To appear in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2021).

A Sampling-based Motion Planning Framework for Complex Motor Actions

Shlok Sobti[1], Rahul Shome[1], Swarat Chaudhuri[2], and Lydia E. Kavraki[1]

***Abstract*— We present a framework for planning complex motor actions such as pouring or scooping from arbitrary start states in cluttered real-world scenes. Traditional approaches to such tasks use dynamic motion primitives (DMPs) learned from human demonstrations. We enhance a recently proposed state-of-the-art DMP technique capable of obstacle avoidance by including them within a novel hybrid framework. This complements DMPs with sampling-based motion planning algorithms, using the latter to explore the scene and reach promising regions from which a DMP can successfully complete the task. Experiments indicate that even obstacle-aware DMPs suffer in task success when used in scenarios which largely differ from the trained demonstration in terms of the start, goal, and obstacles. Our hybrid approach significantly outperforms obstacle-aware DMPs by successfully completing tasks in cluttered scenes for a pouring task in simulation. We further demonstrate our method on a real robot for pouring and scooping tasks.**

I. Introduction

Robots deployed in dynamic, real-world environments often need to perform manipulation tasks in which the objective extends beyond reaching a goal configuration at the end of the motion. For instance, consider the action of *pouring water into a bowl* (Fig. 1). Here, it is not enough to place the manipulator above the bowl; we must also ensure that water does not get spilled outside the bowl. Planning such *complex motor actions* [1] is a challenging problem which traditional motion-planning algorithms cannot model.

A canonical approach to modeling complex motor actions is dynamic motion primitives (DMPs) [1], [2]. Here, one learns reusable modules to approximate behaviours from a single human demonstration. The demonstration is modeled as a dynamical system, features of which are learned, allowing the recreation of trajectories similar to the demonstration for novel starts and goals. More recently, DMPs methods [3] have been designed to address obstacle avoidance.

Deploying DMPs in real-world environments remains a challenge for two reasons. Firstly, the additional hard constraint of obstacle avoidance might conflict with the underlying task objectives, as in Fig. 1 where obstacles block the approach towards the target bowl in a pouring task. The presence of obstacles also leads to the possibility of local-minima when using potential-based obstacle avoidance strategies [3]. Current attempts to avoid obstacles based on DMPs face challenges in increased clutter. Secondly, the

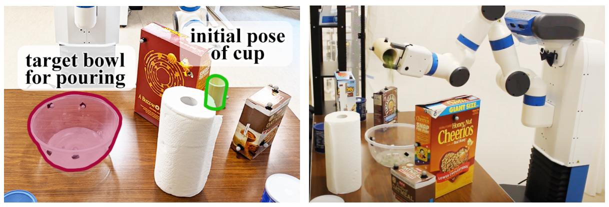

Fig. 1. A pouring demonstration in a cluttered tabletop environment. The robot has to execute a motion that *pours* the contents of the cup into the bowl, while avoiding the clutter in the scene.

problem at hand can be significantly different from the training demonstration. For instance in Fig. 1 the cup and bowl can be situated differently from the poses from the demonstration and new obstacles might be present. In that case the DMP adapts the motion but can introduce artifacts that affect task success, such as tilting the cup before it reaches the bowl. If a DMP is specified for the end-effector, it might also be difficult to follow continuously using inverse kinematics solvers. In contrast, computing obstacle-free motions between general start-goal pairs for high dimensional robots is a strength of sampling-based motion planning [4], [5].

Instead of a purely sampling-based or DMP-centric approach, we propose a *hybrid* motion planning framework for complex motor actions. Our central insight is that DMPs and sampling-based motion planning algorithms have complementary strengths. Sampling-based algorithms excel at generating collision-free motions connecting start and goal configurations in well-modeled problems for high dimensional systems. On the other hand, DMPs are a better fit than sampling-based algorithms for settings in which the validity, constraints, and objective of the motion cannot be easily modeled but can be deduced from demonstrations, as is the case in complex motor actions.

We propose a novel general framework that can discover complex motor actions that achieve an underlying task objective in cluttered scenes. Our framework divides the motion plan into two parts. The first constitutes easily modeled motions leading to regions that are likely to yield high quality, valid DMP motions. This part of the plan avoids obstacles, and can be effectively discovered using a sampling-based planner. We build on a multi-goal motion planner [6] capable of efficiently reaching promising regions from where DMPs are attempted. The second part of the motion is a complex motor action that is generated using an obstacle-aware DMP [3]. For the specific task we consider a DMP that is trained from a demonstration of the task. The integration proposed by our framework allows us to solve general cluttered problems that cannot be addressed by state-of-the-art complex motor action generation schemes [3]. We

[1]SS, RS and LEK are affiliated with the Department of Computer Science, Rice University, Houston. {sobti, rahul.shome, kavraki}@rice.edu. [2]SC is affiliated with the University of Texas, Austin {swarat@cs.utexas.edu}. This work is supported in part by NSF RI 2008720, NSF IIS 1718478, NSF CCF 1918651, NSF CCF 1704883, and Rice University Funds.

