NOD-TAMP: Multi-Step Manipulation Planning with Neural Object Descriptors

Anonymous Author(s)
Affiliation
Address
email

Abstract: Developing intelligent robots for complex manipulation tasks in household and factory settings remains challenging due to long-horizon tasks, contact-rich manipulation, and the need to generalize across a wide variety of object shapes and scene layouts. While Task and Motion Planning (TAMP) offers a promising solution, its assumptions such as kinodynamic models limit applicability in novel contexts. Neural object descriptors (NODs) have shown promise in object and scene generalization but face limitations in addressing broader tasks. Our proposed TAMP-based framework, NOD-TAMP, extracts short manipulation trajectories from a handful of human demonstrations, adapt these trajectories using NOD features, and compose them to solve broad long-horizon tasks. Validated in a simulation environment, NOD-TAMP effectively tackles varied challenges and outperforms existing methods, establishing a cohesive framework for manipulation planning. For videos and other supplemental material, see the project website: https://sites.google.com/view/nod-tamp/.

Keywords: Task and Motion Planning; Learning from Demonstration; Neural Object Representations

1 Introduction

Developing intelligent robots that can automate complex manipulation tasks in households or factories has been a longstanding goal for robotics and AI. Among the multitudes of challenges, three key factors stand out. We illustrate these challenges in a simulated tabletop cleaning task in Fig. 1. First, these tasks are often *long-horizon* and full of sequential dependencies. For example, the robot must reason about the best pose to grasp a mug in order to stow it in a tight cabinet, among other steps to organize the entire table. Second, the *contact-rich* manipulation steps, such as the process of stowing the mug, can make model-based planning intractable [1]. Finally, to be effective in broad environments, the robot must handle a wide *variation of object shapes* and *scene layouts*.

Task and Motion Planning (TAMP) [2, 3] has been the de facto solution for such problems because it can effectively resolve sequential dependencies through hybrid symbolic-continuous planning. However, TAMP systems typically require accurate, special-purpose perception and hand-engineered manipulation skills. This makes it difficult to apply these methods to previously unseen objects and tasks with complex dynamics such as contact-rich manipulation. Recent works have proposed to learn manipulation skills from demonstration [4, 5] to partially relax these constraints. However, their generalization ability remains bounded by the training data, which is costly and difficult to collect at scale [6].

At the same time, neural representation models trained on broad data have shown remarkable potential in enabling generalizable manipulation systems [7–10]. In particular, neural object descriptors (NODs) [8, 11, 12] emerged as a powerful tool to extract dense, part-level features that generalize across object instances. Simeonov and Du *et al.* [8] showed that Neural Descriptor Fields (NDF),

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

a type of NOD that encodes $\mathrm{SE}(3)$ poses relative to a given object, can adapt key-frame actions (e.g. grasp poses) for one object instance to others in the same object category (e.g. mugs), thereby achieving category-level generalization. However, despite this progress, existing NOD-based methods [8, 13, 14] are limited to adapting individual key-frame poses instead of solving a long-horizon task, and they struggle with contact-rich manipulation because they still rely on conventional motion planners to generate the approaching trajectories. Leveraging NODs to solve long-horizon tasks requires addressing a number of fundamental limitations, including planning long action sequences and generating trajectories for contact-rich manipulation.

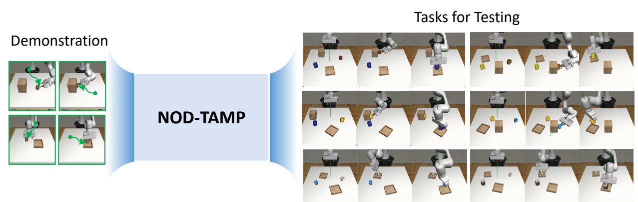

Figure 1: **Overview.** A subset of the diverse long-horizon tasks NOD-TAMP can solve with just a handful of demonstrations.

To address these limitations, we introduce NOD-TAMP, a TAMP-based framework that extracts adaptable skills from a handful of human demonstrations using NOD features and compose the skills to solve diverse long-horizon tasks. Central to NOD-TAMP is a skill reasoning module that composes short-horizon skills to solve novel long-horizon goals never seen in demonstration. To synthesize fine-grained manipulation trajectories and adapt to new objects, we propose a NOD-based trajectory adaptation module that can consistently adapt a recorded skill trajectory according to the observed objects. Finally, NOD-TAMP can flexibly switch between adapting recorded trajectories and using existing kinematic motion planning ability of a TAMP system to generalize to drastically different scene layout.

We evaluate NOD-TAMP with simulated multi-step manipulation tasks that evaluate different factors of generalization across long-horizon and contact-rich tasks, including object shapes, number of objects, scene layout, task length, and task objectives. We empirically demonstrate that NOD-TAMP can consistently solve the evaluation tasks, despite using just a small set of short-horizon demonstrations. NOD-TAMP also outperforms other methods [8, 15] that share a subset of its traits, highlighting the value of building a cohesive manipulation planning system.

