjusted after self-experience [19], [20]. Actions move the hand over a preprogrammed orientation and distance by a minimum-jerk Cartesian controller which reaches a position as close as possible to target considering the kinematic constraints of the robot (i.e., joint limits, damage avoidance, hand orientation [22]). Action execution is preprogrammed due to simulator limitations and iCub hand complexity. The robot's perceptual skills include color segmentation and 3D object localization as in [19]. Effects are measured as differences in object attributes before and after the action.

The robot receives a relational affordance model of two-object interactions, learned by exploration [19], and provided as a PPL program. The model has an object property `shape`, and two relational properties: the `relative distance` and `orientation` between pairs of objects, and the effects `displacement` and `orientation changes` for each object and `contact` between objects (computed from the intersection between the segmented regions in both camera images).

IV. Relational Affordance Models

We first briefly describe relational affordance models [19].

A. Probabilistic Programming Languages

For modelling object interactions in a multi-object setting, we need first-order logic to capture and generalize over the (spatial) relations in the domain, probabilistic information to deal with perception and action uncertainty, and learning to induce the affordance models from interaction with the environment. A *probabilistic programming language* (PPL) is a programming language designed to efficiently describe and reason with probabilistic relational models. We use the state-of-the-art PPL ProbLog [6], a probabilistic extension of the well-known Prolog logic programming language, with a causal-probabilistic logic (CP-logic)[28] style syntax.

A probabilistic relational model in ProbLog is described by a program consisting of a set of clauses defining probabilistic facts and logical rules. A CP-logic clause is a statement $\forall x(p_1 : A_1) \vee ... \vee (p_n : A_n) \leftarrow \phi$, where $\sum_{i=1}^{n} p_i = 1$, ϕ is a universally quantified first-order formula of some tuple of variables x, and A_i are atoms containing variables in x, such that for each x, ϕ causes at most one of the A_i to become true; A_i becomes true with probability p_i. [28] If $x = \emptyset$ the clause is ground, and assigning values to the variables in x is called grounding. A logical rule is a deterministic clause, ϕ causes some atom A with probability one, $(1 : A) \leftarrow \phi$ [28]. In ProbLog-code style, to model that any object can be a cylinder or a cube, one can write:
$\frac{1}{2} :: \texttt{shape}(\texttt{Obj}, \texttt{cube}); \frac{1}{2} :: \texttt{shape}(\texttt{Obj}, \texttt{cyl}) \leftarrow \texttt{obj}(\texttt{Obj})$.
$\texttt{Obj}$ is universally quantified over the set of all objects.

A ProbLog program $\mathcal{T} = \{p_1 : c_1, ..., p_n : c_n\}$, with clauses c_i that can be probabilistic facts with probability p_i or logical rules (i.e., $p_i = 1$), then defines a probability distribution over logic programs $L \subseteq L_{\mathcal{T}} = \{c_1, ..., c_n\}$ as $P(L|\mathcal{T}) = \prod_{c_i \in L} p_i \prod_{c_i \in L_{\mathcal{T}} \setminus L} (1 - p_i)$. Once the program is defined, several inference methods can be used for computing the probabilities of a query. This means asking for the success probability $P(q|\mathcal{T})$ of a query q, i.e., the probability that q has a proof given the distribution over logic programs[6].

The robot first explores a single object environment. It learns an affordance model as a BN, which as shown in [19] can be transformed in a ProbLog program with predicates modeling single object O, A and E (e.g., $\texttt{dispMag}(\texttt{Obj}, 1)$).

B. Multiple Object Relational Affordances

For multi-object scenes we introduce, in addition to the predicates modeling single object affordances, the following (two-object) spatial relations: relative distance, relative orientation and contact between objects. To learn a relational affordance model, the robot explores interactions in a two object environment (composed of a main object the robot acts upon, and a secondary one with which the main one interacts) and builds a non-grounded Bayesian Network (BN) introducing variables for objects, as illustrated in Fig. 3.

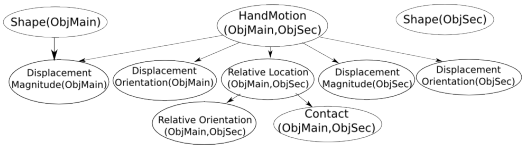


Fig. 3. Non-grounded BN for two-object interaction [19].

