

Optimizing Sequences of Probabilistic Manipulation Skills Learned from Demonstration

Lukas Schwenkel

Institute for Systems Theory and Automatic Control, Stuttgart, Germany
Lukas.Schwenkel@ist.uni-stuttgart.de

Meng Guo, Mathias Bürger

Bosch Center for Artificial Intelligence (BCAI), Renningen, Germany
Meng.Guo2, Mathias.Buerger@de.bosch.com

Abstract: While manipulation skills such as picking, inserting and placing were hard coded in classical setups, it is now widely understood that this leads to poor flexibility and that more general skill formulations are required to ensure re-usability in new scenarios. We thus adopt a skill-centric approach where each skill is learned independently under various scenarios but not attached to any specific task. Afterwards, complex manipulation tasks can be achieved by composing these skills in sequence or parallel. One essential challenge there is to optimize the parameters of each skill such that the success rate of the whole task is maximized. Common approaches require first a discretization of the state or action space to generate such parameters and second a precise simulator to evaluate the performances under different parameters. Instead, we propose to learn task-parameterized models of each skill directly from few human demonstrations. Such models allow us to infer the success rate of executing a skill within a new scenario conveniently, via computing a novel measure of execution confidence. This measure encapsulates both the robot state and the workspace configuration. Furthermore, we introduce task-parameterized transition skills that change the object poses of interest via translation and rotation. We show that such skills can be extremely useful for changing skill parameters and thus potentially improving the success rate of a given task. The proposed scheme optimizes skill parameters in the continuous domain without the need for simulators. We demonstrate the proposed approach on a 7 DoF robot arm solving various manipulation tasks.

Keywords: Learning by Demonstration, Task Parametrized Skills, Hidden Semi-Markov Models, Motion and Task Planning.

1 Introduction

Deploying service robots in daily household environments or in highly flexible manufacturing sites is promising, but also highly challenging as stated in [1]. To begin with, it is impossible for robot manufacturers to pre-program *all* robot capabilities (referred to as *skills*) that final users may potentially require from the robot. To avoid inquiring engineers whenever a new skill is needed, it is crucial to provide an easy and efficient method with which laymen can teach the robot new skills. Simply recording and replaying a demonstrated trajectory is often insufficient, because changes in the environment, such as varying robot and/or object poses, would render any attempt unsuccessful. In other words, the robot needs to recognize and encode the intentions behind these demonstrations and, more importantly, to generalize over unforeseen situations. Many learning-from-demonstration (LfD) frameworks have shown great improvements in this aspect. Compared to hard-coded alternatives, they embed extracted knowledge into probabilistic models. Examples are probabilistic motion primitives (ProMPs) by [2] and Task-Parameterized Gaussian Mixture Models (TP-GMMs) by [3].

Complex manipulation tasks can be achieved by composing these skills in sequence or parallel. Thus it is desirable that these skills are learned in a general way such that they can be re-used in different

tasks. However, a direct execution of the sequence as it is given would often be unsuccessful due to mainly two reasons: unseen scenarios and incompatible skills. For instance, the task “grasp the coffee pot, pour coffee into the mug, grasp the mug, and pass the mug” would fail if the pot is too far away, or if the mug is grasped badly to pass it properly. Most of the existing optimization of task performance such as success rate and cost have been relying on statistics of numerous executions either in simulation or directly on the robots (see [4, 5, 6]). This not only requires the access to precise simulators that can model physical interactions, but also imposes great computational complexity on the planning and control cycle.

On the other hand, transition skills such as translating and rotating an object are used *extensively* by humans to ease the transition from one motion to another, e.g., pull the pot closer, and grasp the mug by its handle, for the previous example. However, their application to robotic manipulation tasks has been quite limited. Several works treat them in the same way as other skills or merge them manually into the execution sequence, see [7, 6]. This limits the re-usability of transition skills as they are inherently different from other skills, i.e., the final system configuration after executing is not fixed but rather a design choice. This choice is normally not optimized but hard-coded for specific tasks, which thus can not be applied to general manipulation tasks.

The contributions of this work are as follows. First, we extend the well-known TP-GMMs framework for learning-from-demonstration (LfD) by introducing so-called *free* task parameters, which are not directly attached to a physical object but can be freely chosen. Free task parameters are an important step towards integrating a representation of movable objects in the LfD setting. Second, we introduce task-parameterized models of pre-conditions and effects for general skills, which can be used to close the existing gap between trajectory planning and symbolic planning. Third, based on these probabilistic models, we introduce a measure of execution confidence for both a single skill and a skill sequence given a new scenario. A novelty of the proposed measure is that it computes purely from the learned models, without relying on any simulator. Fourth, we propose an algorithm that maximizes the execution confidence by optimizing the free task parameters in the continuous domain. This approach enables robots to autonomously decide how to use auxiliary transition skills to improve the success rate of a complete sequence of skills. The proposed formalism directly combines LfD and planning with continuous object poses into one formal framework, and therefore enables more flexible usage of learned skills in complex manipulation tasks. Finally, we demonstrate the real-world relevance of the approach in several experiments with a 7-DoF robot.

