2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Do What I Want, Not What I Did: Imitation of Skills by Planning Sequences of Actions*

Chris Paxton[1], Felix Jonathan[1], Marin Kobilarov[1], and Gregory D. Hager[1]

***Abstract*— We propose a learning-from-demonstration approach for grounding actions from expert data and an algorithm for using these actions to perform a task in new environments. Our approach is based on an application of sampling-based motion planning to search through the tree of discrete, high-level actions constructed from a symbolic representation of a task. Recursive sampling-based planning is used to explore the space of possible continuous-space instantiations of these actions. We demonstrate the utility of our approach with a magnetic structure assembly task, showing that the robot can intelligently select a sequence of actions in different parts of the workspace and in the presence of obstacles. This approach can better adapt to new environments by selecting the correct high-level actions for the particular environment while taking human preferences into account.**

I. Introduction

Learning from demonstration has emerged as a useful paradigm to teach robots the skills they need to interact with the real world. The challenge in learning from demonstration is to generalize what is learned to new contexts and new tasks. Consider a moderately complex task such as assembling part of a structure, shown in Fig. 1 and defined by the PDDL in Fig. 3. The precise movements and the particular movement goals and parameters will vary from one situation to the next. When attempting to execute this task in a new environment, the robot must be able to select the particular actions, motions, and manipulation goals that will allow completion of the task in this new environment. By exploiting learned models for actions, we are able to demonstrate a planner that is able to produce solutions for performing tasks in an effective manner, is able to improve with additional demonstration data, and can adapt to new circumstances.

Adapting to new environments in the context of task and motion planning poses several challenges when attempting to generalize learned actions. Recently there has been significant progress in integrating symbolic task planning and continuous motion planning [1]–[6], which have in the past evolved as two separate fields. At the same time, learning from demonstration has been established as a powerful tool for learning models of individual actions [7], [8]. Learning from demonstration has previously been connected to symbolic task planning [9]–[11], but these approaches are yet to be incorporated in the context of a motion-constrained task planning in a principled manner. This paper aims to address this gap.

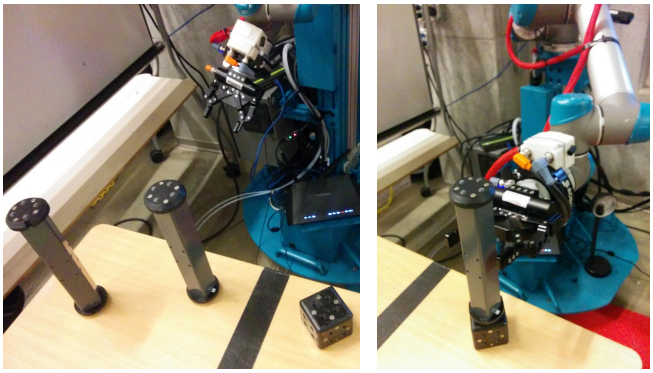


Fig. 1: UR5 performing part of a structure assembly task by grabbing a link object in order to connect it to a node. Actions and goals were defined by human demonstrations.

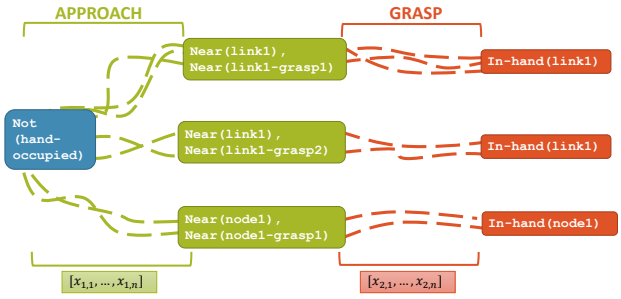


Fig. 2: Approach for grounding actions in symbolic planning. Training data represents actions connecting symbolic states in the graph of possible actions that constitute valid solutions to the task plan.

In our approach, probabilistic models over features associated with each action are learned from human demonstrations and later refined in a supervised manner using additional robot-generated examples scored by a human teacher. At the core of this approach lies a mapping from symbolic actions (e.g. `approach`, `grasp`) to physical motions encoded probabilistically as a distribution over observed features along each motion trajectory. Fig. 2 shows this relationship: multiple demonstrations connect predicate states, which allow us to learn a model of each action. Features x are defined as a set of functional relations between the robot and its environment (e.g. relative position and orientation between robot end-effector and desired object to grasp).

Planning in a new environment is accomplished by updat-

*This work was supported by NSF NRI 1227277.

Chris Paxton, Felix Jonathan, Marin Kobilarov, and Gregory D. Hager are with the Laboratory for Computational Sensing and Robotics, Johns Hopkins University, Baltimore, MD 21218 USA. {cpaxton3,fjonath1,marin,hager}@jhu.edu

```
(define (domain structure-assembly)
  (:requirements :typing :adl)
  (:types link node grasp-pt)
  (:predicates
    (colliding) (hand-occupied)
    (near ?x - link) (near-grasp ?g - grasp-pt)
    (in-hand ?x - link) (standing ?x - link)
    (aligned ?x - link ?y - node)
    (grasp-feasible ?g - grasp-pt)
    (grasp-for ?x - link ?g - grasp-pt)
    (grasp-for ?y - node ?g - grasp-pt)
    (attached ?x - link ?y - node)
    (approach-feasible ?x - link))
  (:action approach
    :parameters (?x - link)
    :precondition (and
      (not (in-hand ?x))
      (approach-feasible ?x))
    :effect(near ?x))
  (:action grasp ...)
  (:action align ...)
  (:action place ...)
  (:action release ...)
  (:action disengage ...))

(define (problem build-simple-structure)
  (:domain structure-assembly)
  (:objects
    link1 - link link2 - link
    node1 - node node2 - node
  (:goal (exists (?x - link ?y - node) (attached ?x ?y))))
```