2 Related Work

TAMP. Task and Motion Planning (TAMP) is a powerful paradigm for addressing long-horizon manipulation challenges by decomposing a complex planning problem into a series of simpler subproblems [2, 3, 16–18]. Nonetheless, TAMP techniques presuppose knowledge of the object models and the underlying kinodynamic systems. Such presuppositions can be limiting, particularly for real-world domains with diverse objects and steps that involve compelex physical processes such as contact-rich manipulation.

Learning for TAMP. Recent works have set to address such limitations by replacing hand-crafted components in a TAMP system with learned ones. Examples include environment models [19–22], skill operator models [4, 23], skill samplers [24, 25], and learning to imitate actions from TAMP supervisors [26–28]. However, these learned components are often limited to the tasks and environments that they are trained on. Two notable exceptions are M0M [29] and GenTP [30], but both methods plan with predefined manipulation skills and assume the skills are robust to variations in tasks and environments. In contrast, our work directly tackles the generalization challenge at the level of motion generation. Closely related to our work are methods that learn manipulation skills for TAMP systems [4, 31, 32]. However, the resulting systems remain bottlenecked by the generalizability of the skills, which are trained using conventional Reinforcement Learning [31] or Behavior Cloning [4, 32] algorithms. Instead, our work develops skills with object category-level generalization ability and integrates such skills with the existing planning ability of a TAMP system.

Learning from Human Demonstrations. Modern deep imitation learning techniques have shown remarkable performance in solving real-world manipulation tasks [6, 33–37]. However, the prominent data-centric view of imitation learning [6, 37, 38], i.e., scaling up robot learning via brute-force data collection, remains limited by the sample efficiency of the existing learning algorithms and the challenges in collecting demonstrations for long-horizon tasks in diverse settings. Other recent works have proposed to replay a small set of human demos in new situations to facilitate sample-efficient generalization [15, 39–44], but replay without adaptation can fail for novel object instances. Some other works leverage pretrained object representations to dramatically improve the generalization of policies given a handful of demonstrations [8, 10, 14]. However, these methods are limited to adapting a short skill [10] or a single manipulation action [8]. Our work develops a long-horizon planning framework that seamlessly integrates skills augmented with latent object representations into a classical TAMP framework.

3 Problem Formulation

We seek to apply NDFs [45], a type of Neural Object Descriptor, to robot manipulation. First, we review background on NDF for category-consistent frame estimation (Section 3.1). Then, we describe our problem setting (Section 3.2).

3.1 Neural Descriptor Fields (NDF)

NDF [8] were first proposed for category-invariant modeling of object rigid transformations. A NDF model $F(T \mid P)$ takes in an object shape that is represented as a point cloud $P \in \mathbb{R}^{N \times 3}$ and a set of query positions $\{R \cdot x_i + t \mid x_i \in X\}$. Let $T = [R \mid t] \in \text{SE}(3)$ be a rigid transformation and $X \in \mathbb{R}^{M \times 3}$ be a vector of query points. The NDF outputs features corresponding to the query points:

$$z \leftarrow F(T \mid P) \equiv \bigoplus_{x_i \in X} f(R \cdot x_i + t \mid P).$$

A key advantage of NDF features is that they are only related to the queries in the object's local frame, therefore a rigid transformation $R \in \text{SE}(3)$ applied to both the shape and the query pose has no affect on the feature:

$$F(R \cdot T \mid R \cdot P) = F(T \mid P).$$

NDFs are typically used to solve an inverse problem: recover a transformation T that corresponds to a query feature z. This can be framed as the following optimization problem and solved by 1) randomly initializing the transform, 2) backpropgating to compute the error gradient, and 3) iteratively moving along the gradient to minimize the error:

$$\text{NDF-OPTIMIZE}(F, P, z) \equiv \underset{T}{\text{argmin}} ||z - F(T \mid P)||.$$

3.2 Problem Setting

We address controlling a robot to perform multi-stage manipulation. The robot is tasked with manipulating one or more movable objects $o \in O$ in the world to achieve a goal. The robot observes RGB-D images, which it can process into segmented point clouds for each object $\mathcal{P} = \{o : P_o \mid o \in P\}$. Although we assume that we can detect the category of each object, critically, we do not assume instance detection or have a geometric model of any object (*e.g.* mesh).

The state of the world is described by robot configuration $q \in \mathbb{R}^d$ and frame state $S = \{o : T_w^o \mid o \in O\}$, where $T_{f_2}^{f_1} \in \text{SE}(3)$ represents a rigid transformation from frame f_1 to f_2 and w is the world frame. Let $S_{f_2}^{f_1}$ be shorthand for the rigid transformation from frame f_1 to f_2 in state S. The robot takes actions $a = T_a^e$ that correspond to target end-effector poses, where e is the end-effector frame. These task-space set points are converted to joint-space commands using Operational-Space Control (OSC) [46].

We are interested in re-purposing a set of demonstrations for a ensemble of tasks into manipulation policy that is effective in scenes with new objects and varying initial poses. To that end, we assume a dataset of demonstrations that is collected by human teleoperation or some other process. Let a demonstration trajectory $\tau = [\langle S_1, a_1 \rangle, ..., \langle S_h, a_h \rangle]$ be a sequence of state-action pairs $\langle S, a \rangle$. At training time, we assume that we can observe or estimate frame states; however, this assumption does not hold at test time.