This BN, with learned parameters, can now be modelled using ProbLog. An example relation between displacement magnitude of the main object, hand motion and relative (Euclidean) distance between main and secondary object is:

$$0.03 :: \texttt{dispMag}(\texttt{ObjM}, 1); 0.22 :: \texttt{dispMag}(\texttt{ObjM}, 2);$$
$$0.25 :: \texttt{dispMag}(\texttt{ObjM}, 3); 0.5 :: \texttt{dispMag}(\texttt{ObjM}, 4)$$
$$\leftarrow \texttt{handMotion}(\texttt{ObjM}, \texttt{ObjS}, \texttt{push}),$$
$$\texttt{initRelDist}(\texttt{ObjM}, \texttt{ObjS}, 4).$$

Additional logical rules are added for defining background knowledge, which the robot would take longer to learn by itself, and for enforcing constraints (e.g., if there's contact between two objects don't attempt a grasp). As shown in [19], this relational affordance model can then be used successfully to model settings with an arbitrary number of objects. Interactions between three or more objects need not be explicitly considered because the relevant dependencies can be captured by pairwise interactions, and any additional influence of these interactions is negligible [19]. By representing only the inherent structure, the relational model only requires a small number of parameters as opposed to BNs. The complexity of applying any of these two approaches grows with the number of objects and actions.

V. High Level Concepts and Planning

In this section we describe our main contribution: the extension of the relational affordance model with the use of high level conceptual actions in order to handle a sequential object placement task. We assume a table-top setting with several initial objects, and the rest to be placed. The robot can observe the initial (properties of) objects on the table: O_0. The robot will be provided through demonstration with a set of high level concepts. Each concept C is composed of a sequence of the basic actions (push, tap, grasp) represented in the affordance model: $C = \{A_1, A_2, ...\}$. These concepts can either manipulate some objects on the table, or choose a new object, insert it on the table at a default location, and then continue with object manipulation. The robot will be provided with a goal, a final configuration of objects to achieve, which is the set of final object properties O_F and final effects E_F. Achieving the goal entails two tasks: learning and modelling of high level concepts, and building a plan using these concepts. Both extensions expand our initial relational representation and through the use of variables are able to generalise over an arbitrary number of objects in the scene, which would need tailored modelling otherwise.

A. Learning high level concepts

We want to provide the robot with high level skills based on its low-level motor capabilities. We demonstrate a set of five labelled high level concepts the robot has to use for achieving the goal, namely: *nextTo, moveAround, inaRow, makeSpace, moveAway*, illustrated in Fig. 4. Three of the

concepts (*nextTo, moveAround, inaRow*) first introduce a new object on the table and then manipulate it, while two of them (*makeSpace, moveAway*) manipulate the objects already on the table in order to create space to insert a new object.

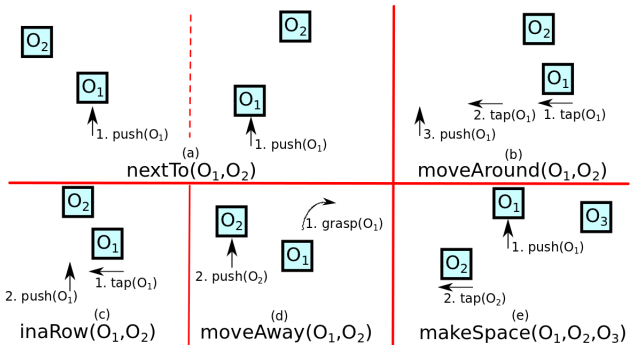


Fig. 4. The high-level concepts: a) two different instances of nextTo, b) moveAround, c) inaRow, d) moveAway, e) makeSpace.

In order for the robot to use the concepts for planning in a multi-object environment, it needs first to perform three tasks: (i) recognise the basic actions that compose the demonstrated concepts, (ii) model for the objects taking part in the concepts the state transition rules in terms of initial O and final O and E, and (iii) model the interactions between objects involved in a concept and those not taking part in it with the aid of the previous relational affordance model.