Fig. 3: Partial PDDL domain and problem definition for the structure assembly task. The domain can be thought of as a version of the basic blocks world task, where the goal is to latch two pieces together.

ing each action distribution to remain as close as possible to the prior while satisfying new environment constraints such as different obstacles and object shapes. This is accomplished through importance sampling and optimal distribution re-estimation using the cross-entropy method [12], [13]. Transitions from symbolic states to actions are similarly encoded as a discrete probability distribution representing the "preference" of executing different actions. A product model is induced over a complete task from the sequence of probabilistic action models, together with discrete transition models. Planning a complete task then corresponds to optimally updating this model to reproduce the prior and satisfy the new scenario.

The contributions of this paper are: (1) a new method for reproducing demonstrated actions in novel environments, derived from sampling-based motion planning; (2) an algorithm for combining these learned actions for executing multi-step tasks with multiple valid plans; and (3) experimental validation of this algorithm on a simple assembly task as shown in Fig. 1. Experiments in a 2D Android game domain were omitted for reasons of space.

II. Related Work

Prior work exists in describing the relationship between high- and low-level actions and in learning representations of actions from demonstration, but does not combine learning with action selection and motion planning.

Object-Action Complexes (OACs) have been proposed as a way of formalizing actions unifying perception and learning that can be associated with learned low-level actions, and sequenced based on predicate effects by a symbolic planner [10]. The proposed method is similar to work such as [14], which grounded PDDL position predicates with Gaussian Mixture Models, and [11], which associated Dynamic Movement Primitives (DMPs) for particular actions with expected visual features.

Probabilistic models are commonly used in imitation learning, e.g. [15], [16]. Dynamic Movement Primitives (DMPs) are a policy representation that has proven useful for modeling low-level actions from demonstration as a set of dynamical systems [17]. Prior work has added object avoidance to these methods through potential fields [18], [19] or through reinforcement learning [20].

Pastor et al. used Path Integral Policy Improvement with DMPs and multiple human demonstrations to learn a model of expected features when executing two robotic tasks in [21]: shooting pool and flipping over a box with a pair of chopsticks. This method was expanded upon by Stulp et al., who proposed Path Integral Policy Improvement with Covariance Matrix Adaptation [22]. These techniques are closely related to the Cross-Entropy Method for motion planning [13] from which we draw inspiration.

Our work is also related to the method proposed by Engbert et al. use the KL divergence between an expert demonstration and trajectories sampled from a Gaussian Process forward model to optimize imitation learning policies [23]. Similarly in [24] the authors propose a method for inverse reinforcement learning based on minimization of relative entropy. In addition, the proposed approach can be thought of as a parameterized set of actions; this has been shown to improve performance on policy learning in Markov Decision Processes [25].

Other work combines learned motion primitives into state machines for execution, but in a purely reactive way: selecting only the next action, rather than the next sequence. Examples include Niekum et al. [26], who build a task plan from unstructured demonstrations. Manschitz et al. [27] learned classifiers to determine the next action when sequencing motion primitives. Work by Kappler et al. [28] uses Associative Skill Memories to perform dexterous tasks.

Symbolic task planning and motion planning have commonly been integrated through algorithms that "fill in the gaps" in symbolic plans with callouts to continuous-space motion planners. Recent work in combined task and motion planning include work by by Plaku et al. [1], by Shivashankar et al [3], by Wolfe et al [2], and by Lagriffoul et al. [4]. These works do not analyze actions with a wide variety of goals and cost functions, focusing instead on exploration and pick-and-place tasks. Similarly, Srivastava et al. [29] efficiently integrate task planning with continuous-space reasoning about goal positions, but still rely on callouts to a traditional motion planner to instantiate trajectories. Work by Toussaint describes a hierarchal approach for integrated task and motion planning that first examines feasible end states before optimizing kinematics and motion planning [30]. Unlike these methods, our approach jointly optimizes sequences of trajectories by adaptively allocating trajectory simulations to

different actions. However, the proposed approach suggests directions for future work in improving efficiency in complex domains.

III. Task Description

We assume existence of (1) a symbolic description of a task, and (2) labeled training data associating features with each low-level action that can appear in this domain. The symbolic description naturally decomposes the task into a sequence of predicate world states $w_0, w_1, ..., w_g$ For the structure assembly task, part of the symbolic description is shown in Fig. 3. A world state w is then defined as a combination of predicates. In turn, actions a are the connections between these predicate states as shown in Fig. 2. Each a in a given task is represented as a probability distribution over a set of features associated with a successful instantiation of a skill in a new environment given w.

The features are denoted by $x \in \mathbb{R}^n$ and defined using the function ϕ through relationship $x = \phi(t, s, u)$, where $t \in [t_0, t_f]$ denotes time in the action starting at t_0 and ending at t_f, $s \in S$ is the robot state, and $u \in U$ are the applied controls. With these definitions, a probabilistic model associated to each action a is denoted by $p_d(x|a)$ and is computed using unsupervised learning from expert demonstrations, typically assuming a parametric density p_d. A joint model of a task T consisting of multiple actions can be constructed using a density $p_d(x|T) \propto p_d(x|a_0) \cdots p_d(x|a_{n_T})$ assuming conditional independence between actions.