3.3 Skill Demonstrations

In order to deploy demonstrations new 1) objects and 2) tasks, we need to understand more about the context behind each action involving which objects are interacting or are about to interact as well as qualitatively how they are interacting. In this work, we assume that each state-action pair $\langle S, a \rangle$ can be labeled with an interaction type l, a source movable object o, and target coordinate frame f, forming a tuple $\langle l, o, f, S, a \rangle$. Our technique will be to characterize the motion of o relative to f using NDFs and then adapt it to new circumstances. Figure 2 demonstrates a pair of insertion interactions involving pegs and holes that vary in geometry. Here, the movable object o is the peg and the target frame f is the hole.

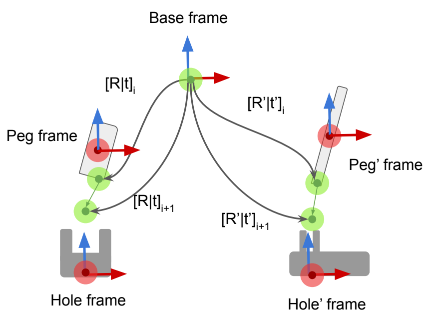

Figure 2: **Trajectory Adaptation.** Illustration of generating motion trajectory for test scenario based on the reference demonstration.

Once provided a labeled demonstration, we can temporally segment and collapse contiguous state-action pairs with the same label into a sequence of trajectories, producing a *segmented demonstration* $[\langle l_1, o_1, f_1, \tau_1 \rangle, ..., \langle l_h, o_h, f_h, \tau_h \rangle]$, where each segment represents a *skill*. Still, these demonstrations are currently specialized toward specific 1) object geometries and 2) initial poses.

To generalize them, we transform them using NDFs by extracting the pose of the target frame f relative to the observed object o. Specifically, for each demonstration data point $\langle o, f, S, a \rangle$ and current point cloud P_o for object o, we compute the NDF feature $z \leftarrow F(S_w^f \mid P_o)$ of the pose of frame f relative to observed object o. When applied to a state-action trajectory τ, this results in a *feature trajectory* $\mathcal{Z} = [z_1, ..., z_k]$. And when applied to a segmented demonstration, this produces a *feature demonstrations* $d = [\langle l_1, o_1, f_1, \mathcal{Z}_1 \rangle, ..., \langle l_h, o_h, f_h, \mathcal{Z}_h \rangle]$. Thus, we accumulate a *skill dataset* of feature demonstrations $\mathcal{D} = \{d_1, ..., d_n\}$, which can includes demonstrations that span multiple tasks.

3.4 Peg-in-Hole Example

Continuing the "Peg-in-Hole" running example in figure 2, we discuss relevant skills and plans. It requires two types of interactions: 1) moving the end-effector to grasp an object and 2) inserting a grasped object into another entity. These interactions can be formalized by the following planning operators, where $l \in \{\text{GRASP}, \text{ATTACH}\}$:

1. $\text{GRASP}(o, e; \mathcal{Z})$: move to grasp object o with end-effector frame e using feature trajectory $\mathcal{Z}$.
2. $\text{INSERT}(o, f; \mathcal{Z})$: while grasping object o, move to insert o relative to frame f using feature trajectory $\mathcal{Z}$.

A plan that directly adapts the demonstrations has the following form:

$$\pi = [\text{GRASP}(\text{peg}, \text{ee}; \mathcal{Z}_1), \text{INSERT}(\text{peg}, \text{hole}; \mathcal{Z}_2)].$$

In between the contact-involved component of these interactions are contact-adverse robot movement. We can plan motions between the end of the last component and the start of the next one in order to more robustly and efficiently move between segments. These segments can also be represented by planning operators [47]:

1. TRANSIT(τ): while not grasping any object, move the robot along trajectory τ.
2. TRANSFER(o; τ): while grasping object o, move the robot along trajectory τ.

A plan that includes motion operators has the following form:

$$\pi = [\text{TRANSIT}(\tau_1), \text{GRASP}(\text{peg}, \text{ee}; \mathcal{Z}_1), \\ \text{TRANSFER}(\text{peg}; \tau_2), \text{ATTACH}(\text{peg}, \text{hole}; \mathcal{Z}_2)]$$

4 NOD-TAMP

We present a set of algorithms for adapting a dataset of demonstrations to new problems. First, we show how a single demonstration can be adapted to a new environment using NDFs (Section 4.1). Then, we propose a planning algorithm that's able to combine skill segments from multiple demonstrations to maximize effectiveness (Section 4.2). Finally, we use motion planning to connect each segment in order to efficiently and robustly generalize to new workspaces (Section 4.3).

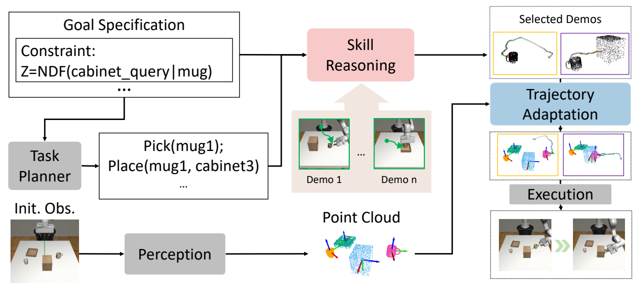

Figure 3: **System.** Overview of the NOD-TAMP system.

