# Optimizing Motion Primitives to Make Symbolic Models More Predictive

Andreas Orthey, Marc Toussaint and Nikolay Jetchev

***Abstract*— Solving complex robot manipulation tasks requires to combine motion generation on the geometric level with planning on a symbolic level. On both levels robotics research has developed a variety of mature methodologies, including geometric motion planning and motion primitive learning on the motor level as well as logic reasoning and relational Reinforcement Learning methods on the symbolic level. However, their robust integration remains a great challenge. In this paper we approach one aspect of this integration by optimizing the motion primitives on the geometric level to be as consistent as possible with their symbolic predictions. The so optimized motion primitives increase the probability of a "successful" motion—meaning that the symbolic prediction was indeed achieved. Conversely, using these optimized motion primitives to collect new data about the effects of actions the learnt symbolic rules becomes more predictive and deterministic.**

## I. Introduction

Reinforcement Learning (RL) is one of the most promising approaches towards autonomously learning agents. In the field of robotics, RL has successfully been applied on both, the motor as well as the symbolic level, which is necessary because in natural environments an agent has to reason on a geometric as well as on an abstract level to learn and plan sequences of motor primitives [1]. Concerning the first, *motion primitive learning* [2] has been rather successful in dealing with the high-dimensional state spaces of general dynamic control problems and allowing the autonomous learning of robot motor skills that would otherwise be hard to program explicitly by an engineer. Concerning symbolic manipulation, recent advances in *relational RL* [3] have successfully tackled the challenges of the exponential state space spanned by the relations and properties of objects, leading to efficient methods for learning and generalizing of relational rules, using them for planning, and driving exploration in relational domains to speed up learning [4]. These two "extremes" – motion primitive learning on the motor level and relational Reinforcement Learning on the symbolic level – would need to be eventually combined in order to yield an integrated approach to Reinforcement Learning in robotics.

The present paper addresses one issue in the integration of these extremes. Standard relational RL assumes the existance of a fixed set of actions, respectively motion primitives, which it uses to explore the domain and learn probabilistic rules. This choice of motion primitives therefore determines the symbolic data that is being collected. For instance, using mediocre motion primitives for pick&place will lead to experiences where objects fail to be lifted or drop, and the learned rules will be more stochastic. In this paper we propose to use the learned relational rules as a criterion to re-train motion primitives so that the rules become more predictive. This implies an iterative bootstrapping process, where we start with an initial set of motion primitives, use them to collect symbolic data and learn relational rules, then use the learned rules to re-train the motion primitives, and so forth. The idea is that the learning on the symbolic and motion primitive level should inform each other, leading to a set of motion primitives that is most coherent with symbolic predictive models.

In Section II we will discuss related work before, in Section III we introduce relevant background on relational RL as well as the motion planning methods that we use to generate motion primitives. Section IV describes our contributions: the specific parameterization of our motion primitives, an objective function measuring how the primitives fit to the learned relational rules, and the optimization procedure. Section V reports on our evaluations, where we consider pick and place scenarios that proved challenging to hand tuned parameters and show how our optimization leads to better motion primitives.

## II. Related Work

Large-scale demonstrations of complex manipulation tasks, integrating symbolic reasoning with motion planning, has been the topic of a number of projects. Beetz et al. [5] [6] considered an assistive kitchen scenario and demonstrated tasks like cooking a pancake. Asfour et al. [7] considered similarly complex kitchen tasks that require sequential manipulation. These approaches exploit reasoning on a symbolic level to different extends, but the motion primitives are designed by hand instead of implied by and optimized to be consistent with the symbolic action model. Our work eventually aims to make such integrated systems more robust by iteratively optimizing the motion and symbolic levels for consistency.

Learning motion primitives has been the focus of a very successful line of research. For instance, Ijspeert et al. [2] and Peters et al. [8] have developed a series of powerful learning methods that optimize parameterized motion primitives to achieve various motor skills. Examples for motion primitive parameterizations are using linear combinations of basis functions to specify a feedback regulator [8], or via a

A. Orthey is with CNRS/LAAS, Université de Toulouse UPS, INSA, INP, ISAE, F-31077 Toulouse, France. aorthey@laas.fr

M. Toussaint is with the Machine Learning and Robotics Lab, University Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany. marc.toussaint@informatik.uni-stuttgart.de