Specific features are derived from the PDDL description of the task. For example, in Fig. 3, the `approach` action describes the arm moving to pick up a `link` object without knocking it over. In this case $x = \phi(t, s, u)$ would return the relative position, orientation, and velocity between the robot end effector and the `link` object. To use the proposed method, one would provide the identifier for an action and a list of associated symbols from perception.

An optimal task T^* is a sequence of actions $T^* = \{a_i\}_{i=1}^N$ that takes the robot from the initial state w_0 to goal w_g that have the highest probability given our expert model, while also avoiding hard constraints such as collisions and joint limits:

$$T^* = \arg\max_T p(T|w_0, w_g) \tag{1}$$

$$= \arg\max_{a_1,...,a_N} \prod_i^N p(a_i|w_i, w_g) \tag{2}$$

Our goal is to learn a stochastic "symbolic" policy $\pi(a|w)$ over the sequence of predicate states, as well as continuous-space "physical" policy $p(u|s, \xi_a)$ generating trajectories for each action a. We represent trajectories using parameters $\xi \in \mathcal{Z}$, where $\mathcal{Z}$ represents the space of all possible parameters resulting in valid trajectories in the new environment. Since robot perception and motion are uncertain, each parameter induces a density $p(\tau|\xi)$ where

$$\tau = \{\langle t_0, s_0, u_0 \rangle, \langle t_1, s_1, u_1 \rangle, \ldots, \langle t_N, s_N, u_N \rangle\}$$

denotes the system trajectory. For instance, ξ would typically define a reference trajectory and an associated tracking control law resulting in the density

$$p(\tau|\xi) = p(s_0) \prod_{i=0}^{N-1} p(s_{i+1}|s_i, u_i) p(u_i|s_i, \xi).$$

In practice, given ξ the trajectory τ will either be sampled using a high-fidelity simulator or generated by the real robot.

IV. Planning Algorithm

A. Local Planning Algorithm

First, we consider adaptation of only a single action a to a new environment. When presented with a new environment, we pose the planning task as the problem of learning a new parameterized policy ξ^*. To do so we employ a stochastic optimization technique using a surrogate distribution $\xi \sim \pi(\cdot|v)$ which is iteratively updated so that generated trajectories τ produce feature observations x with high likelihoods under the expert distribution $p_d(x|a)$ for action $a \in A(w)$, where $A(w)$ is the set of actions available from predicate state w.

We follow the Cross Entropy Method described by Rubinstein et al. [12], particularly following its application to motion planning by Kobilarov [13]. This is accomplished by introducing an artificial *surrogate* distribution over $\mathcal{V}$ that will induce a distribution over trajectories τ and over the corresponding features x along these trajectories. The surrogate will then be iteratively optimized until it becomes optimally close (in a distribution sense) to the expert density $p_d(x|a)$ without violating the constraints of the environment such as obstacles and joint limits. The surrogate model is built using a parametric density $\pi(\xi|v)$ such as a multivariate Gaussian or a GMM with parameters v. Assuming that a nominal (prior) parameter v_0 is known the problem can be formalized as the optimal estimation of the expecation

$$l = E_{p(x|v_0)}[p_d(x|a)]. \tag{3}$$

The optimal importance sampling density [12] for estimating this integral is

$$q^* = \frac{p_d(x|a)p(x|v_0)}{l} \tag{4}$$

where the numerator in (4) can be thought of as the correlation between the expert feature distribution $p_d(\cdot|a)$ and the parameterized distribution $p(\cdot|v_0)$. Unfortunately we cannot compute the solution to (4) as it involves computing the estimator l from (3). Instead, we approximate this optimal q^* by finding the appropriate parameters v of $p(x|v)$. A logical way of doing this is to minimize the Kullback-Leibler (KL) divergence:

$$\min_v D_{KL}(q^*||p(\cdot|v)) \tag{5}$$

To find the value of v that minimizes this expression, we approximate this solution by drawing M i.i.d. samples $\xi_1, \ldots, \xi_M$ from v_0. In this case $x_{i,j} = \phi(t_i, s_{i,j}, u_{i,j})$ is a generated feature from robot state $s_{i,j}$ at time t_i along the sampled trajectory $\tau_j \sim p(\cdot|\xi_j)$ for $\xi_j \sim \pi(\cdot|v_0)$.

This can be more formally expressed as

$$v^* = \arg\max_v \int_x p_d(x) p(x|v_0) \log p(x|v) \tag{6}$$

$$\approx \arg\max_v \frac{1}{NM} \sum_i \sum_j p_d(x_{i,j}) \log p(x_{i,j}|v) \tag{7}$$

If we assume that there is a bijection between a tuple $\langle t, s, u \rangle$ along a trajectory τ and a feature $x \in \phi(\tau)$ then we have the following approximation

$$p(x_{i,j}|v) \approx \frac{p(\xi_j|v)}{p(\xi_j|v_0)}, \tag{8}$$

since ξ_j were sampled under v_0, and substituting (8) into (7) results in:

$$\arg\max_v \frac{1}{NM} \sum_i \sum_j p_d(x_{i,j}) \log p(\xi_j|v). \tag{9}$$

The necessary conditions for a minimum correspond to setting the gradient of (9) to zero, i.e. by solving the equality:

$$\sum_{i=0}^{N} \sum_{j=1}^{M} -z_{i,j} \nabla_v \log \pi(\xi_j|v) = 0, \tag{10}$$

where the weights $z_{i,j}$ are given by $z_{i,j} \triangleq p_d(x_{i,j})$.