2 Related Work

LfD is an intuitive and natural way to transfer human skills to robots, which recently gained much attention [8, 9, 10]. Gaussian Mixture Models (GMMs) provide an elegant probabilistic representation of motion skills. For instance, the work by [8] or [10] show how to use them to extract important features from only few human demonstrations, see [11, 10, 12]. Furthermore, TP-GMMs reviewed in [3] provide a powerful extension to GMMs by incorporating observations from different perspectives. This allows automatic adaptation to new situations in [13], and has shown reliable performance in numerous applications, e.g. human-robot collaboration [14], robot bimanual sweeping [15] and service robot [3]. In this work, we build on skills that are learned from demonstrations and sequence them in an optimal way to accomplish more complex manipulation tasks.

Planning for a sequence of manipulation skills is closely related to the area of task and motion planning. On the task planning side, many work such as [5] has been focusing on constructing symbolic representations of skills such as preconditions and effects, which is used for high-level task planning, e.g., using standard languages like PDDL [16] and STRIPS [17]. This high-level abstract decision-making greatly accelerates the planning process, thus highly desirable. On the motion planning side, as a fundamental problem of robotics, many methods have been proposed as reviewed in [18] and [19], e.g., sampling-based methods such as PRM from [20] and RRTs from [19]; dynamic programming approaches such as LQG from [21] and MPC from [22]. This low-level sensing and control is unavoidable for the ultimate success of any task. On the other hand, the area of task and motion planning attempts to improve the synergies between high-level task planning and low-level motion planning. One direct challenge that arises is that geometric constraints and state abstractions are difficult to capture symbolically and more so automatically. Early work in [7, 23] combines a finite sampling of continuous actions and hierarchical task networks (HTNs). Recent work in [4] develops efficient search heuristics for the symbolic planning that reflects the geometric

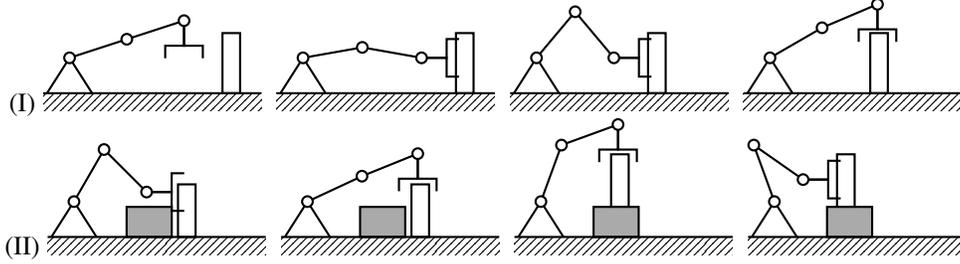


Figure 1: Two situations where transition skills are useful: (I) the peg is too far away for skill `grasp_top`, and transition skill `translate` grasps the peg by side and pulls it closer; (II) the peg is blocked by a platform thus *unsafe* for skill `grasp_side`, and transition skill `translate` grasps the peg by top and places it on the platform, to facilitate `grasp_side`.

constraints in the motion planning process, for the specific task of pick-place-move. A more general framework is proposed in [6] that combines symbolic logic reasoning with stable interaction modes, for sequential manipulation and tool-use planning, with a strong emphasis on the underlying physical interactions. Spatial conditions and effects for skills are learned in [24] as random forests and in [25] as neural networks. Similar ideas are adopted in the domain of hierarchical reinforcement learning as proposed in [26, 27, 28], for domains that involve long-term reasoning but under sparse and delayed rewards. However, the availability of accurate simulators is *essential* for all the aforementioned work to generate enormous amount of training data without running the actual robot. In this work, we remove this necessity but relying on only probabilistic models learned from human demonstrations.

Transition skills can be of many forms. Recent work by [29] uses in-hand manipulation to change the grasp on objects so to facilitate subsequent tasks, while tool-use skills such as hitting and hooking are proposed in [6]. Navigation can be seen as a special transition skill for mobile manipulators as shown in [5, 7], i.e., by moving directly the base to a more favorable position for grasping. In this work, we focus on a more specific form of transition skills that are learned from demonstrations.

3 Problem Description

Consider a multi-DoF robotic arm, of which the end-effector has state $\xi \in \mathcal{M}_\xi$, where \mathcal{M}_ξ is the robot operation manifold. For instance, \mathcal{M}_ξ could contain its 3D Cartesian position, orientation in quaternion and gripper state. We assume that the robot operates within a static and known workspace. Also, within the reach of the robot, there are objects of interest denoted by $\mathcal{O} = \{o_1, o_2, \dots, o_J\}$. Without loss of generality, we assume that the state of all objects $p \in \mathcal{M}_p$ lies in the object configuration manifold \mathcal{M}_p . For instance, \mathcal{M}_p could contain its 3D Cartesian position and orientation in quaternion.