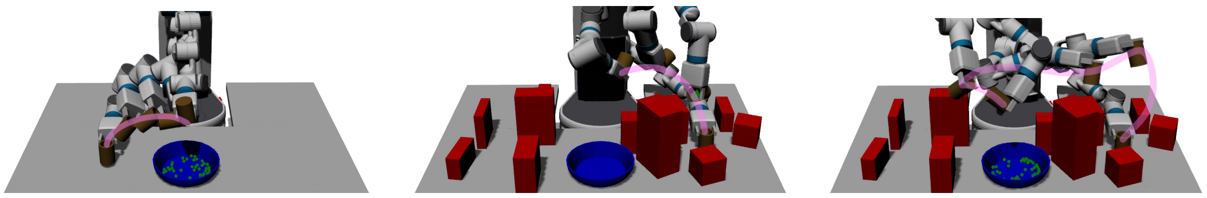
Fig. 2. The key problem being addressed in the current work is highlighted in a pouring task. ***Left:*** A DMP in an empty scene for a problem that resembles the training demonstration succeeds. ***Middle:*** An obstacle-aware DMP [3] avoids the clutter but fails at the task because it tilted the cup too early. ***Right:*** Our proposed hybrid method motion plans to different regions from where DMPs are evaluated. Here planning avoided all the obstacles to the right of the table, and a DMP was discovered from the left of the table which successfully poured all of the contents of the cup into the bowl.

evaluate our framework for a pouring task in randomized benchmarks where the start and target of the action are varied. As the extent of clutter in the tabletop scene is increased our hybrid solution significantly outperforms an obstacle-aware DMP-based method [3] in pouring the contents of a cup into a bowl. Fig. 2 illustrates a case where our approach is successful in a cluttered environment. Demonstrations of pouring and scooping actions on a *Fetch* robot highlight the applicability of our method to real world problems.

The rest of the paper is organized as follows. Section II provides an overview of related work. Section III gives a formal description of the problem statement. Section IV describes the details behind the hybrid planning strategy. Section V describes our experiments and real-world demonstrations.

II. Related Work

Modelling complex motor actions: Various lines of work have looked at the problem of generating complex motions for robotic systems that exhibit desired task-specific characteristics. Splines, potentials and other optimization techniques [7]–[10] are commonly used tools. However these techniques require well-defined optimization objectives which might not always be available especially in the case of complex motor actions. Human demonstrations [11] have been suggested as an effective reference for such motions. To this end, several learning/statistical tools have been proposed [12]–[14]. However, these often require large demonstration datasets which can be cumbersome to collect in learning-from-demonstration settings. Assuming that the primitives can be neatly parametrized, it has been shown that learning parameters for effective actions can provide increased success rates [15]. Our work in contrast deals with arbitrary primitives while allowing sampling-based motion planners to help alleviate the complexity of the action.

DMPs: DMPs [1], [2], [16] were proposed as a trajectory modeling framework to learn representations from single demonstrations. While DMPs are effective at encoding stylistic behaviours, they provide no guarantees of resolving into intended motion. Generalization to new queries is dependent on hyperparameters and the relative start and goal positions. This gap becomes even more pronounced in cluttered environments where the system must avoid obstacles while maintaining desired kinematic profiles. Such limitations are shared by all current available representation tools - any query vastly different from the dataset distribution is likely to yield low success. Since DMPs are a well-formulated and canonical modeling tool, this paper focuses on them.

DMPs with obstacle avoidance: The problem of obstacle avoidance is usually addressed by introducing additional potential fields associated with the obstacles [3], [17]. This is likely to lead to local-minima in high clutter environments. Further, these potentials must be carefully designed based on the task and environment to ensure that the local perturbations don't conflict with the underlying task objectives (if even such a potential field exists). Challenging clutter in scenes makes it difficult to respect the desired motion profiles while avoiding obstacles. The state-of-the-art obstacle avoidance proposes a volumetric scheme using superquadratic potentials [3]. Our work compares against the strategy defined in this approach.

Traditional motion planners: On the opposite end of the spectrum, sampling-based planners [4], [5] have proven to be effective planners in high-dimensional spaces. A particular extension to traditional tree-based planners [6], relevant to this work, allows for the lazy evalution of multiple goals, i.e. it will resolve to a grounding of the goal that is the 'easiest' to find. Sampling-based planners only allow for constrained point-to-point (or point-to-region) queries and fall short of effectively modeling complex motion characteristics that DMPs are capable of. Trajectory optimization methods [10], [18] avoid the sampling step and attempt to deform trajectories constrained by some convex objective function. Tree based sampling planners have benefited from incorporating optimization planners to exploit local domain information [19]. Methods such as splines require a well-formed objective function which is often non-trivial to construct. Our work builds upon the idea of extending sampling-based planners using techniques that can resolve arbitrary motions.