When $\pi(\cdot|v) = \mathcal{N}(\cdot|\mu, \Sigma)|_{\mathcal{V}}$ (i.e. a single multivariate Gaussian with domain restricted to feasible parameter set $\mathcal{Z}$), the relationship (10) can be solved in closed form as

$$\mu = \sum_{j=1}^{M} \bar{z}_j \xi_j, \quad \Sigma = \sum_{j=1}^{M} \bar{z}_j (\mu - \xi_j)(\mu - \xi_j)^T, \tag{11}$$

where $z_j = \sum_{i=0}^{N} z_{i,j}$ and $\bar{z}_j = z_j / \sum_{j=1}^{M} z_j$. When $\pi(\cdot|v)$ is a GMM the minimization from Eq. (9) is performed using a weighted expectation maximization (EM) algorithm.

In practice, the optimal parameter v is computed iteratively starting with some nominal choice v_0 which approximately covers the trajectory space of interest. At each iteration we draw M samples $\xi_j \sim \pi(\cdot|v_0), j \in 1, \ldots, M$ and compute the next v by minimizing (9). At the next iteration v_0 is set to v and the process continues until the cost converges.

We add a fixed normalization term to the diagonal entries in Σ of p_d and of $\pi(\cdot|v)$ to make sure covariances stay well-defined. In addition, to prevent premature convergence, we introduce an extra parameter $0 < \alpha < 1$, which controls the size of steps taken at each iteration. In the case where v is multivariate Gaussian, with Σ_i^* as the optimal Σ at iteration i, we compute μ_{i+1} and Σ_{i+1} as:

$$\begin{aligned} \mu_{i+1} &= (1-\alpha)\mu_i - \alpha \mu_i^* \\ \Sigma_{i+1} &= (1-\alpha)\Sigma_i - \alpha \Sigma_i^* \end{aligned} \tag{12}$$

Avoiding Obstacles and Joint Limits: We constrain $\mathcal{Z}$ to consist only of the space of valid trajectories, removing any samples that would collide with objects or pass joint limits. This means that when drawing our M samples, we remove samples currently in collision or past joint limits in our new environment and continue to draw sample trajectories until we have all M valid examples. This works effectively in practice as long as the task does not require generalization in environments with very narrow passages that the system has never been trained on. Such cases are extremely difficult since the probability of obtaining samples in the narrow passage is close to zero, unless an informative nominal density parameter v_0 is used with enough probability mass over such regions.

B. Task Planning Algorithm

We wish to optimize parameters for all possible actions in a successful execution of the task, where our cost is the joint probability over any sequence of actions that represent a valid execution of the task as per Eq. (2). Our task planning approach takes the algorithm described in Section IV-A and expands it into a recursive algorithm similar to Monte Carlo Tree Search.

First, consider the problem of choosing one of $N_{A(w)}$ possible actions. We think of this as the choice of which action would be most similar to our expert's demonstrations in other scenes, starting in symbolic state w. The optimal action is given by:

$$a^* = \arg\max_{a \in A(w)} \int_x p_d(a|w_0) p_d(x|a) \tag{13}$$

$$\approx \arg\max_{a \in A(w)} \frac{1}{NM} \sum_i^N \sum_j^M p_d(a|w) p_d(x_i|a) \tag{14}$$

Action selection is modeled as a stochastic policy over possible worlds. We introduce a surrogate distribution into our trajectory search that captures the probability of choosing each future action from the current w. When sampling trajectories, we draw the next action $a \sim \pi(\cdot|w)$ according to this probability.

Furthermore, we can extend this reasoning to consider which of a whole tree of possible actions is the most similar to an expert tree, allowing us to capture expert preferences for particular actions in addition to continuous-space trajectories. Assuming that all actions in a branch of the tree are independent given time, we can define the expert probability of a particular action starting at continuous robot state s_0:

$$Q(s,a) = \frac{1}{NM} \sum_i^N \sum_j^M p_d(x_i|a) V(s_{N,j}) \tag{15}$$

$$V(s) = \sum_{a' \in A(w)} p_d(a'|w) Q(s,a') \tag{16}$$

Where $s_{N,j}$ is the final state in sampled trajectory τ_j and w represents the world after symbolic action a. Eq. (16) describes the probability of all possible actions from a continuous world state s occurring after execution of an action a.

When recording a set of N_w demonstrations starting in the same predicate state w, we compute the conditional probability for action $a \in A(w)$:

Algorithm 1 Pseudocode algorithm for optimal reproduction of demonstrated tasks in new environments.