Moreover, there is a set of *core* manipulation skills that enable the robot to manipulate these objects, denoted by $A = \{a_1, a_2, \dots, a_H\}$. For each skill, a human user performs several kinesthetic demonstrations on the robot. Particularly, for skill $a \in A$, the set of objects involved is given by $\mathcal{O}_a \subseteq \mathcal{O}$ and the set of demonstrations is given by $D_a = \{D_1, \dots, D_{M_a}\}$, where each demonstration D_m is a sequence of states s that consists of the robot end-effector state ξ within the manifold \mathcal{M}_ξ , and object states $\{p_o, o \in \mathcal{O}_a\}$ each within the manifold \mathcal{M}_p , i.e.,

$$D_m = [s_t]_{t=1}^{T_m} = [(\xi_t, \{p_{t,o}, o \in \mathcal{O}_a\})]_{t=1}^{T_m}. \quad (1)$$

Via a combination of these skills, the objects can be manipulated to reach a desired final state.

In addition, there is a special type of skills called *transition* skills, which are general skills used to change the states of movable objects, such as position and rotation. They can be demonstrated in the same way as the aforementioned skills, with a key difference that the final states of these demonstrations are *freely* chosen.

Now consider a new scenario with the robot and objects at different states, where the robot is given a sequence of manipulation skills to perform (denoted by a). The problem we tackle is to evaluate *whether, where and how* transition skills should be inserted in a to improve the success rate of

the whole task. Some motivating examples are shown in Fig. 1. The transition skill `translate` is used to improve the execution of both `grasp_top` and `grasp_side` within different scenarios.

4 Object-Centric and Task-Parameterized Models for Skills

In this section, we expand the classic framework from [3] such that the skill model now contains two parts: (I) the trajectory model that encodes spatio-temporal features of the demonstrations; (II) the precondition and effect models for the skill execution, which is proposed for the first time. These models are essential for the definition and optimization of skill confidence later.

4.1 Learning Trajectory Model

We use the task-parameterized hidden semi-Markov Models (TP-HSMMs), which extend TP-GMMs to encode spatio-temporal features of the demonstrated trajectories, as proposed in [3]. Task parameters are essential for the flexibility and generalization of the learned model. Here we mainly discuss how to design the task parameters for both the core manipulation skills and transition skills.

Task parameters are usually *attached* to the objects relevant to the skill and constructed from their poses. Constructing a task parameter from an object pose $\mathbf{p} \in \mathcal{M}_p$ is straightforward, see [30]. However, for transition skills there are relevant frames which are not related to a physical object. For example, the destination pose of the skill `translate` is freely chosen during the demonstration. Such a *free* task parameter can not be perceived and should be set explicitly by the user or a planning algorithm. Clearly, the choice of such free task parameters would directly influence the outcome of skill execution. Thus, for any skill a , the set of task parameters (denoted by TP_a) can be chosen among the following parts: the set of relevant objects O_a , the set of free task parameters (denoted by F_a), and the robot arm initial pose (denoted by r). Different choices of task parameters can result in significant changes in the performance. As rule of thumb, attaching frames to all parts covers many cases, i.e., $\text{TP}_a = \text{O}_a \cup \text{F}_a \cup \{r\}$. In the supplementary material, we discuss why this is not always a good choice and an iterative method could be applied to choose the best task parameters.

Once the task parameters are chosen, the associated TP-HSMMs can be derived from the procedures described in [3], which involves the iterative Expectation-Maximization (EM) algorithm. Very generally speaking, TP-HSMMs consist of the following parameters

$$\theta_a = \left\{ \{a_{kh}\}_{h=1}^K, (\mu_k^D, \sigma_k^D), \{\pi_k, \{(\mu_k^{(p)}, \Sigma_k^{(p)})\}_{p \in \text{TP}_a}\}_{k=1}^K \right\}, \quad (2)$$

where a_{kh} is the transition probability from state k to h ; (μ_k^D, σ_k^D) describe the Gaussian distributions for the duration of state k , i.e., the probability of staying in state k for a certain number of steps; $\{\pi_k, \{(\mu_k^{(p)}, \Sigma_k^{(p)})\}_{p \in \text{TP}_a}\}_{k=1}^K$ contains the K TP-GMMs for all task parameters in TP_a .

4.2 Learning Precondition and Effect Models

The precondition of a skill refers to the relative relations between the robot arm and the relevant objects, which should be satisfied initially for the skill execution to be successful. The effect of a skill refers to how the skill execution would change the state. In this part, we describe how to learn these models purely from the available demonstrations.