N. Jetchev is with the Machine Learning and Robotics Lab, FU Berlin, Arnimallee 7, 14195 Berlin, Germany. nikolay.jetchev@fu-berlin.de

reference trajectory like in Dynamic Motion Primitives [2]. Our work applies motion primitive optimization for a specific high-level objective function, namely that the generated motions should generate most likely the symbolic effects of learned symbolic rules. A further, technical difference to most previous work on motion primitive optimization is that we define motion primitives indirectly as the result of a (fast) local motion planner. The reason for this is that we take hand-tuned trajectory cost functions and task spaces for grasping and placing as starting point for our approach, ensuring the same type of generalization to novel situations (e.g. obstacle configurations) and optimality of the generated motion. But the approach could also be applied on other types of motion primitive parameterizations.

Learning on the symbolic level in the context of robotic manipulation tasks is still a great challenge. Generally, previous work on relational Reinforcement Learning always presumed a given set of symbolic actions [9]. Our work specifically builds on the method of [10] to learn probabilistic relational rules from symbolic data. However, to our knowledge this is the first work trying to consider optimizing the grounding of actions in relational RL as robot motion primitives.

## III. Background

### A. Learning relational rules

In this paper we will optimize motion primitives essentially to become "consistent" with learned symbolic probabilistic rules. Consistency means that the symbolic effect predicted most likely by this rule should be made as likely as possible by the corresponding motion primitive. The symbolic probabilistic rules we consider are noisy indeterministic deictic (NID) rules [10]. Generally, a NID rule is given as

$$\text{action}(\mathcal{X}):\ \text{context}(\mathcal{X}) \quad \rightarrow \quad \begin{cases} p_1 & : & \text{outcome}_1(\mathcal{X}) \\ & \vdots & \\ p_m & : & \text{outcome}_m(\mathcal{X}) \\ p_0 & : & \text{noise} \end{cases}$$

which states that a certain *action*, applied on a set of objects $\mathcal{X}$ (formally a set of logic variables) in a certain *context* can have $m$ different outcomes, each with probability $p_m$ – or lead to a special outcome, the so-called *noise*, which includes rare and overly complex outcomes typical for noisy domains, which are not covered explicitly for compactness and generalization reasons. The context and outcomes are described as conjunctions of predicates of the objects $\mathcal{X}$. The formal details of such relational rules are not relevant for the remainder of this paper. Relevant here is that, *given a set of symbolic actions* and their respective motion primitives, we can sample concrete experience $\mathcal{D} = \{(s_t, a_t, s_{t+1})\}_t$, where $s_t, s_{t+1}$ are propositional state descriptions before and after the application of an action $a_t$.

Clearly, the symbolic data $\mathcal{D}$ depends on the implementation of the motion primitives which influence the probabilities of effects. This is where our work contributes: We aim to optimize the motion primitives such that the rules learned from the new data $\mathcal{D}$ will be more predictive (less entropic).

### B. Motion Planning

We will later define motion primitives as the result of a standard motion planner. Any motion planner that can address the following problem formulation can in principle be used. Let $x_t$ be the state of the system at time step $t$—we will always consider the dynamic case where $x_t = (q_t, \dot{q}_t) \in \mathbb{R}^n$ with $q_t$ being the robot joint angles at time step $t$. Consider the problem of minimizing (the expectation of) the cost

$$C(x_{0:T}, u_{0:T}) = \sum_{t=0}^{T} c_x(x_t) + c_u(u_t) \tag{1}$$

where $c_u$ describes costs for the control and $c_x$ describes task costs depending on the state. For our purposes we assume that the task costs are of the form $c_x(x_t) = \Phi_t(x_t)^\top \Phi_t(x_t)$ (lending to efficient Gauss-Newton type methods) where we can efficiently evaluate the *task vector* $\Phi_t(x) \in \mathbb{R}^m$ as well as its Jacobian $J_t(x) = \frac{\partial}{\partial x}\Phi(x)$ for any state $x$.

Any motion optimizer can be employed. We used the Approximate Inference Control framework, which translates this cost function to a graphical model and employed Gaussian message passing to find a MAP trajectory [11], which is very similar to differential dynamic programming [12] or iLQG [13].

## IV. Optimizing motion primitives

### A. Parameterization of motion primitives

In our approach, different motion primitives correspond to different cost functions $C(x_{0:T}, u_{0:T})$: grasping has its cost function, placing has another. Motion primitives are generated as the result of the motion planner, which comprises both, the optimal trajectory as well as the local linear quadratic regulator around this reference trajectory (which the above mentioned methods return or can easily be computed using the Riccati equation and the local linearization $J$ around the optimal trajectory).