4.1 Trajectory Adaptation

The first algorithm we introduce directly adapts a demonstration d to the current task. Algorithm 1 displays the trajectory adaptation pseudocode. It iterates over each labeled feature trajectory in demonstration d and then over each timestep in the trajectories. For each timestep, it computes the target transformation T_z that corresponds to NDF feature z. Depending on whether object o is attached to another frame f' in the scene graph G, it composes the transform into a target world pose T_w^f for manipulation frame f. It then converts this into a target end-effector pose T_w^f, which will serve as a setpoint for a downstream controller or planner. After iterating over a labeled trajectory, the scene graph G is updated with the new state of the manipulated object o. This also models that the point cloud P_o for object object o is now attached to and thus moving with frame f.

4.2 Skill Planning

It can be limiting to be only able to reuse whole demonstrations. First, it is inflexible in problems where the, *plan skeleton*, or the sequence of skills changes. Second, segments of multiple demonstrations in conjunction might better address a new problem. For example, we may just transfer the picking part of the pick-and-place trajectory that work with a mug to interact with a new object, e.g., a bowl.

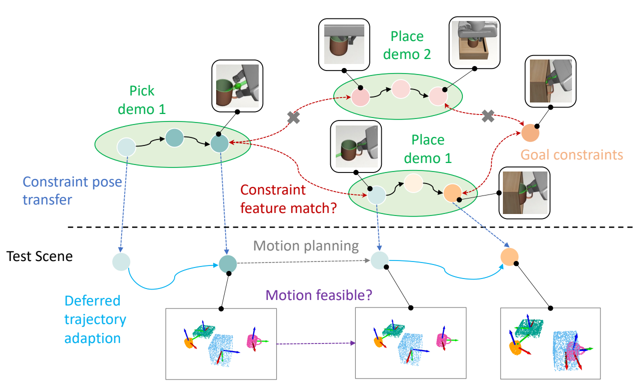

Figure 4: **Skill Planning and Trajectory Generation.** We illustrate how skill planning and trajectory adaptation collaborated to generate mug insertion trajectories in this example.

Algorithm 2 displays the pseudocode for the NOD-TAMP planner. It takes in a plan skeleton $\hat{\pi}$ that defines a sequence of skill types to consider. It first compiles a list of skills in dataset

$\mathcal{D}$ relevant to π. Then, it iterates through all possible plans using these skills. Each candidate π is scored based on the compatibility of subsequent actions using NDF features, the score c is computed as $c = \sum_{i=1}^{|\pi|-1} ||z_{i+1} - z_i||$, where for specific i, the NDF feature z_i and z_{i+1} are extracted as:

$$z_i, z_{i+1} \leftarrow F(T_i \mid P), F(T_{i+1} \mid P)$$

Here T_i and T_{i+1} are the query pose determined at the last time step and first time step of the trajectories from skill i and skill $i+1$ respectively, and P is the point cloud of the target object for skill i and skill $i+1$ captured during demonstration.

After we obtain all scores for all skill combinations, the plan with the lowest plan-wide NDF feature distance is returned. For simplicity, we present this as a Cartesian product over relevant skills, but this can be done more efficiently by performing a Uniform Cost Search in plan space, where the NDF feature distance serves as the cost function. The visualization of the feature matching process for a mug placing example is presented in Fig. 5, showing that the grasping on the mug rim would be compatible with placing the mug uprightly in a bin, and grasp on the handle would be compatible with inserting the mug into a cabinet.

4.3 Transit & Transfer Motion

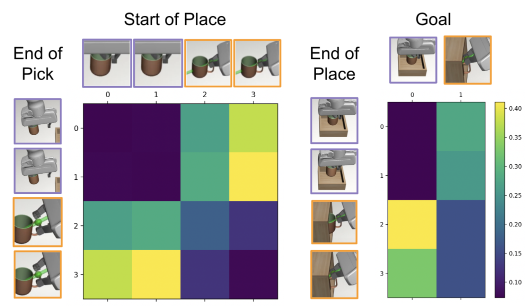

Figure 5: **Feature Matching.** We show the feature matching distance for different skill combinations when placing the mug.

Adapting demonstrated skills is particularly effective at generating behavior that involves contact. However, demonstrations typically contain long segments without contact (outside of holding an object). Because these components do not modify the world, it is often not productive to replicate them. Thus, we temporally trim skill demonstrations to focus on the data points that involve contact. In our implementation, we simply select the 50 steps that are most close to the time point when the contact happening.

After trimming, and in many cases before trimming, two adjacent skills might be quite far away in task space. While we could simply interpolate between them, this is not generally safe because the straight-line path may cause the robot to unexpectedly collide. To address this, we use motion planning to optimize for safe and efficient motion that reaches the start of next the skill. Motion planning generally requires some characterization of the collision volume of the obstacles to avoid. Because we do not assume access to object models, we use the segmented point clouds $\mathcal{P}$ the collision representation, where each point in the cloud is a sphere with radius $r > 0$.