4.2.1 Task Parameterized Model

The trajectory model θ_a does not incorporate how the objects or robot arm are located w.r.t each other when the skill execution starts and finishes. The proposed idea is to learn task-parameterized Gaussians (TP-Gs) for each object to fit its pose from demonstrations. The *precondition* model is:

$$\gamma_{1,a}(s, \mathbf{p}_F) \triangleq \left\{ (\mu_{1,o}^{(p)}, \Sigma_{1,o}^{(p)}), \forall p \in \text{TP}_a \setminus \{o\} \right\}_{o \in \text{O}_a \cup \text{F}_a}, \quad (3)$$

where $(\mu_{1,o}^{(p)}, \Sigma_{1,o}^{(p)})$ is the Gaussian distribution of object o 's initial pose at time 1 from the perspective of object p 's initial pose at initial time 1. Thus it is also called the "initial-to-initial" precondition model. Similarly, the *effect* model of a skill is defined by:

$$\gamma_{T,a}(s, \mathbf{p}_F) \triangleq \left\{ (\mu_{T,o}^{(p)}, \Sigma_{T,o}^{(p)}), \forall p \in \text{TP}_a \right\}_{o \in \text{O}_a}, \quad (4)$$

where $(\boldsymbol{\mu}_{T,o}^{(p)}, \boldsymbol{\Sigma}_{T,o}^{(p)})$ is the Gaussian distribution of object o 's final pose at time T from the perspective of object p 's initial pose. Thus it is also called the ‘‘initial-to-final’’ effect model. Note that both models are computed within the object pose manifold \mathcal{M}_p . Low variance in the learned models indicate a consistent geometric relation in the demonstrations, e.g., the precondition of skill `insert` is that the object is grasped by arm initially, while the effect of skill `translate` is that the object is put at the destination. Derivation details for (3) and (4) can be found in the supplementary file.

4.2.2 Evaluation of Precondition

Given a new system state s and a choice of free task parameters \mathbf{p}_F , we can evaluate how much the precondition of a skill is satisfied using the learned models. In particular, we can compute the product of the observation probability for the robot arm and each object, or equivalently the logsum:

$$c_a(s, \mathbf{p}_F) \triangleq \log \left(\sum_{k=1}^K \pi_k \mathcal{N}(\boldsymbol{\xi} | \hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}}_k) \right) + \sum_{o \in \mathcal{O}_a \cup \mathcal{F}_a} \log \left(\mathcal{N}(\mathbf{p}_o | \hat{\boldsymbol{\mu}}_{1,o}, \hat{\boldsymbol{\Sigma}}_{1,o}) \right), \quad (5)$$

where $\{(\hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}}_k)\}$ are the combined Gaussians of initial robot arm pose in the global frame from the learned trajectory model $\boldsymbol{\theta}_a$; $\{(\hat{\boldsymbol{\mu}}_{1,o}, \hat{\boldsymbol{\Sigma}}_{1,o})\}$ are the combined Gaussians of object o 's initial pose in the global frame from the learned precondition model $\boldsymbol{\gamma}_{1,a}$. The computation of these components involves transforming Gaussians from local frames to the global frame and then computing their product. No closed analytical forms exist for general Riemannian manifolds, see [30]. Note that the measure above is not a probability, but computation over probability densities. It provides a continuous value that evaluates how similar the current situation is to the demonstrations.

4.2.3 Prediction of Effect

The effect includes the poses of both robot arm and objects after executing the skill. First, the final pose of the robot arm follows the learned trajectory model $\boldsymbol{\theta}_a$, i.e., $\boldsymbol{\xi}_T | (s_0, \mathbf{p}_F) \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_K, \hat{\boldsymbol{\Sigma}}_K)$, where $(\hat{\boldsymbol{\mu}}_K, \hat{\boldsymbol{\Sigma}}_K)$ is directly the K -th combined Gaussian the global frame. Second, the final poses of all objects follow the learned effect model $\boldsymbol{\gamma}_{T,a}$, i.e., $\mathbf{p}_{T,o} | (s_0, \mathbf{p}_F) \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_{T,o}, \hat{\boldsymbol{\Sigma}}_{T,o})$, where $(\hat{\boldsymbol{\mu}}_{T,o}, \hat{\boldsymbol{\Sigma}}_{T,o})$ is the combined Gaussian of object o in the global frame. Thus, the estimated final state \hat{s}_T after executing skill a is given by:

$$\hat{s}_T \triangleq \Omega_a(s_0, \mathbf{p}_F) \triangleq \left(\boldsymbol{\xi}_T | (s_0, \mathbf{p}_F), \{\mathbf{p}_{T,o} | (s_0, \mathbf{p}_F), o \in \mathcal{O}_a\} \right), \quad (6)$$

where the mean of the corresponding Gaussians can be directly used as the most likely prediction. Different from the symbolic representations for planning from [31], the precondition and effect models learned in (5) and (6) are continuous and adaptive to the actual scenario during execution.

5 Confidence Optimization for Skill Sequence

In the previous section, we describe the task-parameterized trajectory, precondition and effect models of a single skill. In this section, we first define the execution confidence of a sequence of skills within a new scenario. Then, we show how to compute and further optimize this confidence measure.

5.1 Confidence Measure for Skill Sequence

Typically, regardless of the way being implemented, a manipulation skill can only be executed successfully within a limited set of scenarios and robot configurations. A measure of *confidence* is used in this work to indicate how likely it is to perform the skill successfully within a new scenario. For instance, the success rate of a ‘‘grasping’’ skill varies greatly given different orientations and locations of the targeted object. As mentioned in Sec. 2, most related work computes this measure via simulating this execution in an accurate simulator numerous times and then averaging over the outcome. However, in this work, we rely on only the available demonstrations.