1) Recognising basic actions: For the robot to execute a concept, it needs to map it to a sequence of basic actions from its own action repertoire (which can be different than that of the demonstration). We demonstrate each labelled concept N times and assume the robot is able to discretise each concept into its TS different action timesteps (e.g., by observing the timestep boundary when there are several consecutive identical video frames). For each timestep $t \in [1, TS]$ over the N demonstrations of the same concept, the robot can detect the object properties: ${O_t}^1, \ldots, {O_t}^N$, and effects: ${E_t}^1, \ldots, {E_t}^N$. To imitate the effects of the timestep with its own basic actions, the robot needs to calculate the MAP estimate: $\arg\max_{A_t} P(A_t|{O_t}^1, {E_t}^1, \ldots, {O_t}^N, {E_t}^N)$. Given that object types and specific object locations in demonstrations are independent and each action is equally likely, this is equivalent to: $\arg\max_{A_t} P(A_t|{O_t}^1, {E_t}^1) * \ldots * P(A_t|{O_t}^N, {E_t}^N)$, where each of the terms of the product can be computed using the relational affordance model of [19]. For example, in the 5 demonstrations of the *moveAround* concept (Fig. 4(b)), in the first timestep O_1 moves to the left, so the predicted most likely action is a tap. After performing the MAP estimate for the TS timesteps, the robot can now perform the concept as a sequence of actions $\{A_1, \ldots, A_{TS}\}$ in its own action space.

2) Concept state transition rules: To use concepts for planning, the robot needs to be able to infer the outcome of executing a concept C in a particular object configuration or state. This is equivalent to defining outcome probabilities for each C in every state, namely defining a transition model $T(S_c, C, S_{c+1})$ for reaching state S_{c+1} from S_c. In the context of our object arrangement by use of relational affordances, a state S_c corresponds with the observed object properties together with the set of effects obtained after executing the previous concept (e.g., contact between objects). To reduce the size of the state transition model we use the fact that a concept should only be executed when certain conditions hold in a state. For a concept demonstrated N times, we consider the preconditions to be the set of O which have the same values between the N demonstrations, which correspond to a set of states. For example, the *inaRow* concept (Fig. 4(c)) has as preconditions $\texttt{relDist}(\texttt{Obj1}, \texttt{Obj2}, 2)$ and $\texttt{relOri}(\texttt{Obj1}, \texttt{Obj2}, 3)$. The N demonstrations of C also produce the effects $E^1, ..., E^N$, thus defining a probability distribution over final concept states, which we can refer to as a distribution over postconditions. This modeling can be related to a probabilistic version of STRIPS representation [8]. Executing a concept when the prerequisites are not met results in an undefined state (to be penalized when performing planning). Alternative modeling of this noise outcome can be done with noisy indeterministic deictic rules [16], [21]. We can now build a probabilistic relational model of concept state transitions once we augment our previous predicates with a concept sequence number. For example, if the *nextTo* concept was demonstrated only on square (sq) main objects with any type of secondary objects which are a distance 2 away, and the effect was that with 80% probability their distance becomes 1 (20% stays the same):

$0.8 :: \texttt{relDist}(\texttt{Obj1}, \texttt{Obj2}, 1, \texttt{C2});$

$0.2 :: \texttt{relDist}(\texttt{Obj1}, \texttt{Obj2}, 2, \texttt{C2}) \leftarrow \texttt{C1} = \texttt{C2} - 1,$
$\texttt{C1} >= 0, \texttt{nextTo}(\texttt{Obj1}, \texttt{Obj2}, \texttt{C2}),$
$\texttt{shape}(\texttt{Obj1}, \texttt{sq}, \texttt{C1}), \texttt{relDist}(\texttt{Obj1}, \texttt{Obj2}, 2, \texttt{C1}).$

3) Considering general object interactions: In addition to concept transitions, we need to model the concept's object interactions. For example, executing a *nextTo* concept might cause the object being inserted to touch a third object on the table. With the use of SRL to generalise over objects, all these possible object interactions need not be explicitly defined but they can be modelled by the previous relational affordance model [19]. For this we further augment the affordance model in [19] with extra variables for timesteps, so object properties and actions at timestep t define a probability distribution over effects at timestep $t + 1$. Once the concept basic actions are recognised as in (1) we can define the actions to use for each timestep, making sure they apply to one object from the concept and one surrounding object, through the use of variables. For example for the *inaRow* concept which has a tap at step 1:

$\texttt{notinC}(\texttt{OtherObj}, \texttt{Obj1}, \texttt{Obj2}) \leftarrow \texttt{obj}(\texttt{OtherObj}),$
$\texttt{OtherObj} \neq \texttt{Obj1}, \texttt{OtherObj} \neq \texttt{Obj2}.$
$\texttt{tap}(\texttt{Obj1}, \texttt{OtherObj}, \texttt{C}, 1) \leftarrow \texttt{inaRow}(\texttt{Obj1}, \texttt{Obj2}, \texttt{C}),$
$\texttt{notinC}(\texttt{OtherObj}, \texttt{Obj1}, \texttt{Obj2}).$

B. Planning using concepts

We now have a full relational model of object interactions due to the execution of the demonstrated concepts and we

can tackle the object arrangement task. Given the set O_0 and the goal O_F, E_F, the robot needs to come up and execute a sequence of concepts $\{C_1, C_2, ...\}$ to achieve the goal. This can be reduced to a planning problem.

Planning and sequential decision problems are generally described using Markov decision processes (MDP). We want to use the generalisations over objects by the relational affordance and concept state transition models, so we use a *relational* MDP (RMDP) [27]. An RMDP is defined as a tuple $\langle P, A, D, T, R\rangle$ where in this case D is the domain of objects, A is the set of action predicates which is the set consisting of the demonstrated concepts (e.g., nextTo(O1,O2,C)), and P is the set of predicates derived from our probabilistic transition rules (e.g., relDist(O1,O2,2,T)). The state space S is defined as a subset of the set of first order interpretations of P over D. Our initial state T_0 is the set of initial object properties O_0. The current state S is represented by the grounded predicates corresponding to the observed object properties at that timestep and the detected effects achieved from executing the previous concept. Then the transition model is defined in a similar way as for an MDP: $T(s, a, s')$, and similarly the reward function $R(s)$. In our case the transition model represents the probabilistic transition rules derived from the demonstrated concepts together with the general object interactions model.

There are several ways to model and find a policy for an RMDP. We use DTProbLog [26], which augments ProbLog with a set of (non-ground) decision facts D of the form $? :: d$, and a set U of utility attributes of the form $u_i \rightarrow r_i$ with u_i a literal and r_i the reward for achieving u_i. A strategy σ is a function $D \rightarrow [0, 1]$, mapping a decision fact to the probability that the agent assigns to it [26]. As explained in more detail in [26], the total utility of a strategy σ for a DTProbLog program DT is: $Util(\sigma, DT) = \sum_{a_i \in U} Util(a_i|\sigma(DT)) = \sum_{a_i \in U} r_i * P(u_i|\sigma(DT))$. Computing the optimal strategy resumes to calculating $\arg\max_\sigma Util(\sigma, DT)$ and algorithms for finding the optimal strategy are provided by DTProbLog [26]. Note that the contraints introduced through logical rules in the program as in [19] will hold and this will restrict the amount of viable actions (e.g., don't grasp two object in contact). We extend our previous relational model by changing the concept predicates into decision facts: $? :: nextTo(O1, O2, C)$ (while making sure by the use of logical rules that at each timestep only one concept can be picked), and defining a reward function given the effects according to our RMDP (e.g., $relDist(o1, o2, 2, 1) \rightarrow 10$). Then DTProbLog can infer the optimal concept to execute.

VI. Evaluation and Results

We want to show that probabilistic relational affordance models of object interactions extended to use a set of prior demonstrated high level concepts, composed of a sequence of basic actions, can be used successfully in a planning setting. We want to show that the robot can reach a predefined goal, given as a set of desired effect features, through executing a sequence of learned concepts. We use a table-top setting with objects where a sequence of objects needs to be inserted. The robot is given a final configuration of objects, in the form of grounded relational effects (e.g., relative distance and orientation between pairs of objects). To achieve the goal, the robot will need to infer and execute a sequence of the concepts, illustrated in Fig. 4.

In our setting, using one hand and 5 demonstrated concepts with their respective preconditions, only a limited set of effects can be achieved. To test the sequential decision making, we need first to generate a set of achievable effects. We consider an initial state with 2 objects already placed and investigate sequences of 3 concepts (i.e., sequences of 5 to 7 basic actions). These constraints induce only 10 different high-level concept sequences of length 3, where the position of the initial objects on the table has to correspond with the prerequisites of the first action in the sequence. We execute each of these 10 possible sequences twice, with objects of random shape (unless the shape requirement conflicts with another object position) and generate the 20 effect datasets for our experiments. The robot will need to reach these final configurations, possibly through the use of different concept sequences than the ones that were used to obtain them.