When we want to optimize a motion primitive for a desired effect we therefore need to optimize its corresponding cost function $C$. That is, we parameterize this cost function and optimize these parameters w.r.t. a higher-level objective function which we describe in the next section.

We parameterize the cost function using task variables and weights as follows. The task vector will be composed of multiple *task variables* $\phi_i$,

$$\Phi_t(x) = \begin{pmatrix} \sqrt{\rho_1}\ (\phi_1(x) - y^\star_{1,t}) \\ \sqrt{\rho_2}\ (\phi_2(x) - y^\star_{2,t}) \\ \vdots \end{pmatrix} \tag{2}$$

where each $\phi_i : \mathbb{R}^n \rightarrow \mathbb{R}^{m_i}$ describes a task variable. For example, to control the position of the endeffector we can define a task variable *"reach target"* which consists of a mapping $\phi_i$ given by the forward kinematics. Each task variable is accompanied by a task target $y^\star_{i,t}$ which describes the goal of the task at each time step $t$. In the case of the *"reach target"*, the task target is the desired position of the endeffector. Eventually, every variable is weighted by a

parameter $\rho_{i,t}$, which defines its precision (inverse variance) at each time step $t$. Tuning the targets $y^\star_{i,t}$ and precisions $\rho_{i,t}$ allows us to flexible generate motions that blend various tasks as required.

The space of possible cost functions (and motion primitives) is therefore spanned by the space of possible task variables, its targets $y^\star_{i,t}$ and precisions $\rho_{i,t}$—a specific motion primitive is defined as the tuple $\mathcal{P} = (\phi_i, y^\star_{i,t}, \rho_{i,t})_{i=1}^m$. Note that precision parameters and targets are allowed to change over time – this provides additional flexibility, e.g. for defining in-between targets, which are only important at a specific timeframe.

In the following we will assume only the precision $\rho_{i,t}$ are subject to optimization. Optimizing the task mappings $\phi_i$ themselves is beyond the scope of the current paper and would imply to find new task spaces. (A conceivable approach would be to select from a very large variety of potentially task spaces, similar to feature selection.) The kind of task spaces we use are straight-forward: they include grasp center's position, the distance of finger tips to the object, a measure for the opposedness of fingers, as well as standard collision and limit avoidance task spaces. We also assume the targets $y^\star_{i,t}$ to be predefined. Extending our optimization to include them would be straight-forward. However, for the specific task spaces we considered, the targets are rather obvious: e.g. aligning the grasp center with the object center, keeping collisions and limit cost to zero, etc.

### B. Objective function

The motion primitives (i.e. the respective $\rho_{i,t}$) are optimized w.r.t. a higher-level objective function which evaluated whether the resulting motions really generate the symbolic effect that is predicted by learned NID rules. The prediction of a NID rule is the outcome with the highest probability. For example, assume that, using initially non-optimized motion primitives, we collected experience $\mathcal{D} = \{(s_t, a_t, s_{t+1})\}_t$ here $s_t, s_{t+1}, a_t$ are state and action symbols. From this experience the NID rule learner will learn probabilistic rules, for instance one for grasping:

$$\texttt{grasp}(X) : \texttt{on}(X,Y), \texttt{cylinder}(X), \texttt{table}(Y) \quad \rightarrow \quad \begin{cases} 0.3: & \texttt{inhand}(X), \neg\texttt{on}(X,Y) \\ 0.7: & \mathit{noise} \end{cases} \tag{3}$$

where we summarized all outcomes as noise, except the one with the highest probability – this is the desired outcome, which is considered as a success by the symbolic planner. The rule therefore expresses that only with 30% probability the grasping primitive was successful. Once we have learned this rule we can re-train the grasping primitive (optimizing the respective $\rho_{i,t}$) to increase the probability of the outcome $\texttt{inhand}(X) \wedge \neg\texttt{on}(X,Y)$.