To begin with, it is worth pointing out that to learn the *exact* distribution of the success rate over the high dimensional space of all possible scenarios is extremely difficult. For instance, a manipulation skill that involves two objects has task parameters of dimension $2 \dim(\mathcal{M}_p) + \dim(\mathcal{M}_\xi)$, i.e., 22 for

Algorithm 1: Optimizing Sequences of Probabilistic Skills Learned from Demonstration

Input: Set of skills A ; demonstration D_a for each $a \in A$; skill sequence \mathbf{a} .

```
1 for each  $a \in A$  do // During training
2   Choose task parameters  $TP_a$ . // Sec.4.1
3   Train TP-HSMM model  $\theta_a$ , given  $TP_a$  and  $D_a$ .
4   Learn the precondition model  $\gamma_{1,a}$  and effect model  $\gamma_{T,a}$ , given  $TP_{T,a}$  and  $D_a$ .
   // Sec.4.2
5 Observe the initial system state  $s_0$ . // Sec.5.2
6 Determine  $\mathbf{p}_F^*$  by solving (9). // Optimization
7 for each  $a_h \in \mathbf{a}$  do // On-line execution
8   Observe the current system state  $s_h$ ; Set free TPs of  $a_h$  according to  $\mathbf{p}_F^*$ .
9   Compute the most-likely sequence  $\mathbf{k}^*$  of states in  $\theta_{a_h}$ . // see Calinon [3]
10  Generate reference trajectory  $\xi^*$  based on  $\mathbf{k}^*$ ; Track  $\xi^*$  by motion control till the end.
```

the 7 DoF robot arm described in our experiment. Since only a few demonstrations are available for each skill (rather than thousands), our idea is that scenarios which are similar to the demonstrations are more likely to lead to a successful execution, than those that are quite different.

Design of the confidence measure for a sequence \mathbf{a} are guided by two requirements: (a) the confidence of each single skill within \mathbf{a} should have influence on the overall confidence; (b) the confidence of \mathbf{a} can not be increased by appending additional skills. The first requirement is due to intuition and the second due to causality. In particular, consider a given sequence $\mathbf{a} = a_1 a_2 \dots a_H$. The confidence measure of \mathbf{a} at state s_0 with the chosen free parameters \mathbf{p}_F is defined as

$$c_a(s_0, \mathbf{p}_F) \triangleq -\log \left(\sum_{h=1}^H \exp(-c_{a_h}(\hat{s}_h, \mathbf{p}_F)) \right), \quad (7)$$

$$\hat{s}_{h+1} \triangleq \Omega_{a_h}(\hat{s}_h, \mathbf{p}_F), \quad \hat{s}_1 \triangleq s_0, \quad (8)$$

where (7) computes the non-logarithmic reciprocal sum of the confidence measure of each single skill from (5); and (8) predicts the initial state \hat{s}_h of each skill $a_h \in \mathbf{a}$ based on (6). When the sequence contains only one skill, (7) falls back to $c_a(\cdot)$ in (5). First, it can be verified that the above definition satisfies both requirements mentioned above. Second, the predicted initial state \hat{s}_h is essential for computing the correct confidence, as executing earlier skills changes the system state s_t and thus the associated task parameters for later skills. Due to this, simply computing the confidence $c_{a_h}(\hat{s}_h, \mathbf{p}_F)$ of each skill a_h by setting $\hat{s}_h = s_0$ would yield a quite *bad* estimate of the actual confidence. Lastly, this measure is not a probability, but a relative measure over probability densities, which can not directly tell the exact success rate given a new scenario.

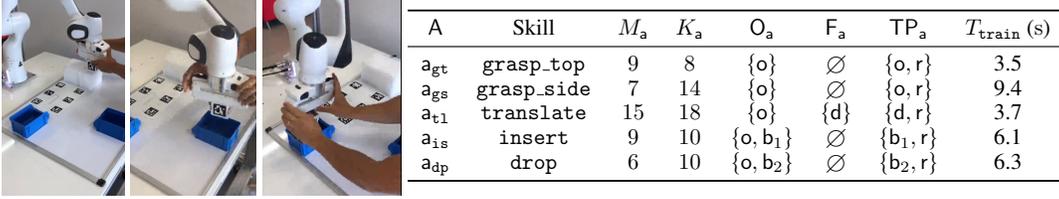
5.2 Confidence Optimization

In this section, we show how to set \mathbf{p}_F such that the overall confidence $c_a(\cdot)$ in (7) is maximized. More precisely, we solve the following optimization:

$$\mathbf{p}_F^* = \arg \max_{\mathbf{p}_F} \{c_a(s_0, \mathbf{p}_F)\}, \quad (9)$$