Given: initial state s_0, horizon H, step size α, max iterations N_{iter}
for $i \in N_{iter}$ **do**
  for $a \in A(w_0)$ **do**
    $Q(s_0, a) =$ SAMPLE(a,$W(a)$,s_0,1,H,M)
  end for
  $V(s_0) = \sum Q(s_0, a)$
  if $V(s_0)$ has converged **then**
    Break;
  end if
end for

$$p_d(a|w) = \frac{\sum_i^{N_w} I_{\{a=A_i\}}}{N_w} \tag{17}$$

We specify a surrogate distribution over possible choices of actions for a world w given as $p(a|w)$. This probability is initialized as $\pi(a|w) = \frac{1}{N_{A(w)}}$. In the case where $H = 0$ this is updated as $\pi(a|w) \propto \frac{1}{M} \sum_j^M z_j$ where M trajectory samples τ have been drawn from a. Otherwise we compute this as:

$$\pi(a|w) = \frac{p_d(a|w)}{M} \sum_{s_0} Q(s_0, a)$$

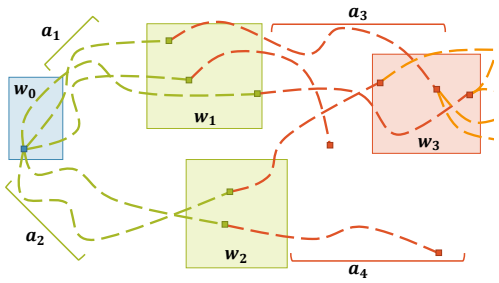
for starting state $s_0 \in S_0$ and action $a \in A(w)$. In practice we use the step size α to prevent this term from converging too quickly.

Each predicate state w corresponds to a range of valid continuous-space states. The algorithm recursively samples from the trajectories associated with each successive action to map to continuous states. As shown in Alg. 1, we repeatedly call the SAMPLE function from Alg. 2, providing it with the set of possible start states S_0. We select a start state from these s_0 according to the cumulative probability of these actions. The process continues until we reach a user-provided horizon H. This approach allows us to maintain a constant number of samples: over successive iterations, more samples will be devoted to promising regions of the search space.

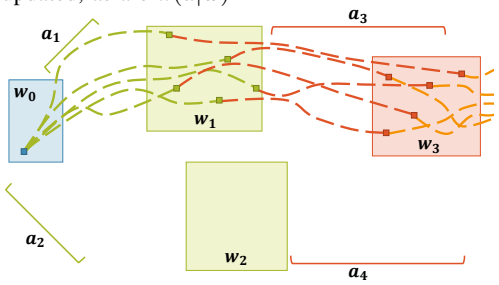
This results in a recursive search strategy outlined in Alg. 1. The return of the SAMPLE function is the average probability of all future actions and trajectories associated with each current start state. This value is used to compute a version of the weights in Eq. (9), where p_d is replaced by the probability of all future actions from each trajectory. Fig. 4 illustrates how the algorithm works in practice.

V. EXPERIMENTS

We performed experiments in a simulated Barrett WAM arm and on a Universal Robot UR5, applied to an object manipulation task. The goal of this task was to build a structure of increasing complexity out of magnetic blocks, as per the task described in [31]. In our case, we only perform a part of the whole structure assembly task: we combine one link and one node object to create a sub-structure. The



(a) At the first iteration of the algorithm, we sample trajectories (dashed lines) corresponding to a_1, a_2, a_3, etc. according to and compute $p_d(\tau|a, w)$. Trajectory distributions for $\pi(\cdot|v_1)$,$\pi(\cdot|v_2)$, etc. are updated, as are $\pi(a|w)$



(b) On subsequent iterations, trajectory sampling is biased towards a_1 due to the comparatively high probability of valid trajectories for each action in this space.

Fig. 4: Illustration of the proposed algorithm. Boxes w_1, w_2, etc. indicate regions corresponding to the predicate state after each action, while dashed lines represent continuous state trajectories.

connections between different skills are described by the PDDL specification in Figure 3. We used FastDownward [32] to translate the PDDL into a graph of possible actions that can be performed assuming all *feasibility* predicates are true.

Figure 5 shows how the planner works in practice. It iteratively sampled out different motions for each selected action, choosing to approach the link from the front and then to mate it to the leftmost node. As the algorithm progressed, successively fewer samples were drawn from actions associated with the rightmost node.

We use three types of features: (1) the time in a particular state, (2) the gripper command variables, (3) the transforms between the end frame and the objects. Relevant features are determined by the parameters specified in the task description in Fig. 3. In cases where two objects are parameters, we used the transform between the in-hand object and the other object. For these examples, $\phi(t, s, u) = [t, p_x, p_y, p_z, r_x, r_y, r_z, r_w, \|p\|, \dot{p}_x, \dot{p}_y, \dot{p}_z, \|\dot{p}\|]$, where values are computed from the offset between the current manipulation frame and and the relevant object. The values (r_x, r_y, r_z, r_w) define a unit quaternion. The manipulation frame is either an end effector position or the coordinate frame associated with the object in the gripper, for actions defined where the `hand-occupied` predicate is true.