Motion primitives alongside sampling-based planners: Motion primitives have been used to bias the sampling strategy to acheive motions simlar to the the primitive [20]. We remark that the final solution is only as good as the inital primitive grounding and the sampling risks loosing key characteristics. Another class of work has proposed using primitives for the exploration of the configuration space (C-Space) in sampling-based planners [21], [22]. These methods propose variants to tree-based planning algorithms that use primitives to grow the tree. These methods are typically interested in using primitives to speed-up the motion-planning problem, rather than the successful execution of a desired motor action. Similar ideas have been explored in the context of grasp planning [23]. Simultaneous grasp and motion-planning [24] has considered coupling grasping attempts with tree-based planners. Though these hybrid methods build on sampling-based planning schemes, the objective is still to plan to goal configurations or end-effector poses. In contrast our proposed hybrid framework accomplish underlying task objectives instead of simply reaching a goal, by using DMPs from different parts of the search space to express motion profiles of task demonstrations.

III. Problem Formulation

A robotic arm r has a d-dimensional C-Space $\mathcal{Q} \subset \mathbb{R}^d$. The workspace $\mathcal{W} \in SE(3)$ also contains obstacle geometries, causing robot configurations $x \in \mathcal{Q}$ to be invalid through collisions or collision-free, creating subsets $\mathcal{Q}_{\text{obs}}$ and $\mathcal{Q}_{\text{free}}$. The robotic arm has a hand geometry called an *end-effector*. A point in the workspace is denoted by a pose $p \in \mathcal{W}$. There exists a mapping (forward kinematics) from a configuration to an end-effector pose, $FK : \mathcal{Q} \to \mathcal{W}$, and the inverse mapping is $IK : \mathcal{W} \to \mathcal{Q}$. A sequence of motions performed by a robot called a *trajectory*, is represented by a parameterized curve in the C-Space, $\pi : [0, 1] \to \mathcal{Q}$. Here $\pi(0)$ and $\pi(1)$ denote the start and end of the trajectory respectively. Let $\pi(t' : t'')$ denotes a segment of the trajectory between time parameter instants t' and t''. For every C-Space trajectory there exists a corresponding workspace end-effector trajectory $\bar{\pi}$, and vice versa. We are often interested in generating robotic motions that exhibit specific characteristics, or obey certain constraints for their C-Space, or workspace trajectories. In this work we want to plan for complex motor actions.

Definition 1 ***Motion Planning*** The general motion planning problem is defined by a start configuration $x_0 \in \mathcal{Q}_{\text{free}}$, and a goal set $\mathcal{Q}_{\text{goal}} \subset \mathcal{Q}_{\text{free}}$. A solution trajectory π is a parametrized curve such that $\pi : [0, 1] \to \mathcal{Q}_{\text{free}}, \pi(0) = x_0, \pi(1) \in \mathcal{Q}_{\text{goal}}, \pi(i) \in \mathcal{Q}_{\text{free}} \forall i \in [0, 1]$. The motion planning subroutine is defined as

$$\pi_{\text{MP}} \leftarrow \text{MP}(x_0, \mathcal{Q}_{\text{goal}}, \mathcal{Q}_{\text{free}}).$$

Of interest to us are motions which represent complex motor behaviors [1] (Sec 2.1) referred to as **complex motor actions**, associated with underlying task objectives.

Arbitrary human-centric motor task objectives are difficult to model with engineered constraints and goals. *For instance in pouring such a goal function might measure the amount of cup contents that end up in the bowl, and the motion ends at an end-effector pose right above the bowl.* We assume that for a task, we can estimate this from an action demonstration (π_{demo}[1]) that successfully achieves the task objective.

Given two trajectories, their similarity is calculated using a spatio-temporal error measure, *dynamic time warping (DTW)* [25], [26] as $\text{D}(\pi_1, \pi_2)$. The choice of DTW allows for temporal flexibility, a more relaxed error measure compared to euclidean distance. Assume that two trajectories are *similar* if $\text{D}(\pi_1, \pi_2)^{-1} > \Gamma$ for threshold Γ.

Definition 2 ***Complex Motor Action from Demonstration*** Given a demonstration of a successful complex motor action π_{demo}, a trajectory π, similar to π_{demo}, i.e., $\text{D}(\pi, \pi_{\text{demo}}) < \Gamma$, is considered a complex motor action from demonstration.

Definition 3 ***Complex Motor Action Generation*** Given a task-specific goal p_{goal}, and a start x_0, a module capable of generating a complex motor action from demonstration outputs π_{CMA} such that $\text{D}(\pi_{\text{CMA}}, \pi_{\text{demo}}) < \Gamma,\ \pi_{\text{CMA}}(0) = x_0,\ FK(\pi_{\text{CMA}}(1)) = p_{\text{goal}}$.

$$\pi_{\text{CMA}} \leftarrow \text{CMA}(x_0, p_{\text{goal}}, \mathcal{Q}_{\text{free}}).$$

Note that it's not necessary that the complex motor action trajectory π_{CMA} remains collision-free. For that we introduce the stronger guarantees afforded by planning. We now define the primary problem of *planning for complex motor actions.*

Problem Definition

***Input*:** The problem specification outlined by

- a demonstration of a successful action execution, π_{demo}
- a starting configuration of the robotic arm x_0
- a task objective that defines a goal region $\mathcal{Q}_{\text{goal}}$

***Output*:** A solution trajectory π such that,

- $\pi(0) = x_0, \pi(1) \in \mathcal{Q}_{\text{goal}}$
- $\pi(t) \in \mathcal{Q}_{\text{free}} \quad \forall t \in [0, 1]$
- π *includes* a sequence of motions that fulfills the task, i.e., $\exists t', t'' \in [0, 1]\ s.t.\ \text{D}(\pi(t' : t''), \pi_{\text{demo}})^{-1} > \Gamma$.