More precisely, based on the learned rules we define the higher level objective function of a motion primitive as follows. We choose a fixed *training set* $\mathcal{S} = \{s_1, .., s_K\}$ of scenarios, where $s_i$ describes the full geometric state of a training scenario, that is, the pose of all objects and the initial robot. For each training scenario we invoke the motion planner $\mathcal{M}$ and obtain $K$ trajectories $\bar{x}_{1:K}$. For each trajectory

**Algorithm 1** Fitness function for motion primitive $\mathcal{P}_i$

**function** FITNESS($\mathcal{P}_i$)
  $\overline{x}_{1:K} \leftarrow \mathcal{M}(S_{1:K}, \mathcal{P}_i)$ //evaluate $\mathcal{P}_i$ on $S_{1:K}$
  $\gamma \leftarrow \frac{1}{K} \sum_{k=1}^{K} g(\overline{x}_k)$ //average costs of trajectories
  **return** $\gamma$
**end function**

$\bar{x}_k$ we compute a number of indicators of its quality, namely 1) the success w.r.t. achieving the symbolic outcome of the respective rule, 2) whether the minimum collision distance throughout the trajectory is below a margin, 3) whether the minimum joint limit distance throughout the trajectory is below a margin, 4) the euclidean length of the trajectory, 5) the number of iterations needed by the motion planner. All these criteria are combined to define the average performance $g(\bar{x}_k)$ of a motion primitive on the training scenario set.

Note, to evaluate whether an object really is `inhand` after a motion primitive we instantiated the hand and object pose within the physical simulator, jiggled at the hand with a Brownian motion, and empirically observed whether the object fell out of hand. This seemed much more realistic than evaluating force closure or alike. Further, to evaluate whether an object really is on after a motion primitive, we measured the vertical displacement of their centers.

To formalize the above discussion, we define the objective function as

$$\mathcal{P}^\star = \arg\max_{\mathcal{P}} \gamma(\mathcal{P}) \tag{4}$$

whereby $\gamma$ constitutes the outcome of a fitness function, as displayed in algorithm 1. To find the best precisions $\rho_{i,t}$ to obtain $\mathcal{P}^\star$, we used a stochastic optimization method [14] to maximize it.

As a theoretical note, one might ask "When we've defined a higher-level objective function for the motion, why do we not optimize it directly with a motion planner and instead use it to optimize the parameters of a cost function $C$ that is passed on to the motion planner?" The clear answer is that the criteria of the higher-level objective are not in the form (1) with time-slice-wise and differentiable task costs which can efficiently be optimized. Instead the higher-level objectives are global properties of the trajectory and/or non-differentiable. In constrast, the specific form of our cost function, based on semantic task variables (like relative endeffector positions), ensures that we incorporate our expertise on what might be relevant task variables, that we generalize well, and that we can optimize them efficiently (during online execution).

### C. Iterative rule learning and motion primitive optimization

Once we have re-trained motion primitives they will of course lead to different statistics on the symbolic effects of actions. Therefore, using the new primitives we can collect new symbolic data and relearn the rules. In our experiments we will show how the outcome probabilities of the learnt rules change due to the optimization of the motion primitives.

## V. EXPERIMENTS

In this section we will demonstrate that our method can indeed lead to motion primitives that make symbolic state transitions more deterministic. We consider two symbolic actions, *GraspCylinder* and *PlaceCylinder*. Using non-optimized, manually tuned motion primitives to collect symbolic data we can learn corresponding NID rules for each action. We simplified each rule by assuming that all undesired outcomes are noise, and that there is one desired outcome probability, which we will call the *success rate*. To increase this *succes rate*, we optimize with respect to the motion primitive parameters $\rho$.

We conduct for each symbolic action two experiments, one reporting on the optimization on a set of training scenarios, the second reporting on the generalization of the learned motion primitive parameters on test scenarios. We constrained the range of the parameters $\rho_{i,t}$ to the rather large interval $[10^{-3}, 10^{9}]$ – reflecting our experience from manual tuning that precision parameters may differ by orders of magnitude. To facilitate optimization we transformed the parameters $\rho$ to a logarithmic scale, ensuring better covering of this wide interval of possible precisions.

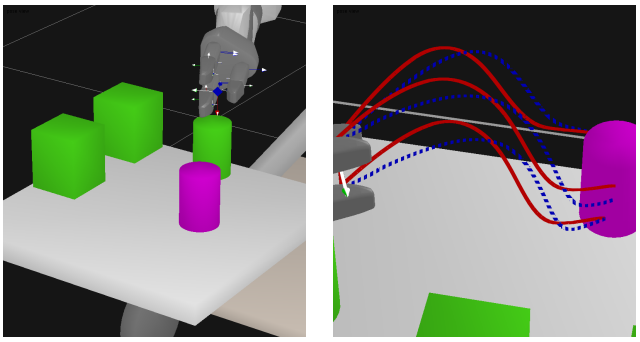
Concerning the stochastic optimization algorithm we tested both, CMA as well as a basic a monte carlo (MC) sampling technique, which simply samples a parameter vector $\alpha$ at random from the domain. We conducted each experiment on a simulated robotics platform, using the libraries *Open Dynamics Engine (ODE)*[1] for physical simulations and *SWIFT++*[2] for proximity measurements. Furthermore, we used the *SHARK*[3] library for the CMA stochastic optimization algorithm.