We parameterize trajectories ξ with Dynamic Movement Primitives with 5 basis functions in the robot's joint space,

Algorithm 2 Recursive trajectory sampling and update step.

function SAMPLE($a, S_0, p(S_0), H, M$)
  $w = W(a)$ ▷ Predicate world after performing action
  for $j \in [1, ..., M]$ **do**
    $s_0 \sim S_0 \propto p(S)$ ▷ Sample start points
    $\xi_j \sim \pi(\cdot|v_a)$ ▷ Sample parameters
    $\tau_j \sim p(\cdot|\xi_j, s_0)$ ▷ Compute trajectories
  end for
  if $H > 0$ **then**
    $S'_0 = [s_N]_{j=1}^N$ ▷ Set start points
    $p(S'_0) = [p(s_{0,j})p_d(\tau_j|a)]_{j=1}^M$
      ▷ Compute probabilities of each start point
    for $a' \in A(w)$ **do**
      $H' = H - 1$ ▷ update horizon
      $M' = \pi(a'|w)M$ ▷ Number of samples
      $Q(S'_0, a') =$ SAMPLE(a' S'_0,$p(S'_0)$,H',M')
      $\pi(a'|w) = \frac{p_d(a'|w)}{M'} \sum_{s'_0} Q(s'_0, a')$
    end for
  end if
  for $j \in [1, \ldots, M]$ **do**
    $V(s_{N,j}) = \sum_{a' \in A(w)} p_d(a'|w)Q(s, a')$
    $z_j = \sum_i p_d(x_{i,j}|a)V(s_{N,j})$
    ▷ Compute update weights from child probability
  end for
  for $s_0 \in S_0$ **do**
    $V(s_0) = \frac{\sum_j I_{s_{0,j}=s_0} z_j}{\sum_j I_{s_{0,j}}}$
    ▷ Average probability from continuous start state
  end for
  $v'_a = \arg\min_v \frac{1}{N} \sum_i \sum_j z_j \log p(\xi_j|v_a)$
  return $V(S_0)$
end function

plus a goal pose $g \in SO(3)$. This allows us to find paths in the space of the robot arm, but to adapt to different possible continuous-space goals. We implemented the system using ROS [33] with Orocos KDL for inverse kinematics [34].

In practice, the "link" object can shift in unpredictable ways after a grasp action, so we adjust the plan after completion of the grasp action. We add noise to the parameters of the trajectory distribution associated with the subsequent `align` and `place` actions and replan. In the real robot experiment we omit this step due to the lack of accurate position information once the object is in the gripper.

The current implementation of the planner is single-threaded. The single largest inefficiency was detecting collisions, followed by computation of inverse kinematics. Particularly in scenes with more obstacles, both of these are very important: inverse kinematics are required to adapt trajectories to different possible grasp points, and accurate collision detection guarantees safe execution. As inverse kinematics and collision detection are outside the purview of this paper, we did not focus on efficiency.

A. Simulation Experiments

We collected three demonstrations of each of the different skills with a dynamic simulation of the Barrett WAM arm.

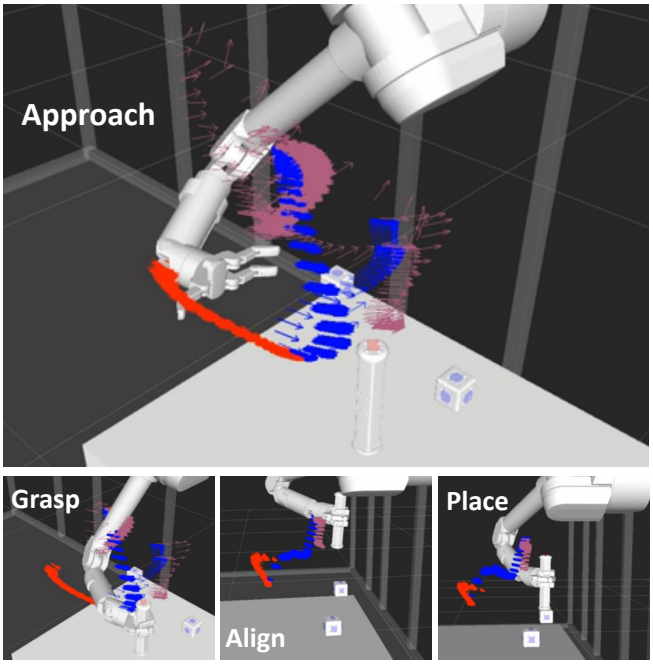


Fig. 5: Graphic showing example plan for the simple structure assembly task discussed here. Different colors indicate `approach`, `align`, and `place` actions. The planner has selected the leftmost node, and chose to grasp the link from the right.

We then place these pieces in different positions in the environment, and validated our method by performing the task in different locations. The results of one performance in a novel environment are shown in Fig. 5.

To create a model of each of these skills, we collect three demonstrations of the object manipulation action using the same grasp, with two of the Barrett Hand's fingers on the left side of the link and one on the right. The simulated WAM arm was teleoperated with a Razer Hydra to collect training data. The user provided examples of three different grasps: a direct approach and approaching from the left or the right. The user specified $p_d(\texttt{approach}|\texttt{w}_0)$ to indicate a preference for a direct approach.

We perform our task on scenes with one link and two nodes at different positions, and demonstrate task effectiveness for 10 trials with different configurations of the world. The key measure of performance is how easily we can add extra training data to our model and how close these results will be to the target mate. We set $\alpha = 0.5$ and used $M = 200$ trajectories, with a maximum of 15 iterations. Our full algorithm used a depth of $H = 5$: a long enough horizon to plan the whole assembly task. Average likelihood of sampled trajectories converged exponentially as we proceeded through various iterations. Fig 6 shows distance to an ideal final mate after outliers were removed.