where remind that $\mathbf{p}_F = \{p_o, o \in F_a\}$ consists of the poses of all free task parameters. There are commonly as many free task parameters as the number of transition skills within \mathbf{a} . Note that the confidence measure $c_{a_h}(\cdot)$ for each intermediate skill a_h requires an iterative algorithm to determine the Gaussian products on the robot and object manifolds, as shown in (5). This renders computing the gradients of $c_a(s_0, \mathbf{p}_F)$ w.r.t \mathbf{p}_F directly *intractable*. We tackle this difficulty via two means: first we reduce the problem dimensionality by optimizing several free task parameters in sequence instead of simultaneously; then we rely on gradient-free direct-search optimization methods such as Nelder-Mead Simplex algorithm from [32]. The first aspect stems from the insight that the significance of each dimension of \mathbf{p}_F is encoded in its variance in the demonstrations, i.e., we optimize first in the directions that have the largest variations demonstrated.

Last but not least, for many manipulation tasks, we provide a fast-to-compute rough estimate of the optimum \mathbf{p}_F^* . Consider the skill sequence $a_h a_{h+1} \dots$ where a_h is a transition skill. If the



A	Skill	M_a	K_a	O_a	F_a	TP_a	$T_{\text{train}} \text{ (s)}$
a_{gt}	grasp_top	9	8	{o}	\emptyset	{o, r}	3.5
a_{gs}	grasp_side	7	14	{o}	\emptyset	{o, r}	9.4
a_{t1}	translate	15	18	{o}	{d}	{d, r}	3.7
a_{is}	insert	9	10	{o, b ₁ }	\emptyset	{b ₁ , r}	6.1
a_{dp}	drop	6	10	{o, b ₂ }	\emptyset	{b ₂ , r}	6.3

Figure 2: Left figure: the experiment set-up and snapshots of the kinesthetic teaching for skills a_{gs} , a_{is} , a_{dp} ; Right table: demonstrated skills A, number of components K_a and demonstrations M_a , involved objects O_a , free TPs F_a , choice of TPs TP_a , and the training time.

free task parameter d is the destination where object o is moved to, and skill a_{h+1} changes p_o during execution, then the optimum is very close to the mean of the initial pose of object o in the demonstrations of skill a_{h+1} , i.e., $p_d^* = \hat{\mu}_{1,o}$. This rough estimate can be either directly used if the resulting confidence is sufficient, or used as the initial guess for the optimization algorithm.

5.3 Execution of Skill Sequence

The complete procedure is summarized in Algorithm 1. The training is done off-line for each skill between Line 2–4. Before the execution starts, the optimal TPs p_F^* is computed in Line 6 after observing the initial state s_0 . Then, each skill is executed as described in Lines 8–10. Real-time execution relies on external modules such as perception and low-level motion control.

6 Experiments

In this section, we describe the experiment setup on a 7-DoF robotic manipulator. The manipulation task we consider is the typical assembly task with various picking and placing skills. We demonstrate how transition skills can be used to optimize the execution of such tasks within different scenarios. The algorithm is implemented in Python and its communication with other modules are through ROS. More details and experiment videos attached as supplementary files.

6.1 Setup and Manipulation Tasks

The Franka Emika Panda robot [33] has 7-DoF and is equipped with a two-finger gripper, as shown in Fig. 2. It provides a mode to intuitively perform kinesthetic teaching, during which the trajectory of the end-effector and the gripper can be fetched directly from the on-board control manager. The manipulated object is a cardboard cube of approximately $6 \times 4 \times 2 \text{ cm}^3$. To record trajectories also from other frames, the task parameters associated with the objects of interest in the scene are estimated based on the ArUco fiducial marker library from [34], which provides a 6D pose estimation with around 1 cm accuracy. Moreover, an impedance control module based on Hogan [35] is used to track a reference trajectory with the end-effector. Demonstrations are recorded at 50 Hz, while impedance controller runs at 1 kHz. Collision avoidance with unknown obstacles in the workspace is not considered. We adopt quaternions $S^3 \subseteq \mathbb{R}^4$ for orientation representation, which leads to the observation space $\mathcal{M}_\xi = \mathbb{R}^3 \times S^3 \times \mathbb{R}$ representing position, orientation and gripper width in that order. The poses of involved objects lie within $\mathcal{M}_p = \mathbb{R}^3 \times S^3$.

Within the range of the manipulation, there is an object o, two containers b_1 and b_2 where the object fits in. As shown in Fig. 2, we demonstrated the following five skills to the robot: `grasp_top` where the robot grasp o via the top (denoted by a_{gt}); `grasp_side` where the robot grasp o from the side (a_{gs}); `translate` where the robot translate o from one pose to the virtual destination d (by a_{t1}); `insert` where the robot inserts o into the container b_1 (by a_{is}); `drop` where the robot drops o into the container b_2 (by a_{dp}). Snapshots, number of demonstrations and involved objects are summarized in Fig. 2. Due to kinematic constraints and the object size, there are very limited regions where the robot could grasp o via the side without colliding with the table. Thus a platform of 5 cm is added to facilitate skill a_{gs} . We consider two pick-and-place assembly tasks: (a) grasp o by its top and drop it in b_1 , denoted by $\mathbf{a}_1 = a_{\text{gt}}a_{\text{dp}}$. Alternatively, the translation skill can be added in the beginning, i.e., $\mathbf{a}'_1 = a_{\text{gt}}a_{\text{t1}}a_{\text{gt}}a_{\text{dp}}$. (b) grasp o by its side and insert it in b_2 , denoted by $\mathbf{a}_2 = a_{\text{gs}}a_{\text{is}}$. Or with the translation skill, i.e., $\mathbf{a}'_2 = a_{\text{gt}}a_{\text{t1}}a_{\text{gs}}a_{\text{is}}$. The objective is to choose the appropriate sequence for different tasks under different scenarios.