IV. Approach

We propose a hybrid planning framework to compute complex motor actions. The key idea is to build on a sampling-based motion planning algorithm that starts growing a search-tree from the start configurations eventually covering the C-Space. During this exploration, a complex motor action generation module is invoked from different parts of the C-Space. This is done to compute motions, from various configurations to a task-specific goal region, similar to the demonstration. Such motions are only accepted if they remain collision free with obstacles in the scene. We attempt multiple complex motor action rollouts, and ensure that we reach parts of the C-Space from where these motions are collision-free, thereby discovering a valid motion. Now we introduce *hybrid planning* in the context of complex motor actions.

Hybrid Planning: In our proposed hybrid framework the components involved are a hybrid planning module HP, a transition set of configurations $\mathcal{Q}_{\text{HP}}$, and a hybrid planning solution trajectory π_{HP}. The current task specification determines a set of target end-effector poses $\mathcal{W}_{goal} \subset \mathcal{W}$. It is up to the hybrid planning module to discover a trajectory that begins at x_0, and ends at a configuration that has the end-effector at a pose $p_{\text{goal}} \in \mathcal{W}_{goal}$. The trajectory has to be collision free. It is assumed that there exists a transition $\mathcal{Q}_{\text{HP}}$, such that motion planning can connect to a configuration within it. From the same such transition configuration a complex motor action generator should roll out a trajectory that is a) similar to the demonstration π_{demo} of the complex motor action, b) ends at a task specific goal region, and c) is collision-free.

$$\begin{aligned} \pi_{\text{HP}} &\leftarrow \text{HP}(x_0, \mathcal{W}_{goal}, \mathcal{Q}_{\text{free}}) \\ s.t.,\ \exists \mathcal{Q}_{\text{HP}} \subset \mathcal{Q}_{\text{free}}, \quad &\exists t \in [0, 1), \pi_{\text{HP}}(t) \in \mathcal{Q}_{\text{HP}}, \\ \pi_{\text{HP}}(0 : t) &\leftarrow \text{MP}(x_0, \pi_{\text{HP}}(t), \mathcal{Q}_{\text{free}}), \\ \pi_{\text{HP}}(t : 1) &\leftarrow \text{CMA}(\pi_{\text{HP}}(t), p_{\text{goal}}, \mathcal{Q}_{\text{free}}) \\ s.t.\ FK(\pi_{\text{HP}}(1)) &= p_{\text{goal}} \in \mathcal{W}_{goal} \end{aligned}$$

It is the responsibility of the hybrid planning framework *to discover transition regions $\mathcal{Q}_{\text{HP}}$, use a motion planner to*

[1]The demonstration might be recorded in either $\mathcal{Q}$ or $\mathcal{W}$.

achieve them, and execute valid motions from these regions to successfully complete the complex motor action. Our framework has two primary components, *a sampling-based forward search tree* and an *obstacle aware DMPs*. The first allows us to navigate to promising regions while avoiding obstacles and the second allows us to generate complex motor actions. The high-level approach is illustrated in Fig. 3.

A. Components of our Hybrid Framework

Sampling-based Forward-search Tree: We build a sampling-based forward search tree rooted at x_0 and sample to grow the tree across $\mathcal{Q}_{\text{free}}$ to compute $\pi_{\text{HP}}(0:t)$ that ends at a transition point $x_{end} = \pi_{\text{HP}}(t)$. A randomized tree-based exploration is *probabilistically complete*, i.e., given enough samples it will explore all reachable parts of $\mathcal{Q}_{\text{free}}$. The exploration of $\mathcal{Q}_{\text{free}}$ must be directed towards transition points from where different rollouts of the *complex motor action* can be attempted. The complex motor action generation module, when triggered from different regions of the $\mathcal{Q}_{\text{free}}$, exhibits varying rates of success. This gives rise to multiple goal regions with varying likelihood of successful execution of the complex motor action. We find previous work [6] particularly helpful in weighing and prioritizing different goal regions, although any sampling-based planner with similar functionality can be used. This allows for an adaptive exploration of the search space. For instance, consider a scenario where navigating to the region of highest promise is unreachable due to obstacles. The planner must adapt and advance towards alternative regions.

Obstacle-aware DMPs: Given the demonstration, π_{demo}, we build complex motor action representation using DMPs [1], [2] with state-of-the-art obstacle avoidance [3]. This serves as our underlying complex motor action generation module. Given workspace demonstration[2], $\bar{\pi}_{\text{demo}}$, we train a reusable complex motor action generation module, DMP. Leveraging the workspace-centric power of DMPs, such a module takes as input a starting state x_0, from where the end-effector pose p_{start} is recorded as $FK(x_0)$, a task relevant goal pose, $p_{\text{goal}} \in \mathcal{W}$, and returns a workspace trajectory $\bar{\pi}$ that is similar to $\bar{\pi}_{\text{demo}}$ i.e., $\text{D}(\bar{\pi}, \bar{\pi}_{\text{demo}}) < \Gamma$. The motions of the robot ultimately exist in the C-Space so the inverse kinematics mapping IK can be used to obtain a C-Space trajectory π_{CMA} from $\bar{\pi}$. The workspace trajectories represent a sequence of poses $p \in SE(3)$ over time, where each the i^{th} $SE(3)$ coordinate is p^i. The DMP models each coordinate of a workspace trajectory as a second order dynamical system, with additional non-linear terms [2] (eq 2.1).