### A. The grasping action

*a) Optimization:* The symbolic grasping action will be optimized on a set $\mathcal{S}$ of 6 different training scenarios. In each scenario, the robot has to grasp a purple cylinder, as depicted in Figure 1. The choice of those scenarios was motivated by situations, which were difficult to perform with the non-optimized motion primitives. The optimization algorithms start with a set $\rho$ of random parameters and optimize them until a total of 2000 evaluations are performed. Because we do not have any guarantee on convergence to the global optimum we restart the optimization procedure 10 times.

The optimization process and variance over the optimization restarts can be seen in Figure 2. It shows in the upper graph the *success rate* $p_{grasp}$, for CMA and Monte Carlo. CMA slightly outperforms plain Monte Carlo optimization, leading more robustly to parameters with maximal success rate in all optimization restarts. The lower graph additionally shows the objective function $\gamma$ during optimization.

*b) Generalization:* We also want to evaluate whether the optimized motion primitives significantly improve the success rate on previously unseen test scenarios. We generate random test scenarios by adding uniform noise to the position of each object in each of the 6 test scenarios. During this generation of random test scenarios, we rejected configurations where objects touch each other or where objects fall from the table. Since the computational cost for testing are low, we generated a total of 1200 test scenarios for each noise level in $\{0, 1, 2, 3, 4, 5\}$, which denote the centimeters of random displacement of the object positions. With increased noise, the test scenarios deviate more from the training scenarios, as shown in Figure 3. If we start with zero noise, the best $\rho$ parameters reach a *success rate* of $p_{grasp} = 1.0$. Moving away from the training data decreases the performance of our optimized parameters, but they remain nevertheless significantly better than the non-optimized ones, and thus show the ability to generalize. The dashed blue line indicates the success rate of the non-optimized parameters on random scenarios (this is a baseline, generated by testing 1200 scenarios at high-noise level). The obtained $p_{grasp}$ outcome probabilities translate directly to the new NID rules for the optimized motion primitives, as depicted for two noise levels in Table I. The left column shows the outcome for high noise levels of the non-optimized motion primitive, which have a lower performance compared to the optimized primitives, shown in the right column.



(a) One fixed scenario.

(b) The grasping trajectories for one scenario: solid red line for the optimized parameters, dashed blue for the manually specified ones.

Fig. 1. The grasping experiment.

TABLE I

OBTAINED NID RULES FOR THE SYMBOLIC GRASPING ACTION.

| Noise | Symbolic NID Rule | |
|---|---|---|
| | $\mathcal{P}$ | $\mathcal{P}^\star$ |
| 0 | `GraspCylinder`(X): `on`(X,Y), `cylinder`(X), `table`(Y) $\rightarrow$ {0.3 `inhand`(X), ¬`on`(X,Y); 0.7 *noise*} | `GraspCylinder`(X): `on`(X,Y), `cylinder`(X), `table`(Y) $\rightarrow$ {**1.0** `inhand`(X), ¬`on`(X,Y); 0.0 *noise*} |
| 1.5 | `GraspCylinder`(X): `on`(X,Y), `cylinder`(X), `table`(Y) $\rightarrow$ {0.3 `inhand`(X), ¬`on`(X,Y); 0.7 *noise*} | `GraspCylinder`(X): `on`(X,Y), `cylinder`(X), `table`(Y) $\rightarrow$ {**0.94** `inhand`(X), ¬`on`(X,Y); 0.06 *noise*} |