By way of comparison, we remove one or both of two parts of our algorithm. We use a single randomly selected task plan ("no options" in Fig. 6). We also compare against the case where our planner only examines the currently available

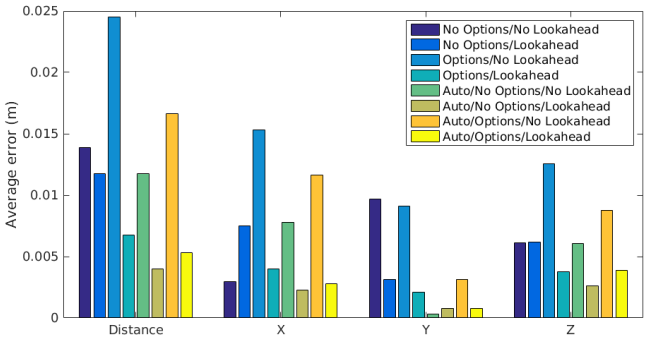


Fig. 6: Plot showing absolute error in distance, x, y, and z from "perfect" mate position between link and the selected node. Our full algorithm ("Options/Lookahead") achieved high mate accuracy, roughly equivalent to the version with no options, and was able to complete the task in more challenging scenarios.

actions, setting $H = 1$ ("no lookahead" in Fig. 6). The "no options/no lookahead" case functions as our baseline: it uses the algorithm in Sec. IV-A to reproduce an action based on a GMM. While our approach is technically unconstrained, due to the sampling method we implicitly constrain the trajectory search to a feasible set of valid trajectories. To demonstrate how we can improve performance by improving action models, a human user selected three successful trials from the automatic performance of this task and added them to the model ("auto" in Fig. 6).

Table I shows the number of planning failures associated with different environments. These are cases where the algorithm failed to find a trajectory with nonzero probability under the expert distributions defining each of our actions. Without the full algorithm, either the robot often cannot find a solution that will accomplish the task or performance is significantly degraded. The case where there are options and no lookahead is a good example. While the robot is almost always able to find a plan in this situation, the quality of plans is far worse, as shown by Fig. 6. In the higher-performing "Auto" case, the robot was always able to find a plan but few of these plans were successful: only 4/10 achieved high-quality mates, and several outliers fell off the node and the table completely. This is because without knowledge of the `place` and `release` actions, the `align` action will often not terminate in a good state to complete the task.

Fig. 6 shows a comparison on successful trials in different environments without obstacles. The full algorithm was highly reliable and accurate, achieving less than 1 cm of placement error. Other versions of the algorithm made mistakes that planning alone could not recover from. In addition, performance of all versions of the algorithm showed improvement when extra data was added, though the full version of the algorithm was still better and more flexible.

We also introduced different obstacles into the environment. In these cases, the algorithm is able to avoid these objects and still complete its required task. Figure 7 shows examples of these results. Since the planner removes paths that are in collision, this restricts the set of feasible trajectories. As in the upper left of Figure 7, the most likely action in a particular scenario might be an approach from a particular direction. Once this grasp is blocked, the planner can either attempt a less likely trajectory that results in that grasp, or it can approach from a different direction. This tradeoff illustrates why our approach is more powerful than adding a potential field term to action primitives as in [18] or similar work.

	Expert Data Only	With Auto Data
No Options/No Lookahead	7	4
No Options/Lookahead	3	1
Options/No Lookahead	1	0
Options/Lookahead*	0	0

TABLE I: Number of failures when generalizing to novel environments. (*) indicates the full algorithm. Columns represent whether model was taught using only expert demonstrations or whether extra data was added from successful executions.

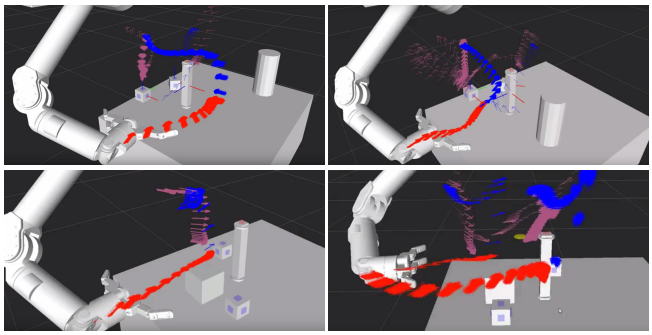


Fig. 7: Performance of the planner in different environments with the addition of obstacles. The planner chooses paths that are consistent with taught actions as much as possible.

B. Real Robot Experiments

Finally, we explore how we can use human preferences to improve our planning. The UR5 was given the option of grabbing either of two links or a node object and combining them to create the same structure as in the simulation experiments. Demonstrations were provided in which the robot grasped a node and mated it with a link, and in which the robot grasped a link and mated it with a node. The object localization technique described by Li et al. [35] was used to determine the poses of all objects in the scene.

Our system intelligently selected whether to grasp the link or the node, and it was able to select which face of the link to grasp based on feasibility and presence of other obstacles. The UR5 had a fairly limited workspace and has a limited ability to interact with objects when compared to the Barrett WAM arm used in the simulation experiments, making this a more challenging problem. However, the robot was able to grasp both node and link objects and complete the task. Our video supplement provides examples of the UR5 performing this task in different configurations, as well as an overview of the algorithm and videos of the simulation experiments.

In particular, when presented with both link and node objects in different orientations, the robot was able to correctly select available faces not blocked by other obstacles. If the node was better aligned with the robot's gripper, then the algorithm chose to grasp the node; if one of the links was better aligned, it would grasp this link.

VI. Conclusions

We have described a practical approach for task and motion planning based on models of skills grounded from expert demonstrations of skills. By representing actions as probability distributions learned from expert demonstrations, we create a framework that allows us to describe a broad range of actions and combine them to accomplish a task. We validated this approach with experiments in a structure assembly domain both in simulation and in a real robot.