Algorithm 3 displays the pseudocode for the full NOD-TAMP policy with motion planning. It displayed in an manner with online motion planning and execution, but full plans can also be computed offline. The policy first computes a skill plan π and then adapts each labeled trajectory in sequence. Between trajectories, it uses PLAN-MOTION to plan a trajectory from the current robot configuration q to a configuration that reaches the first end-effector pose T_w^e. Specifically, it samples goal configurations q' using inverse kinematics and then invokes a sampling-based motion planner between q and q'. Once the end-effector arrives at T_w^e, for each pose yielded by the skill, the policy deploys OSC to track the target poses.

A detailed example of how the system work for placing the mug into the cabinet is presented in Fig. 4. The skill planning process first identifies the useful trajectories of the pick skill and place skill through matching the constraint feature. A motion planner then generates the transition motion between skills. Once the motion transition among every connections between the consecutive skills are determined, we transfer the trajectories to the test scene through trajectory adaptation.

5 Experiments

We aim to validate NOD-TAMP and how its components contribute to solve long-horizon tasks, perform contact-rich manipulation, and generalize to new object shapes.

5.1 Experimental Setup

Tasks.

We introduce four simulated [48] table organization tasks (Fig. 6 bottom), which vary in difficulty and generalization challenges. We use 10 mug models of varying shapes and dimensions from the ShapeNet dataset [49]. Demonstrations are provided for one mug and the testing environment randomly samples among the remaining 9 mugs. We provide all methods with only **two** picking and placing skill trajectories on **one** mug instance (Fig. 6 top). The two trajectories vary by grasping pose (side vs. top) and placing pose (side vs. top).

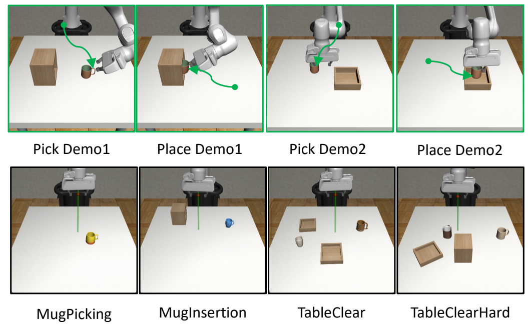

Figure 6: **Demonstration Skills and Tasks.** Illustration of the demonstration skills and the task setups.

MugPicking - Pick up different mugs with varying shapes and initial poses. The task tests the ability to adapt the trajectory based on object shapes and poses.

MugInsertion - Insert mugs of varying shape into a tight cabinet. Both the mug and the cabinet are randomly placed on the table. This task requires fine-grained manipulation for the insertion and adaptation to different initial configurations.

TableClear - This long-horizon task requires the robot to place two mugs into two bins, which aims to test the ability to achieve long-term goals by reusing the skills.

TableClearHard - This long-horizon task requires the robot to stow one mug into a cabinet with side opening and place another mug into a bin. The robot must select a feasible chain of skills (side-picking to side-stowing) to transport each mug.

Baselines. We compare NOD-TAMP with ablation baselines and adapt state-of-the-art methods to facilitate fair comparison. All NDF-based methods share the same NDF model pretrained on mug shapes provided by the original implementation [8].

NDF$^+$ [8]: We augment the original NDF manipulation planner, which only generates key-frame manipulation poses, with our task planner and the skill reasoning module, which provides the baseline with the target object and the query pose at different stages. This baseline also uses a motion planner to transition between key-frame poses.

MimicGen$^+$ [15]: MimicGen directly pieces together contact-rich segments from human demonstrations and linearly interpolates the intermediate steps. The robot control poses in the contact-rich segments are transformed to the relevant object frame and then sent to the controller without further adaptation. To ensure fair comparison, we augment MimicGen with a motion planner for collision avoidance.

Ours w/o Skill Reasoning (Ours-SR): This ablation removes the skill reasoning module. For each skill, we randomly choose a reference trajectory from the collected demonstrations belonging to this skill, and we bridge the intermediate transition with the motion planner. This baseline validates the importance of skill reasoning for generalizing across tasks.

Ours w/o Motion Planning (Ours-MP): This ablation removes the motion planning component and uses linear trajectory interpolation to transition between the adapted skill trajectories generated by the optimization. We set up this baseline to validate the benefit of leveraging motion planning, a capability present in TAMP systems.

Naive Skill Chaining (NSC): This baseline ablates both the skill reasoning and the motion planning component: it randomly selects a reference trajectory for each skill, adapts the skill with NDF, and uses linear trajectory interpolation to transition between the selected trajectories.

5.2 Main Results

Table 1: Evaluation results (success rate) of all methods.