[1] http://www.ode.org/

[2] http://gamma.cs.unc.edu/SWIFT++/

[3] http://sourceforge.net/projects/shark-project/

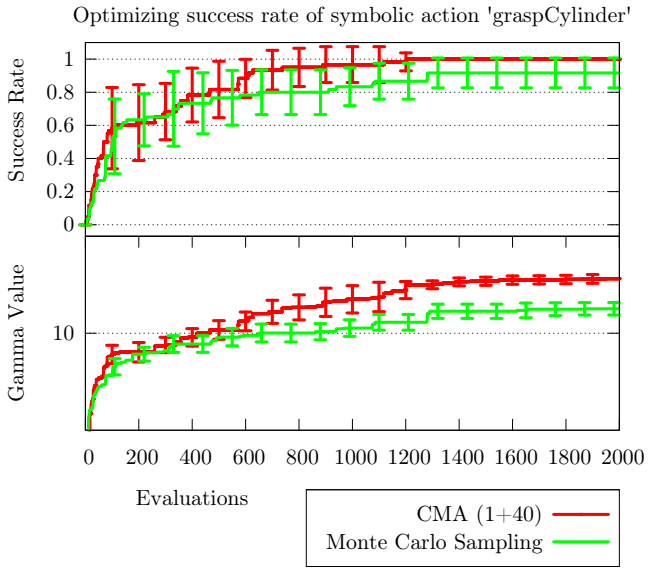



Fig. 2. Success rate and $\gamma$ for the grasping experiment, fixed scenario set.

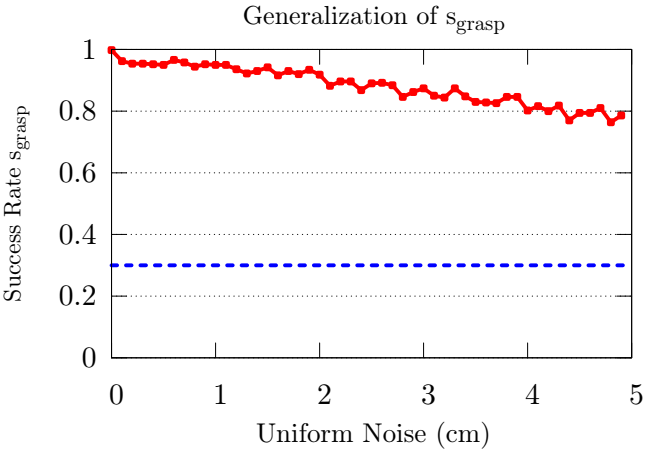
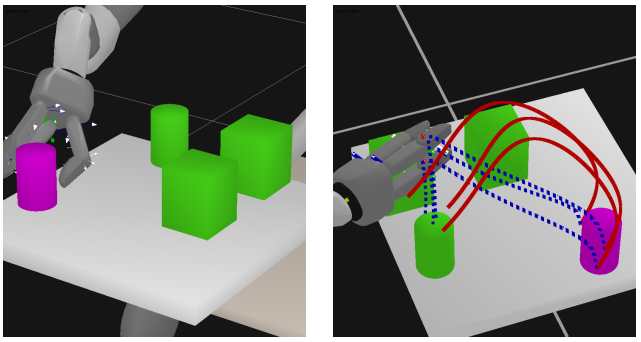



Fig. 3. Success rate for the grasping experiment, noisy test scenario set. Comparison between non optimized $\mathcal{P}$ (blue,dashed) and $\mathcal{P}^\star$ (red,solid).

### B. The Placing Action

*c) Optimization:* The second symbolic action which we optimized is the *PlaceCylinder* action. Likewise to the *GraspCylinder* action, we used $K = 6$ test scenarios for optimization. The start configurations for one of the test scenarios can be seen in Figure 4. They are similar to the *GraspCylinder* action, but with a different starting configuration, where the hand of the robot is grasping the purple cylinder. The goal of each scenario is to place the purple cylinder onto the green cylinder. Similar to the *GraspCylinder* action, we optimized the parameters by using CMA and MC sampling. Figure 5 shows the optimization process for both methods. Each function represents the average over 10 restarts, together with its mean estimator. The final trajectories, obtained by using the best optimized $\alpha$ parameters from CMA and the manually specified ones, are shown exemplarily in Figure 4(b).

*d) Generalization:* We again performed an experiment, which demonstrates how the optimized parameters for *PlaceCylinder* generalize to unseen scenarios. Figure 6 shows the performance of the *success rate* , if we add different uniform noise to the position of the objects. Each datapoint is obtained by using the average over 1200 scenario evaluations. The optimized parameters again retain a good generalization ability on test scenarios different to the training scenarios. Table II shows the corresponding NID rules, obtained at two different noise levels.



(a) One fixed scenario.