Training the DMP: Given a demonstration $\bar{\pi}_{\text{demo}}$, ending at $\bar{\pi}_{\text{demo}}(1)$, the terms for the i^{th} coordinate of an instantaneous demonstrated pose p_{demo}, and its higher order derivatives, these terms can be expressed as a second order system. Within this system, a *forcing function* f_{demo} is defined to capture the *motion profile* of $\bar{\pi}_{\text{demo}}$ as follows [2] (Eq 2.12):

$$f^i_{\text{demo}} = \tau^2 \ddot{p}^i_{\text{demo}} - \alpha(\beta(\bar{\pi}_{\text{demo}}(1)^i - p^i_{\text{demo}}) - \tau \dot{p}^i_{\text{demo}})$$

[2]In this work we focus on demonstrations arising from humans

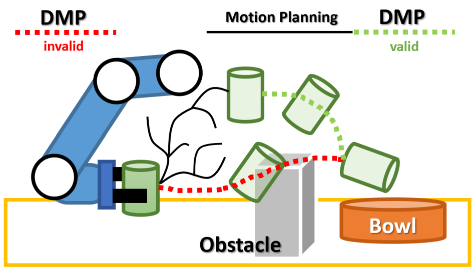

Fig. 3. The hybrid planning framework for a pouring action. We build a tree from the start, and invoke a DMP for complex motor action generation.

Here τ is the temporal scaling term, and α,β are the spring and damping constants. We choose to represent f^i_{demo} as a weighted linear combination of exponential basis functions, and use locally weighted regression to *learn the parameters*, from which we *learn* f^i_{demo}. This is used to generate $\bar{\pi}$.

Generating a complex motor action from demonstration: Given the new problem trying to reach p_{goal}, while following an action similar to $\bar{\pi}_{\text{demo}}$, we want to compute the i^{th} coordinate of instantaneous poses p^i along $\bar{\pi}$ modeled as

$$\tau \ddot{p}^i = \alpha(\beta(p^i_{\text{goal}} - p^i) - \dot{p}^i) + f^i_{\text{demo}} + \phi^i.$$

Note that this is essentially a reorganization of the second order system model, but using the current goal p_{goal}, and the pre-trained forcing function component f^i_{demo} to *force* the current motion to be similar to $\bar{\pi}_{\text{demo}}$ temporally and spatially. The term ϕ^i is an additional term responsible for potential-based obstacle avoidance [3]. The sequence of poses p generates $\bar{\pi}$, which is converted to π_{CMA} using IK.

Limitations of DMPs: DMPs, although effective modeling tools to capture motion features, are susceptible to errors in cluttered environments, or when the query vastly differs from the demonstration. These scenarios are likely to yield warped paths that violate the underlying task objective. Furthermore the output workspace path $\bar{\pi}$ might not have a continuous mapping to $\mathcal{Q}$ if the IK is infeasible along the way, leading to a discontinuous π_{CMA}. These limitations make relying exclusively on DMPs ineffective and necessitate the incorporation into a sampling-based planning framework.

B. Algorithmic Details

Algorithms 1 and 2 detail the high-level steps of our approach. Alg 2 is responsible for exploring the different groundings of the complex motor action, while Alg 1 builds a sampling-based tree to reach the previously explored transition regions similar to previous work [6]. It adaptively grows towards regions from where DMP rollouts are promising while biasing towards easier to reach transitions.

Sampling candidates: The limitations of DMPs underscores the importance of deliberately choosing promising transition configurations as points to transition between motion planning and DMPs. In this work we choose to sample these points from a task-space region (TSR) [27] that is defined from the complex motor action objective. A TSR is essentially a set of constraints in $SE(3)$ on the end-effector of the manipulator, and maps to the configuration space of the robot. For instance the TSR for pouring can be defined as a region around the bowl, with the cup frame fixed to be upright.

Algorithm 1: HybridPlanner

```
input  : Start configuration $x_0$, Task specific goal
         region $\mathcal{W}_{goal}$, Demonstration $\bar{\pi}_{\text{demo}}$, $\mathcal{Q}_{\text{free}}$
output : Solution trajectory $\pi$
1  trainDMP($\bar{\pi}_{\text{demo}}$)
2  $\mathbb{T}_{heap} \leftarrow$ TransitionSampler($\mathcal{W}_{goal}, \mathcal{Q}_{\text{free}}$)
3  $\pi \leftarrow \emptyset$
4  while not done do
5      if GoalBiasing() then $x_{rand} \leftarrow$ GetTop($\mathbb{T}_{heap}$)
6      else $x_{rand} \leftarrow$ RandomSample()
7      $x_{tree} \leftarrow$ SelectNode($x_{rand}$)
8      $x_{new} \leftarrow$ Propagate($x_{tree}, x_{rand}$)
9      if IsValid($x_{tree}, x_{new}$) then
10         GrowTree($x_{tree}, x_{new}$)
11         if $\text{x}_{\text{new}} \in \mathbb{T}_{\text{heap}}$ then
12             $\pi \leftarrow$ ReconstructMotorAction($x_{new}$)
13             return $\pi$
14         if GoalBiasing() then
               Promote($\mathbb{T}_{heap}, x_{rand}$)
15     else if GoalBiasing() then
           Penalize($\mathbb{T}_{heap}, x_{rand}$)
16 return $\pi$
```