Tasks	NDF$^+$	MimicGen$^+$	NSC	Ours (-MP)	Ours (-SR)	Ours
MugPicking	80	70	**85**	80	**85**	**85**
MugInsertion	75	55	80	85	80	**90**
TableClear	60	75	80	75	**85**	**85**
TableClearHard	40	55	15	50	10	**80**

We observe that NOD-TAMP consistently achieves a high success rate (80-90%) across all tasks and outperforms the other baselines and ablations (see Table 1, some qualitative results are visualized in Fig. 8). Below, we highlight specific comparisons and takeaways that underscore the importance of the trajectory adaptation, skill planning, and motion planning components of NOD-TAMP.

NOD-TAMP exhibits strong performance across long-horizon tasks and is able to reuse skills in new contexts. The TableClear task requires methods to re-use the existing **two** pick-and-place human demonstrations, which only consisted of single mug and bin interactions, to stow two mugs into two bins. NOD-TAMP achieves strong performance (85%) and outperforms MimicGen$^+$ by 15% and NDF$^+$ by 25% on this task, showcasing a superior ability to re-purposing short-horizon skill demonstrations for long-horizon manipulation.

NOD-TAMP exhibits strong generalization capability across goals, objects, and scenes in long-horizon tasks. The TableClearHard task requires intelligent selection and application of different demonstration trajectories to achieve different kinds of mug placements. The task also requires dealing with novel mug objects, and novel scene variations. Here, NOD-TAMP achieves 80%, outperforming all other baselines by a wide margin (40% better than NDF$^+$, 25% better than MimicGen$^+$). We also see the clear benefit of the skill reasoning component to achieve the different goals in this task – NOD-TAMP outperforms (Ours-SR) by 70% and NSC by 65%. The omission of the skill reasoning module results in an incompatible composition of skills. For example, the robot may grip the rim of a mug and attempt to insert it into the cabinet, leading to collisions between the cabinet and the gripper. Finally, the motion planning component is also valuable in dealing with the scene variation – NOD-TAMP outperforms (Ours-MP) by 30%. Simply connecting end-effector trajectories through linear interpolation without considering obstacles often leads to collisions between the robot or the held object and its surroundings. In particular, we frequently observed such failures for the Ours (-MP) approach that the gripper became obstructed by the cabinet after completing the insertion of the first mug.

6 Conclusion

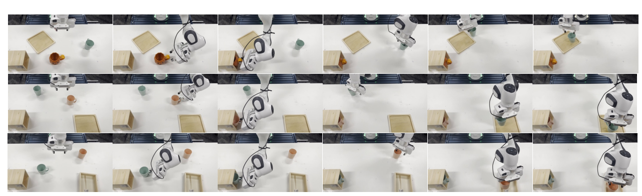

Figure 7: **Real Robot Executions.** Key frames of three task execution processes.

We introduced NOD-TAMP, a framework for adaptable manipulation planning using human demonstrations for long-horizon tasks. While powerful, NOD-TAMP has limitations that inspire future work. First, because of the expensive NDF-based pose optimization process, NOD-TAMP is slow in practice and is far from real-time. We plan to experiment with more efficient NOD variants and faster optimization procedure. Moreover, an important aspect of TAMP problem is representing and satisfying constraints. We plan to explore NOD-based constraint representations and incorporate constraints into the planning objectives. Finally, NOD-TAMP solves the high-level task plans and low-level trajectories in silos. As a next step, we aim to enable bi-level communication in planning and extend NOD-TAMP into a full-fledged integrated TAMP system [17].

References

[1] B. Aceituno-Cabezas and A. Rodriguez, "A global quasi-dynamic model for contact-trajectory optimization in manipulation," 2020.

[2] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *ICRA*, 2011.

[3] C. R. Garrett *et al.*, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[4] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez, "Learning symbolic operators for task and motion planning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 3182–3189.

[5] S. Cheng and D. Xu, "League: Guided skill learning and abstraction for long-horizon manipulation," *IEEE Robotics and Automation Letters*, 2023.

[6] A. Brohan *et al.*, "Rt-1: Robotics transformer for real-world control at scale," in *arXiv preprint arXiv:2212.06817*, 2022.

[7] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[8] A. Simeonov *et al.*, "Neural descriptor fields: Se(3)-equivariant object representations for manipulation," 2022.

[9] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Conference on Robot Learning*, PMLR, 2022, pp. 894–906.

[10] B. Wen, W. Lian, K. Bekris, and S. Schaal, "You only demonstrate once: Category-level manipulation from single visual demonstration," *arXiv preprint arXiv:2201.12716*, 2022.

[11] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation," *arXiv preprint arXiv:1806.08756*, 2018.

[12] P. Sundaresan *et al.*, "Learning rope manipulation policies using dense object descriptors trained on synthetic depth data," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 9411–9418.

[13] A. Simeonov *et al.*, "Se (3)-equivariant relational rearrangement with neural descriptor fields," in *Conference on Robot Learning*, PMLR, 2023, pp. 835–846.

[14] E. Chun, Y. Du, A. Simeonov, T. Lozano-Perez, and L. Kaelbling, "Local neural descriptor fields: Locally conditioned object representations for manipulation," *arXiv preprint arXiv:2302.03573*, 2023.

[15] A. Mandlekar *et al.*, "Mimicgen: A data generation system for scalable robot learning using human demonstrations," in *Conference on Robot Learning (CoRL)*, 2023.