p_{box}	c_{a_2}	$c_{a'_2}$	p_d^*	$T_{\text{opt}} \text{ (s)}$	c_{a_1}	$c_{a'_1}$	p_d^*	$T_{\text{opt}} \text{ (s)}$
(0.5, 0.0, 0.03)	-580	37	(0.36, 0.27, 0.07)	13.6	38	37	(0.42, 0.12, 0.03)	21.7
(0.6, 0.1, 0.03)	-800	32	(0.37, 0.28, 0.07)	16.5	32	31	(0.40, 0.12, 0.03)	15.2
(0.3, -0.08, 0.03)	-197	20	(0.36, 0.19, 0.07)	16.5	-1.2	19	(0.40, 0.1, 0.03)	12.2
(0.4, 0.3, 0.07)	41	34	(0.36, 0.28, 0.07)	16.7	31	31	(0.41, 0.13, 0.03)	22.2
(0.3, 0.4, 0.07)	28	32	(0.36, 0.30, 0.07)	24.6	30	31	(0.42, 0.13, 0.03)	20.6

Table 1: Comparison of the confidence measure at different initial object poses, for both tasks with (i.e., \mathbf{a}'_1 and \mathbf{a}'_2) and without transition skills (i.e., \mathbf{a}_1 and \mathbf{a}_2).

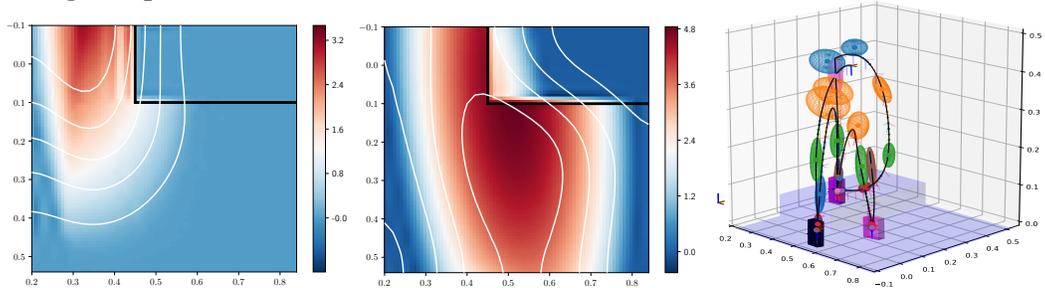


Figure 3: Heatmap of relative improvement over the workspace (\mathbf{X} , \mathbf{Y} -axis in m) when comparing \mathbf{a}_1 and \mathbf{a}'_1 (Left), and \mathbf{a}_2 and \mathbf{a}'_2 (Middle). The area within the black lines is elevated by the platform. Right: one execution trajectory for task $\mathbf{a}'_2 = \text{agtatsagsais}$. Boxes in shaded magenta are the predicted intermediate poses during execution, while box in solid black is the initial pose.

6.2 Confidence Optimization

First, we present the learning results for each skill described above. The procedure in Line 2–4 of Algorithm 1 is followed for each skill. The resulting models such as number of components and optimal choice of TPs are summarized in Fig. 2. All skills have been demonstrated within a wide range of scenarios, e.g., different robot arm state, different object poses for grasping skills and different destinations for the translation skill. The training time is around 6.1 s for each skill, which includes the TP-HSMM model θ_a and the precondition/effect models $\gamma_{1,a}$, $\gamma_{T,a}$. Detailed visualization of demonstrations and learned models are provided in the supplementary files.