We equip the robot with the model in [19] to enable it to do action prediction given the set of desired effects and object properties. These properties are perceived using color segmentation and 3D object localization, and are: shape, discretised relative distance (4 clusters separated at $6cm$, $10cm$, $14cm$) and orientation (8 clusters of $45\,^\circ$) between each pair of objects. Effects include discretised object displacement (4 clusters separated at $1cm$, $3cm$, $5cm$), discretised displacement orientation and contact between 2 objects, which relies on the area of intersection between the convex hull of the 2 segmented areas.

In addition, we demonstrate to the robot the concepts in Fig. 4. Each concept is demonstrated 5 times, and we use the affordance model to predict the basic actions that constitute the concept. The 5 concepts contain together 10 basic actions, out of which 9 were predicted correctly using the method in Section V(A) (a *push* action was predicted instead of a grasp in the *moveAway* concept; *grasp* was second most likely). For planning we will use these predicted actions since they represent the means to achieve the effects.

We use the DTProbLog program described in Section V. At the concept level, we will only use the relative distance and orientation between pairs of objects from the affordance model. The model can be extended to the other affordance object properties and effects, and there is a trade-off between increased model accuracy and inference time. Our RMDP will have a finite horizon of 1 (concept), with preset rewards (which in the future can also be learnt from a set of trials). We heavily penalise ending up in a bad state. We give a high reward for achieving a final effect within the horizon, and a medium reward for achieving an effect that corresponds with an adjacent cluster to the desired one. There are no rewards for placing objects in the remaining clusters of orientations or distances, since it will take extra actions to achieve the desired outcome (in some cases it might be impossible given our limited concept set).

We ran the 20 experiments, each with one of the initial 2 object settings and final sets of effects. Each step the robot infers the best concept to use, executes it, and uses the new state information to re-plan for the next step. We stopped after the concept that introduced the fourth object on the table. Produced plans had length 2 to 4 (original sequence of actions that generated those possible final states all had length 3). Fig. 5 shows a sequence of executed concepts.

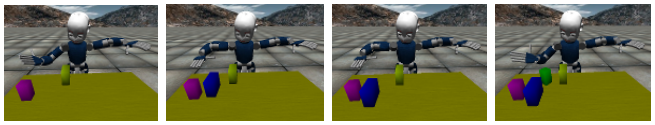


Fig. 5. Sequence of executed concepts: i) initial setting, ii) after a nextTo, iii) after a makeSpace, iv) final state after a moveAround

We compared the desired and obtained configuration after executing the sequence of concepts. Since the relative distances that were input into the model were discretised in clusters of $4cm$, and the average error from the image segmentation was a bit over $1cm$, we considered an object placed successfully if its final location, as detected by the vision system, was within $5cm$ of the target location (e.g., on average a variation of about $\pm 3.5cm$ in both detected x and y-axis positions). Table I shows that in 11 out of 20 cases all objects were placed successfully, and in other cases, either 1 or 2 out of the 4 objects were misplaced.

TABLE I

EXPERIMENT RESULTS: NUMBER OF MISPLACED OBJECTS (OUT OF A MAXIMUM OF 4) DURING THE 20 TEST RUNS

Objects Misplaced	Number of runs	(Runs) Percentage
0	11	55%
1	4	20%
2	5	25%
3	0	0%
4	0	0%

The original affordance model was 58% accurate [19] for a single-action prediction, so in comparison we achieved only a slightly worse accuracy (55%) for whole sequences of 4 to 6 actions. Although replanning at every step can handle some uncertainty in object interaction in our setting, the limited amount of actions in our action set limits the effects that can be achieved (e.g., an object cannot be brought closer to the robot). A more varied action set would improve the results. As noted, planning accuracy increases by modelling the relations between the other affordance features. Out of the 11 successful placements the robot came up with exactly the same plan with which we generated the desired outcome only 3 times. In all the other cases the system came up with a different plan that produced the same effects.