[16] L. P. Kaelbling and T. Lozano-Pérez, "Pre-image backchaining in belief space for mobile manipulation," in *Robotics Research*, Springer, 2017, pp. 383–400.

[17] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 440–448.

[18] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," 2018.

[19] 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*, vol. 61, pp. 215–289, 2018.

[20] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Learning compositional models of robot skills for task and motion planning," *The International Journal of Robotics Research*, vol. 40, no. 6-7, pp. 866–894, 2021.

[21] J. Liang, M. Sharma, A. LaGrassa, S. Vats, S. Saxena, and O. Kroemer, "Search-based task planning with learned skill effect models for lifelong robotic manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 6351–6357.

[22] A. Simeonov *et al.*, "A long horizon planning framework for manipulating rigid pointcloud objects," in *Conference on Robot Learning*, PMLR, 2021, pp. 1582–1601.

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

[24] R. Chitnis *et al.*, "Guided search for task and motion plans using learned heuristics," in *ICRA*, IEEE, 2016.

[25] B. Kim, L. Shimanuki, L. P. Kaelbling, and T. Lozano-Pérez, "Representation, learning, and planning algorithms for geometric task and motion planning," *IJRR*, vol. 41, no. 2, 2022.

[26] M. J. McDonald and D. Hadfield-Menell, "Guided imitation of task and motion planning," in *Conference on Robot Learning*, PMLR, 2022, pp. 630–640.

[27] J. Gu *et al.*, "Maniskill2: A unified benchmark for generalizable manipulation skills," *arXiv preprint arXiv:2302.04659*, 2023.

[28] M. Dalal, A. Mandlekar, C. Garrett, A. Handa, R. Salakhutdinov, and D. Fox, "Imitating task and motion planning with visuomotor transformers," *arXiv preprint arXiv:2305.16309*, 2023.

[29] A. Curtis, X. Fang, L. P. Kaelbling, T. Lozano-Pérez, and C. R. Garrett, "Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 1940–1946.

[30] C. Wang, D. Xu, and L. Fei-Fei, "Generalizable task planning through representation pretraining," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8299–8306, 2022.

[31] S. Cheng and D. Xu, "Guided skill learning and abstraction for long-horizon manipulation," *arXiv preprint arXiv:2210.12631*, 2022.

[32] A. Mandlekar, C. R. Garrett, D. Xu, and D. Fox, "Human-in-the-loop task and motion planning for imitation learning," in *7th Annual Conference on Robot Learning*, 2023.

[33] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," *arXiv preprint arXiv:1710.04615*, 2017.

[34] A. Mandlekar *et al.*, "What matters in learning from offline human demonstrations for robot manipulation," in *Conference on Robot Learning (CoRL)*, 2021.

[35] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, "Learning to generalize across long-horizon tasks from human demonstrations," *arXiv preprint arXiv:2003.06085*, 2020.

[36] E. Jang *et al.*, "Bc-z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning*, PMLR, 2022, pp. 991–1002.

[37] M. Ahn *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[38] C. Lynch *et al.*, "Learning latent plans from play," in *Conference on robot learning*, PMLR, 2020, pp. 1113–1132.

[39] N. Di Palo and E. Johns, "Learning multi-stage tasks with one demonstration via self-replay," in *Conference on Robot Learning*, PMLR, 2022, pp. 1180–1189.

[40] E. Johns, "Coarse-to-Fine Imitation Learning: Robot Manipulation from a Single Demonstration," *ICRA*, 2021.

[41] V. Vosylius and E. Johns, "Where to start? transferring simple skills to complex environments," *arXiv preprint arXiv:2212.06111*, 2022.

[42] A. Chenu, O. Serris, O. Sigaud, and N. Perrin-Gilbert, "Leveraging sequentiality in reinforcement learning from a single demonstration," *arXiv preprint arXiv:2211.04786*, 2022.

[43] E. Valassakis, G. Papagiannis, N. Di Palo, and E. Johns, "Demonstrate once, imitate immediately (dome): Learning visual servoing for one-shot imitation learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 8614–8621.

[44] J. Liang, B. Wen, K. Bekris, and A. Boularias, "Learning sensorimotor primitives of sequential manipulation tasks from visual demonstrations," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 8591–8597.

[45] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 13 438–13 444.

[46] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[47] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 729–746, 2004.

[48] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," Oct. 2012, pp. 5026–5033.

[49] A. X. Chang *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

7 Appendix

7.1 Algorithms

We provide the pseudo-code of our proposed algorithms.