After learning the skill models above, given desired sequence \mathbf{a} and the observed scenario \mathbf{s}_0 , the confidence $c_a(\cdot)$ can be maximized if it contains the transition skill \mathbf{a}_{t1} . The criterion to choose \mathbf{a}_1 or \mathbf{a}'_1 is as follows: \mathbf{a}'_1 is chosen if the relative improvement (RI) by $(c_{a'_1} - \min\{c_{\min}, c_{a_1}\})/\alpha > 1$, where c_{\min} is a predefined confidence lower bound (set to -10) and α is the scale (set to 10). The same rule applies to the choice of \mathbf{a}_2 and \mathbf{a}'_2 . A comparison of confidences for a selection of initial object poses is shown in Table 1, where the optimal choice for the free TP “destination” \mathbf{d} is also shown along with the computation time. The distribution of IR over the *whole* workspace is shown in Figure 3. It can be seen that: (I) for tasks \mathbf{a}_1 and \mathbf{a}'_1 , the difference in their confidences is mostly negligible as skill `grasp_top` has quite high confidence across the workspace. However, it is worth noticing that close to top-left corner, \mathbf{a}'_1 is preferred over \mathbf{a}_1 because skill `drop` has never been demonstrated around that area, thus it is beneficial to translate the box to the center area where skill `drop` is more confident. (II) for tasks \mathbf{a}_2 and \mathbf{a}'_2 , it is almost always beneficial to translate the object onto the platform first when the object is initially on the table, while this translation is unnecessary if the object is already on the platform. The main reason is that skill `grasp_side` is unsafe for the robot when close to the table due to potential collision, thus only demonstrated on the platform. Given the optimal choice p_d^* , the skill sequence \mathbf{a}'_1 is reproduced following Lines 8–10 of Algorithm 1. One actual execution trajectory of task \mathbf{a}'_2 is shown in Fig. 3. Experiment videos are attached as supplementary files.

7 Conclusions and Future Work

We presented in this work a general framework to optimize manipulation skills that are learned purely from human demonstrations. The learned models are object-centric and thus flexible to changing scenarios. Transition skills are introduced to improve the overall success rate of complex manipulation tasks. Future work includes the interplay between the skill-level optimization proposed here and existing high-level task planning techniques.

References

- [1] S. Bedaf, P. Marti, F. Amirabdollahian, and L. de Witte. A multi-perspective evaluation of a service robot for seniors: the voice of different stakeholders. *Disability and Rehabilitation: Assistive Technology*, pages 1–8, 2017.
- [2] A. Paraschos, C. Daniel, J. Peters, and G. Neumann. Probabilistic movement primitives. In *Advances in neural information processing systems (NIPS)*, pages 2616–2624, 2013.
- [3] S. Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016.
- [4] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling. FFRob: Leveraging symbolic planning for efficient task and motion planning. *The International Journal of Robotics Research*, 37(1): 104–136, 2018.
- [5] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61: 215–289, 2018.
- [6] M. Toussaint, K. Allen, K. Smith, and J. Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.
- [7] J. Wolfe, B. Marthi, and S. Russell. Combined task and motion planning for mobile manipulation. In *Twentieth International Conference on Automated Planning and Scheduling*, 2010.
- [8] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.
- [9] R. Lioutikov, O. Kroemer, G. Maeda, and J. Peters. Learning manipulation by sequencing motor primitives with a two-armed robot. In *Intelligent Autonomous Systems 13*, pages 1601–1611. Springer, 2016.
- [10] S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.
- [11] G. Ye and R. Alterovitz. Guided motion planning. In *Robotics research*, pages 291–307. Springer, 2017.
- [12] C. Pérez-D’Arpino and J. A. Shah. Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6175–6182. IEEE, 2015.
- [13] S. Calinon, D. Bruno, and D. G. Caldwell. A task-parameterized probabilistic model with minimal intervention control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3339–3344. IEEE, 2014.
- [14] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, and C. Torras. Learning physical collaborative robot behaviors from human demonstrations. *IEEE Transactions on Robotics*, 32(3):513–527, 2016.
- [15] J. Silvério, L. Rozo, S. Calinon, and D. G. Caldwell. Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 464–470. IEEE, 2015.
- [16] M. Fox and D. Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.
- [17] T. Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

- [18] J.-C. Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.
- [19] S. M. LaValle. *Planning Algorithms*. Cambridge university press, 2006.
- [20] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe. Analysis of probabilistic roadmaps for path planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3020–3025. IEEE, 1996.
- [21] W. S. Levine. Linear quadratic regulator control. In *The Control Systems Handbook*, pages 403–426. CRC Press, 2018.
- [22] F. Allgöwer and A. Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.
- [23] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 639–646. IEEE, 2014.
- [24] O. Kroemer and G. S. Sukhatme. Learning spatial preconditions of manipulation skills using random forests. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 676–683. IEEE, 2016.
- [25] V. Xia, Z. Wang, and L. P. Kaelbling. Learning sparse relational transition models. In *International Conference on Learning Representations*, 2019.
- [26] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379, Oct 2003. ISSN 1573-7594.
- [27] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.
- [28] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroché, T. Barnes, and J. Tsang. Hybrid reward architecture for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 5392–5402, 2017.
- [29] N. C. Daffe, R. Holladay, and A. Rodriguez. In-hand manipulation via motion cones. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.
- [30] M. Zeestraten. Programming by demonstration on Riemannian manifolds. 2017. PhD thesis.
- [31] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Elsevier, 2004.
- [32] F. Gao and L. Han. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51(1):259–277, Jan 2012.
- [33] E. Franka. Panda arm. <https://www.franka.de/panda/>, 2018.
- [34] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marn-Jimnez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014. ISSN 0031-3203.
- [35] N. Hogan. Impedance control: An approach to manipulation. *Journal of dynamic systems, measurement, and control*, 107(1):8–16, 1985.