VII. CONCLUSION AND FUTURE WORK

We have presented an application of relational affordances models extended to use a set of demonstrated high level concepts composed of a sequence of basic actions in a planning setting. We showed that the extended relational model can be used successfully for the task of object arrangement.

Future work will first include using human demonstrations, different high level concepts and transferring the model to an iCub and a real-world multiple object setting. We plan to go towards more complex environments and planning tasks.

REFERENCES

[1] D. Berenson and S. S. Srinivasa. Grasp synthesis in cluttered environments for dexterous hands. In *Humanoids*, 2008.

[2] C. M. Christoudias, B. Georgescu, P. Meer, and C. M. Georgescu. Synergism in low level vision. In *ICPR*, 2002.

[3] S. Coradeschi and A. Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2-3):85–96, 2003.

[4] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman. Push planning for object placement on cluttered table surfaces. In *IROS*, 2011.

[5] L. De Raedt. *Logical and Relational Learning*. Springer, 2008.

[6] L. De Raedt, A. Kimmig, and H. Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, 2007.

[7] V. Emeli, C.C. Kemp, and M. Stilman. Push planning for object placement in clutter using the PR-2. In *IROS PR2 Workshop*, 2011.

[8] R. E. Fikes and N. Nilsson. STRIPS: A new approach to the application theorem proving to problem solving. *Artificial Intelligence*, 5(2):189–208, 1971.

[9] J. J. Gibson. *The Ecologial Approach to visual perception*. Boston: Houghton Mifflin, 1979.

[10] M. Gienger, M. Toussaint, and C. Goerick. Task maps in humanoid robot manipulation. In *IROS*, 2008.

[11] J. Hertzberg and R. Chatila. AI reasoning methods for robotics. In *Handbook of Robotics*, pages 207–223. Springer, 2008.

[12] D. Jain, L. Mösenlechner, and M. Beetz. Equipping robot control programs with first-order probabilistic reasoning capabilities. In *ICRA*, pages 3626–3631, 2009.

[13] N. Jetchev and M. Toussaint. Trajectory prediction in cluttered voxel environments. In *ICRA*, 2010.

[14] D. Kragic, M. Björkman, H. I. Christensen, and J. O. Eklundh. Vision for robotic object manipulation in domestic settings. *Robotics and Autonomous Systems*, 52(1):85–100, 2005.

[15] V. Krunic, G. Salvi, A. Bernardino, L. Montesano, and J. Santos-Victor. Affordance based word-to-meaning association. In *ICRA*, 2009.

[16] T. Lang and M. Toussaint. Planning with noisy probabilistic relational rules. *J. Artif. Intell. Res. (JAIR)*, 39:1–49, 2010.

[17] M. Lopes, F. S. Melo, and L. Montesano. Affordance-based imitation learning in robots. In *IROS*, pages 1015–1021, 2007.

[18] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori. The iCub humanoid robot: an open platform for research in embodied cognition. In *PerMIS*, 2008.

[19] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt. Learning relational affordance models for robots in multi-object manipulation tasks. In *ICRA*, 2012.

[20] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory-motor coordination to imitation. *IEEE Transactions on Robotics*, 24:15–26, 2008.

[21] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling. Learning symbolic models of stochastic domains. *JAIR*, 29:309–352, 2007.

[22] U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini. An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *IROS*, 2010.

[23] F. Stulp and M. Beetz. Combining declarative, procedural, and predictive knowledge to generate, execute, optimize robot plans. *Robotics and Autonomous Systems*, 56:967–979, 2008.

[24] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[25] M. Toussaint, N. Plath, T. Lang, and N. Jetchev. Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference. In *ICRA*, 2010.

[26] G. Van den Broeck, I. Thon, M. van Otterlo, and L. De Raedt. DTProblog: A decision-theoretic probabilistic prolog. In *AAAI*, 2010.

[27] M. van Otterlo. *The Logic of Adaptive Behavior*. IOS Press, Amsterdam, The Netherlands, 2009.

[28] J. Vennekens, M. Denecker, and M. Bruynooghe. CP-logic: A language of causal probabilistic events and its relation to logic programming. *TPLP*, 9(3):245–308, 2009.

[29] M. A. Wiering and M. van Otterlo. *Reinforcement Learning: State-of-the-Art*. Springer, 2012.