Algorithm 2: TransitionSampler

```
input  : Task specific goal region $\mathcal{W}_{goal}$, $\mathcal{Q}_{\text{free}}$
output : A sorted heap $\mathbb{T}_{heap}$ of transition states
1  $\mathbb{T}_{heap} \longleftarrow \emptyset$ while not done do
2      $x_{transition} \leftarrow$ SampleTransitionPoint($\mathcal{Q}_{\text{free}}$)
3      $p_{goal} \leftarrow$ SampleGoalPose($\mathcal{W}_{goal}$)
4      $\pi_{DMP} \leftarrow$ DMP($x_{transition}, p_{goal}, \mathcal{Q}_{\text{free}}$)
5      $W_{quality} = $ Score($\pi_{DMP}$)
6      if $W_{quality} > \Gamma$ and IsValid($\pi_{DMP}$) then
7          $\mathbb{T}_{heap} \leftarrow \mathbb{T}_{heap} \cup x_{transition}$
8      Sort($\mathbb{T}_{heap}$)
9  return $\mathbb{T}_{heap}$
```

This is theoretically a lower dimensional manifold within $\mathcal{Q}$ that can still be sampled [28]. We use the TSR to sample in $\mathcal{Q}_{\text{HP}}$ (line 2 of Alg 2). We iteratively relax the TSR constraints if valid sampling continually fails - this ensures the exploration of $\mathcal{Q}$. The variability introduced by sampling allows us to explore configurations from where the DMP might have a clearer shot at the goal within clutter to allow a successful rollout. Sampling transition configurations also gives a motion planner a richer set of goals. Further, we also sample goal points, p_{goal} from another TSR (line 3 in Alg 2). The constraint on this is task-specfic, i.e., for the pouring action the bowl must end up over the bowl allowing some tolerances. Given $x_{transition}$ and p_{goal} we forward roll-out the trajectory using the pre-trained DMP, capable of obstacle avoidance [3] (line 4 in Alg 2). Fig. 4 illustrates this step. Note that not every DMP will be of the same quality. We want to score successful DMPs.

$\mathcal{Q}_{\text{HP}}$ along with the associated weights are fed into the sampling-based search tree (Alg 1). The planner keeps track of goal-biasing attempts, promoting goals that have high success rates while penalizing goals that the search tree in unable to advance towards (line 7,8 in Alg 1).

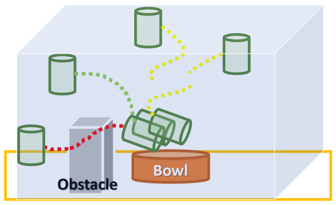

Fig. 4. Sampling $\mathcal{Q}_{\text{HP}}$ and rolling out DMPs. Some are invalid (red), some are incomplete or discontinuous (yellow), while only few are valid (green).

Scoring: Given the sampled set of transition points, we need to estimate the *quality* of each point with respect to the given DMP. The quality score must capture the underlying task objective and other measures like smoothness and continuity.

To allow for rotational invariance, the translational component of the DMP workspace trajectory $\bar{\pi}$ is mapped to a lower two-dimensional space, defined by the radial distance from the goal and the height with respect to the goal. Each dimension is then normalized to be within the $[0, 1]$ range. This is consistent with the local frame in which DMPs are rolled out [2](Sec 3.4). To account for the underlying task objectives, we compare this trajectory to a *template motion* $\bar{\pi}_{\text{demo}}$ which is achieved by rolling out the DMP under conditions similar to training, i.e. identical boundary points and non-existence of clutter. We compare the trajectories using *FastDTW* [26] distance D in $SE(3)$ to allow for temporal flexibility. The similarity is then the inverse of D.

Besides the shape similarity measure, we are also interested in a measure of continuity. We use TRAC-IK [29] with Distance as the null-space objective to map $\bar{\pi}$ to a C-Space trajectory π_{CMA}, similar to prior work [3]. A measure of discontinuity c_{dis} is the maximum $\mathcal{Q}$ jump along the trajectory waypoints. The continuity score is the inverse of c_{dis}. It is possible that IK fails completely at a waypoint. To account for this, a component of the score is allocated to the completion of IK along the waypoints. The percent of the path sucessfully mapped using IK is denoted as $c_{comp} \in [0, 1]$.

The total quality measure then is a weighted combination, $W_{quality} = w_1 \cdot \text{D}^{-1} + w_2 \cdot c_{dis}^{-1} + w_3 \cdot c_{comp}$. We threshold and reject all samples below a certain threshold Γ (line 5 in Alg 2). The transition point allowing successful a DMP is added to a sorted heap (line 6-8 in Alg 2).