(b) The place trajectories for one scenario: solid red line for the optimized parameters, dashed blue for the manually specified ones.

Fig. 4. The placing experiment.

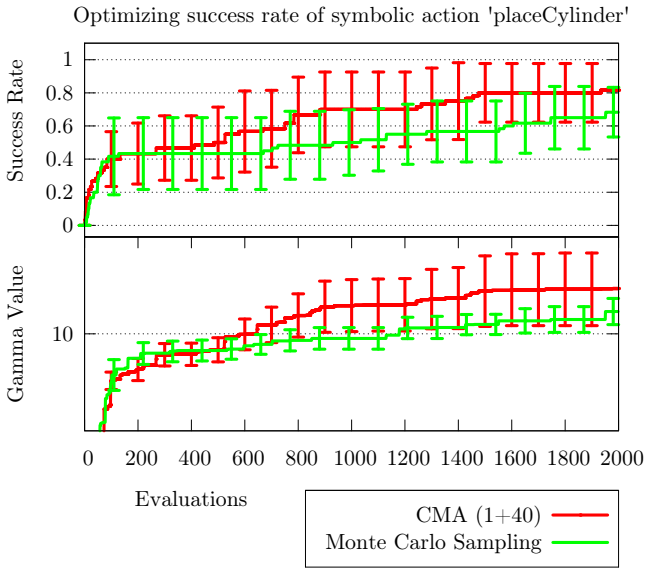



Fig. 5. Success rate and $\gamma$ for the placing experiment, fixed scenario set.

## VI. Conclusions

In this paper we considered motion primitive optimization with the objective to increase the predictability of the corresponding symbolic rules. Generally this means to optimize the primitives to maximize the probability of the desired symbolic state transition—for instance, in our demonstrations the grasp motion primitive was trained to ensure the symbolic prediction of $\texttt{inhand}(X)$, as was the place motion primitive trained to fulfil the prediction of $\texttt{on}(X, Y)$. Closing the loop, the optimized motion primitives imply, when used to continue exploration of the environment, new statistics of

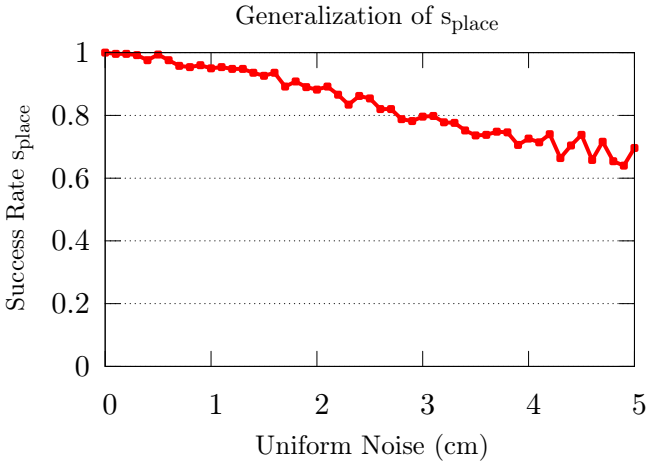



Fig. 6. Success rate for the placing experiment, noisy test scenario set.

TABLE II

COMPARISON OF THE AVERAGE STATE TRANSITION PROBABILITY OF THE *PlaceCylinder* ACTION. IN EACH ROW, A DIFFERENT LEVEL OF UNIFORM NOISE IS ADDED TO THE STARTING SCENARIOS, ON WHICH WE OPTIMIZED $\mathcal{P}$.

| **Noise** | **Symbolic NID Rules** $\mathcal{P}^\star$ |
|---|---|
| 0 | `PlaceCylinder`(X,Y): `inhand`(X), `on`(Y,Z), `table`(Z), `cylinder`(X) $\rightarrow \begin{cases} \mathbf{1.0} & \texttt{on}(X,Y), \neg\texttt{inhand}(X) \\ 0.0 & \textit{noise} \end{cases}$ |
| 1.5 | `PlaceCylinder`(X,Y): `inhand`(X), `on`(Y,Z), `table`(Z), `cylinder`(X) $\rightarrow \begin{cases} \mathbf{0.92} & \texttt{on}(X,Y), \neg\texttt{inhand}(X) \\ 0.08 & \textit{noise} \end{cases}$ |

the symbolic data and thereby new symbolic rules.