While we did not address efficiency in the implementation used in this paper, this is a serious concern moving forward, and we will examine strategies for decreasing the number of costly trajectory evaluations and collision checks. In addition, we will apply our planner to larger and more complex tasks.

References

[1] E. Plaku and G. D. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5002–5008.

[2] J. Wolfe, B. Marthi, and S. J. Russell, "Combined task and motion planning for mobile manipulation." in *ICAPS*, 2010, pp. 254–258.

[3] V. Shivashankar, K. N. Kaipa, D. S. Nau, and S. K. Gupta, "Towards integrating hierarchical goal networks and motion planners to support planning for human-robot teams," 2014.

[4] F. Lagriffoul, D. Dimitrov, J. Bidot, A. Saffiotti, and L. Karlsson, "Efficiently combining task and motion planning using geometric constraints," *The International Journal of Robotics Research*, p. 0278364914545811, 2014.

[5] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI Communications*, no. Preprint, pp. 1–12.

[6] J. Bajada, M. Fox, and D. Long, "Temporal planning with semantic attachment of non-linear monotonic continuous behaviours," in *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, 2015, pp. 1523–1529.

[7] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[8] P. Pastor, M. Kalakrishnan, F. Meier, F. Stulp, J. Buchli, E. Theodorou, and S. Schaal, "From dynamic movement primitives to associative skill memories," *Robotics and Autonomous Systems*, vol. 61, no. 4, pp. 351–361, 2013.

[9] N. Krüger, J. Piater, F. Wörgötter, C. Geib, R. Petrick, M. Steedman, A. Ude, T. Asfour, D. Kraft, D. Omrcen, *et al.*, "A formal definition of object-action complexes and examples at different levels of the processing hierarchy," *PACO-PLUS Technical Report, available fro m http://www. paco-plus. org*, 2009.

[10] M. Wachter, S. Schulz, T. Asfour, E. Aksoy, F. Worgotter, and R. Dillmann, "Action sequence reproduction based on automatic segmentation and object-action complexes," in *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*. IEEE, 2013, pp. 189–195.

[11] S. R. Ahmadzadeh, A. Paikan, F. Mastrogiovanni, L. Natale, P. Kormushev, and D. G. Caldwell, "Learning symbolic representations of actions from human demonstrations," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3801–3808.

[12] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization*. Springer, 2004.

[13] M. Kobilarov, "Cross-entropy motion planning," *International Journal of Robotics Research*, vol. 31, no. 7, pp. 855–871, 2012.

[14] K. Welke, P. Kaiser, A. Kozlov, N. Adermann, T. Asfour, M. Lewis, and M. Steedman, "Grounded spatial symbols for task planning based on experience," in *13th International Conference on Humanoid Robots (Humanoids). IEEE/RAS*, 2013.

[15] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 2, pp. 286–298, 2007.

[16] S. Dong and B. Williams, "Motion learning in variable environments using probabilistic flow tubes," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1976–1981.

[17] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive Motion of Animals and Machines*. Springer, 2006, pp. 261–280.

[18] D. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *8th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2008, pp. 91–98.

[19] A. M. Ghalamzan E., C. Paxton, H. G. D., and L. Bascetta, "An incremental approach to learning generalizable robot tasks from human demonstration," in *Proceeding of IEEE International Conference on Robotics and Automation*. IEEE, 2015.

[20] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 3232–3237.

[21] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3828–3834.

[22] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," *arXiv preprint arXiv:1206.4621*, 2012.

[23] P. Englert, A. Paraschos, J. Peters, and M. P. Deisenroth, "Model-based imitation learning by probabilistic trajectory matching," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1922–1927.

[24] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning." in *AISTATS*, 2011, pp. 182–189.

[25] W. Masson and G. Konidaris, "Reinforcement learning with parameterized actions," *arXiv preprint arXiv:1509.01644*, 2015.

[26] S. Niekum, S. Chitta, A. G. Barto, B. Marthi, and S. Osentoski, "Incremental semantically grounded learning from demonstration." in *Robotics: Science and Systems*, vol. 9, 2013.

[27] S. Manschitz, J. Kober, M. Gienger, and J. Peters, "Learning to sequence movement primitives from demonstrations," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 4414–4421.

[28] D. Kappler, P. Pastor, M. Kalakrishnan, M. Wüthrich, and S. Schaal, "Data-driven online decision making for autonomous manipulation," 2015.

[29] 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)*. IEEE, 2014, pp. 639–646.

[30] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *International Joint Conference on Artificial Intelligence*, 2015.

[31] J. Bohren, C. Papazov, D. Burschka, K. Krieger, S. Parusel, S. Haddadin, W. L. Shepherdson, G. D. Hager, and L. L. Whitcomb, "A pilot study in vision-based augmented telemanipulation for remote assembly over high-latency networks," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3631–3638.

[32] M. Helmert, "The fast downward planning system." *J. Artif. Intell. Res.(JAIR)*, vol. 26, pp. 191–246, 2006.

[33] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[34] R. Smits, "KDL: Kinematics and Dynamics Library," http://www.orocos.org/kdl.

[35] C. Li, J. Bohren, E. Carlson, and G. D. Hager, "Hierarchical semantic parsing for object pose estimation in densely cluttered scenes," 2016, to appear.