Constructing action representations: In this work we are primarily interested in task objectives centered in $\mathcal{W}$(true for most manipulation actions). The DMP is trained via workspace demonstrations $\bar{\pi}_{\text{demo}}$. We developed an augmented reality (AR) application on a MagicLeap device that allows users to manipulate objects and record $SE(3)$ trajectories. AR provides a principled way to rapidly generate and test demonstrations [30].

V. Test Scenarios

We evaluate the performance of our hybrid approach against the state-of-the-art obstacle avoidance aware DMP in simulation settings for a pouring action.

To appear in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2021).

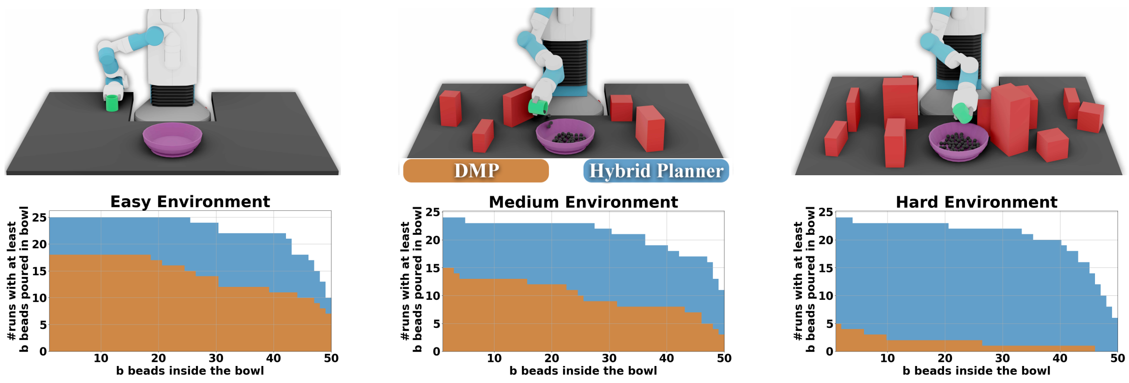

Fig. 5. *Top row:* The pouring benchmark setups for sampled scenes demonstrating examples of three degrees of clutter (*from left to right*) easy, medium, and hard. *Bottom row:* Histograms indicating the success of the action when executed using a obstacle aware DMP (orange) versus our hybrid approach (blue). For every point on the X-axis, the plots count the number of runs where more than the corresponding number of beads ended up inside the bowl.

A. Pouring Benchmark

We evaluate our hybrid planner approach on a pouring motion which is an instance of a complex motor action. Pouring, unlike point-to-point motion, has an underlying task objective, dependent on the motion characteristics. Further, since we want our system to be agnostic to the specific complex motor action, positions of the cup and bowl, and obstacles, handcrafting the motion apriori is infeasible.

Training setup: We train a DMP for the pouring motion, given a single pouring human demonstration. The pouring demonstration was conducted in a clutter-free environment with the cup starting $40cm$ away from the bowl such that the motion was within the dexterous reachability of the robot.

Experiment setup: We design three different environments with varying degrees of clutter, replicating different difficulty levels: easy, medium and hard. Difficulty here corresponds to the number and size of obstacles in the environment, as shown in Fig 5. For each of these environments, we randomly sample 25 different valid cup positions on the table such that there exists a valid grasping configuration to it. The cup is filled with 50 beads with properties of steel. The simulation setup is done on Gazebo [31]. (We compare our hybrid-planning strategy to DMPs rolled out with our implementation of volumetric obstacle avoidance [3].) The transition points are sampled from a TSR centered around the bowl, enforcing the cup stay close to upright. The goal is the cup inverted over the bowl at a height of 30cm. The planning component of our hybrid strategy, for this task, uses a constrained sampler [28] that keeps the cup upright.

Metrics: We chose to quantify performance based on *number of beads poured successfully*, *computation time* and *execution time*. The number of beads poured successfully captures the intrinsic success specific to the task and is most critical. Since these tasks have underlying objectives, measuring real world performance is the only fair evaluation method.

Results: The benchmarking results appear in Fig. 5 as cumulative histograms. The plots display the number of runs for which pouring resulted in at least b beads in the bowl. The performance improvement achieved by our method is especially significant in more clutter. Compared to obstacle aware DMPs, our method has significantly more runs with a higher number of beads successfully poured. For the runs with at least one bead successfully poured, we record the average computation and execution times, presented in Table I. Expectedly, our hybrid planning is slower than a single rollout of a DMP (solutions are still found within a few seconds), but our hybrid framework vastly outperform the DMP baseline in task success.

	Obstacle-aware DMP [3]		**Hybrid Planner**	
	computation(s)	execution(s)	computation(s)	execution(s)
easy	0.226	50.11	1.194	61.51
medium	0.240	57.07	5.082	69.03
hard	0.298	56.10	6.911	70.51

TABLE I

B. Real-world Tests (demonstrations in video)

We used the Fetch mobile manipulator to demonstrate *pouring* (Fig 1) and *scooping* (Fig 6) tasks in tabletop scenes. *Vicon* cameras were used to determine the object poses. Household objects were placed on the table to create clutter. The robot poured lentils to a bowl, and scooped beans from a plate.