With this approach we generally aimed to reciprocally couple motion planning on the motor level with action planning on the symbolic level. We believe that such a tighter integration is necessary for robots to eventually solve complex manipulation tasks more robustly. Equally we generally aimed to provide a new perspective on how the existing powerful methods for motion primitive optimization can be integrated in larger systems for complex sequential manipulation: We essentially proposed to use learned symbolic rules instead of a predetermined task specifications to imply an objective function for the motion primitives.

In our specific approach we considered motion primitives generated via a planner, mostly since this is a natural extension of hand-tuned planning cost functions for each subtask and ensures a strong generalization to novel situations (e.g. obstacle configurations) and optimality of the generated motion. However, future reserach should investigate the full range of alternative parameterizations of motion primitives, perhaps including the possibility to parameterize and optimize the task spaces themselves.

Concerning limitations, in the presented experiments each motion primitive has only been optimized on 6 training scenarios. While we demonstrated reasonable performance, 6 training scenarios are not enough to robustly cover the full space and prevent overfitting. Ideally the system should use each new encountered situation in an online manner to readjust the motion primitives—this would imply a system that directly learns from a failure to produce the desired effect instead of the batch learning approach presented above.

Further, the considered grasp and place primitives lend to rather simple rules. While their success probabilities change with the motion quality, their structure (which predicates are involved) remains invariant. In this case, continued iteration of motion primitive optimization and rule re-learning has no effect after one iteration. In the future we would like to investigate more complex scenarios and motion primitives, where the structure of the rule set really changes with the optimization of the motion primitives, demonstrating the full power of coupling learning on both levels.

## REFERENCES


[1] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *ICRA*, 2011, pp. 1470–1477.

[2] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning rhythmic movements by demonstration using nonlinear oscillators," in *In Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS2002)*, 2002, pp. 958–963.

[3] T. Lang and M. Toussaint, "Planning with noisy probabilistic relational rules," *Journal of Artificial Intelligence Research*, vol. 39, pp. 1–49, 2010.

[4] T. Lang, "Planning and exploration in stochastic relational worlds," Ph.D. dissertation, Fachbereich Mathematik und Informatik, Freie Universität Berlin, 2011.

[5] T. Rühr, J. Sturm, D. Pangercic, D. Cremers, and M. Beetz, "A generalized framework for opening doors and drawers in kitchen environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN, USA, May 14–18 2012.

[6] M. Beetz, F. Stulp, B. Radig, J. Bandouch, N. Blodow, M. Dolha, A. Fedrizzi, D. Jain, U. Klank, I. Kresse, A. Maldonado, Z. Marton, L. Mösenlechner, F. Ruiz, R. B. Rusu, and M. Tenorth, "The Assistive Kitchen – A Demonstration Scenario for Cognitive Technical Systems," in *IEEE 17th International Symposium on Robot and Human Interactive Communication (RO-MAN), Muenchen, Germany*, 2008, pp. 1–8, invited paper.

[7] T. Asfour, K. Regenstein, P. Azad, J. Schrder, N. Vahrenkamp, and R. Dillmann, "Armar-iii: An integrated humanoid platform for sensory-motor control," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2006, pp. 169–175.

[8] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, and Y. Bengio, Eds. Cambridge, MA: MIT Press, 2009.

[9] S. Džeroski, L. de Raedt, and K. Driessens, "Relational reinforcement learning," *Machine Learning Journal*, vol. 43, pp. 7–52, 2001.

[10] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning symbolic models of stochastic domains," *Journal of Artificial Intelligence Research (JAIR)*, vol. 29, pp. 309–352, 2007.

[11] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: ACM, 2009, pp. 1049–1056. [Online]. Available: http://doi.acm.org/10.1145/1553374.1553508

[12] D. M. Murray and S. J. Yakowitz, "Differential dynamic programming and Newtons method for discrete optimal control problems," *Journal of Optimization Theory and Applications*, vol. 43, pp. 395–414, 1984.

[13] W. Li and E. Todorov, "An iterative optimal control and estimation design for nonlinear stochastic system," in *Proc. of the 45th IEEE Conference on Decision and Control*. San Diego, CA, USA: IEEE, 2006, pp. 3242–3247.

[14] N. Hansen and S. Kern, "Evaluating the CMA evolution strategy on multimodal test functions," in *Parallel Problem Solving from Nature PPSN VIII*, ser. LNCS, X. Yao *et al.*, Eds., vol. 3242. Springer, 2004, pp. 282–291.