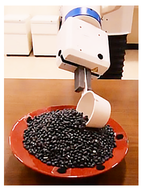

Fig. 6. Real world scooping demonstration.

VI. Discussion and Future Work

We have demonstrated in this work the benefits of combining the powers of sampling-based motion planning, and DMPs to plan for complex motor actions. Experimental validation of pouring actions in various amounts of simulated clutter show that our method succeeds at the task objective by pouring more of a cup's contents into a target bowl than a obstacle aware DMP baseline. The performance gulf grows more stark in scenes with more cluttering obstacles. There is a clear need for planning to overcome the shortcomings of DMPs in arbitrary real-world scenes. Demonstrations of both pouring and scooping further motivate our method as a way to execute such actions on real systems. The proposed work is a stepping stone that opens up possibilities for longer-horizon tasks, modeling more complex demonstrations, and framing such actions in the context of task planning, towards broader goals of designing more capable robots in human-centric tasks.

Acknowledgement - We thank Z. Kingston & C. Chamzas.

References

[1] S. Schaal, "Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics," *Adaptive Motion of Animals and Machines*, pp. 261–280, 2006.

[2] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models formotor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[3] M. Ginesi, D. Meli, A. Calanca, D. Dall'Alba, N. Sansonetto, and P. Fiorini, "Dynamic movement primitives: Volumetric obstacle avoidance," *2019 19th International Conference on Advanced Robotics, ICAR 2019*, pp. 234–239, 2019.

[4] L. E. Kavraki, P. P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[5] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[6] J. D. Hernandez, M. Moll, and L. E. Kavraki, "Lazy evaluation of goal specifications guided by motion planning," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, no. May, pp. 944–950, 2019.

[7] C. Lin, P. Chang, and J. Luh, "Formulation and optimization of cubic polynomial joint trajectories for industrial robots," *IEEE Transactions on Automatic Control*, vol. 28, no. 12, pp. 1066–1074, 1983.

[8] E. Magid, D. Keren, E. Rivlin, and I. Yavneh, "Spline-based robot navigation," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 2296–2301.

[9] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[10] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 489–494.

[11] S. Calinon, "Learning from Demonstration (Programming by Demonstration)," *Encyclopedia of Robotics*, pp. 1–8, 2018.

[12] J. Duan, Y. Ou, J. Hu, Z. Wang, S. Jin, and C. Xu, "Fast and stable learning of dynamical systems based on extreme learning machine," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 6, pp. 1175–1185, 2019.

[13] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[14] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," *Advances in Neural Information Processing Systems*, pp. 1–9, 2013.

[15] Z. Wang, C. R. Garrett, L. Kaelbling, and T. Lozano-Perez, "Learning compositional models of robot skills for task and motion planning," *ArXiv*, vol. abs/2006.06444, 2020.

[16] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance," pp. 2587–2592, 2009.

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

[18] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[19] S. Choudhury, J. D. Gammell, T. D. Barfoot, S. Srinivasa, and S. Scherer, "Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning," in *Proceedings of 2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Ed., May 2016, p. 4207 4214.

[20] K. Hauser, T. Bretl, K. Harada, and J. C. Latombe, "Using motion primitives in probabilistic sample-based planning for humanoid robots," *Springer Tracts in Advanced Robotics*, vol. 47, pp. 507–522, 2008.

[21] V. Vonasek, M. Saska, K. Kosnar, and L. Preucil, "Global motion planning for modular robots with local motion primitives," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2465–2470, 2013.

[22] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini, "Sampling-based optimal kinodynamic planning with motion primitives," *Autonomous Robots*, vol. 43, no. 7, pp. 1715–1732, 2019.

[23] M. T. Mason, "Toward Robotic Manipulation," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 1–28, 2018.

[24] N. Vahrenkamp and T. Asfour, "Simultaneous Grasp and Motion Planning," *IEEE Robotics & Automation Magazine*, no. JUNE, pp. 43–57, 2012.

[25] J. Kruskal and M. Liberman, "The symmetric time-warping problem: From continuous to discrete," *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, 01 1983.

[26] S. Salvador and P. Chan, "Fastdtw: Toward accurate dynamic time warping in linear time and space," 2004.

[27] D. Berenson, S. Srinivasa, and J. Kuffner, "Task Space Regions: A framework for pose-constrained manipulation planning," *International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[28] Z. Kingston, M. Moll, and L. E. Kavraki, "Exploring implicit spaces for constrained sampling-based planning," *The International Journal of Robotics Research*, vol. 38, no. 10-11, pp. 1151–1178, 2019.

[29] P. Beeson and B. Ames, "Trac-ik: An open-source library for improved solving of generic inverse kinematics," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 928–935.

[30] J. D. Hernandez, S. Sobti, A. Sciola, M. Moll, and L. E. Kavraki, "Increasing robot autonomy via motion planning and an augmented reality interface," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1017–1023, 2020.

[31] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.