Algorithm 1 Trajectory adaptation

Declare: Segmented point clouds $\mathcal{P}$
Declare: Initial end-effector pose S_w^e
Declare: Demonstration dataset $\mathcal{D} = \{d_1, ..., d_n\}$
Declare: NDFs $\mathcal{F}$

procedure ADAPT-TRAJ$(\mathcal{P}, S_w^e, d; \mathcal{F})$
  $\mathcal{S} = \{f : \text{NDF-ESTIMATE}(\mathcal{P}, f) \mid f \in F\} \cup \{e : S_w^e\}$
  $G \leftarrow \{\ \}$ ▷ Object scene graph
  for $\langle l, o, f, \mathcal{Z} \rangle \in d$ **do**
    for $z \in \mathcal{Z}$ **do**
      $T_z \leftarrow \text{NDF-OPTIMIZE}(\mathcal{F}[o], \mathcal{P}[o], z)$
      $S_w^e, S_w^f \leftarrow \mathcal{S}[e], \mathcal{S}[f]$ ▷ End-effector & frame
      if $o \in G$ **then** ▷ Relative to scene graph
        $\langle f', T_w^{f'} \rangle \leftarrow G[o]$
        $S_w^{f'} \leftarrow S[f']$
        $T_w^f \leftarrow S_w^{f'} \cdot (T_w^{f'})^{-1} \cdot T_z$
      else ▷ Relative to world frame
        $T_w^f \leftarrow T_z$
      $T_w^e \leftarrow S_w^e \cdot (S_w^f)^{-1} \cdot T_w^f$ ▷ End-effector target
      ▷ If $f = e$, this reduces to $T_w^e \leftarrow T_w^f$
      yield $\langle l, o, f, T_w^e \rangle$ ▷ Yield target to controller
      $S[e] \leftarrow T_w^e$
    $G[o] \leftarrow \langle f, S_w^f \rangle$ ▷ Set f as the parent of o

Algorithm 2 NOD-TAMP planner

Declare: Plan skeleton $\widehat{\pi} = [\langle l_1, o_1, f_1 \rangle, ..., \langle l_h, o_h, o_h \rangle]$

procedure PLAN-NDF-SKILLS$(\mathcal{P}, \widehat{\pi}; \mathcal{D}, \mathcal{F})$
  $D = [\ ]$ ▷ List of demos per skill
  for $\langle l_i, o_i, f_i \rangle \in \widehat{\pi}$ **do**
    $D \leftarrow D + [\{\langle l, o, f, \mathcal{Z} \rangle \mid d \in \mathcal{D}, \langle l, o, f, \mathcal{Z} \rangle \in d.$
  $l{=}l_i \wedge o{=}o_i \wedge f{=}f_i\}]$
  $\pi_* \leftarrow$ **None**; $c_* \leftarrow \infty$
  for $\pi \in \text{PRODUCT}(D_\pi)$ **do** ▷ All combinations
    $c_\pi \leftarrow 0$ ▷ Feature cost
    for $i \in [1, ..., |\pi|{-}1]$ **do**
      $\langle l_i, o_i, f_i, \mathcal{Z}_i \rangle \leftarrow \pi[i]$
      $\langle l_{i+1}, o_{i+1}, f_{i+1}, \mathcal{Z}_{i+1} \rangle \leftarrow \pi[i{+}1]$
      if $o_i = o_{i+1}$ **then** ▷ Actions with same object
        $F \leftarrow \mathcal{F}[o]$; $P \leftarrow \mathcal{P}[o]$
        $T_i \leftarrow \text{NDF-OPTIMIZE}(F, P, \mathcal{Z}_i[-1])$
        $T_{i+1} \leftarrow \text{NDF-OPTIMIZE}(F, P, \mathcal{Z}_{i+1}[0])$
        $z_i, z_{i+1} \leftarrow F(T_i \mid P), F(T_{i+1} \mid P)$
        $c_\pi \leftarrow c_\pi + ||z_{i+1} - z_i||$
    if $c_\pi < c_*$ **then** ▷ Update best plan
      $\pi_* \leftarrow \pi$; $c_* \leftarrow c_\pi$
  return π_*

Algorithm 3 NOD-TAMP policy

procedure NOD-TAMP-POLICY($\mathcal{P}, \pi; \mathcal{D}, \mathcal{F}$)
    $\pi \leftarrow$ PLAN-NDF-SKILLS($\mathcal{P}, \pi; \mathcal{D}, \mathcal{F}$)
    if $\pi =$ **None then**
        return False ▷ Skill planning failed
    $a \leftarrow$ **None** ▷ Current action
    $S^e_w \leftarrow$ FORWARD-KIN(q)
    for $\langle o, f, T^e_w \rangle \in$ ADAPT-TRAJ($\mathcal{P}, S^e_w, \pi; \mathcal{F}$) **do**
        if $\langle o, f \rangle \neq a$ **then** ▷ Action changed
            $a \leftarrow \langle o, f \rangle$ ▷ Update current action
            $q \leftarrow$ OBSERVE-CONF()
            $\tau \leftarrow$ PLAN-MOTION($\mathcal{P}, q, T^e_w$)
            if $\tau =$ **None then**
                return False ▷ Motion planning failed
            EXECUTE-TRAJ(τ)
        $q \leftarrow$ OBSERVE-CONF()
        EXECUTE-OSC(q, T^e_w) ▷ Operational Space
    return True ▷ Policy succeeded

7.2 More Qualitative Results

With **TWO** picking and placing trajectories on just **ONE** mug, we evaluate our method's effectiveness across diverse tasks with different mug shapes, poses, and goal setups. NOD-TAMP consistently achieves a high success rate (80-90%) across all tasks, some executions are visualized in Fig. 8.

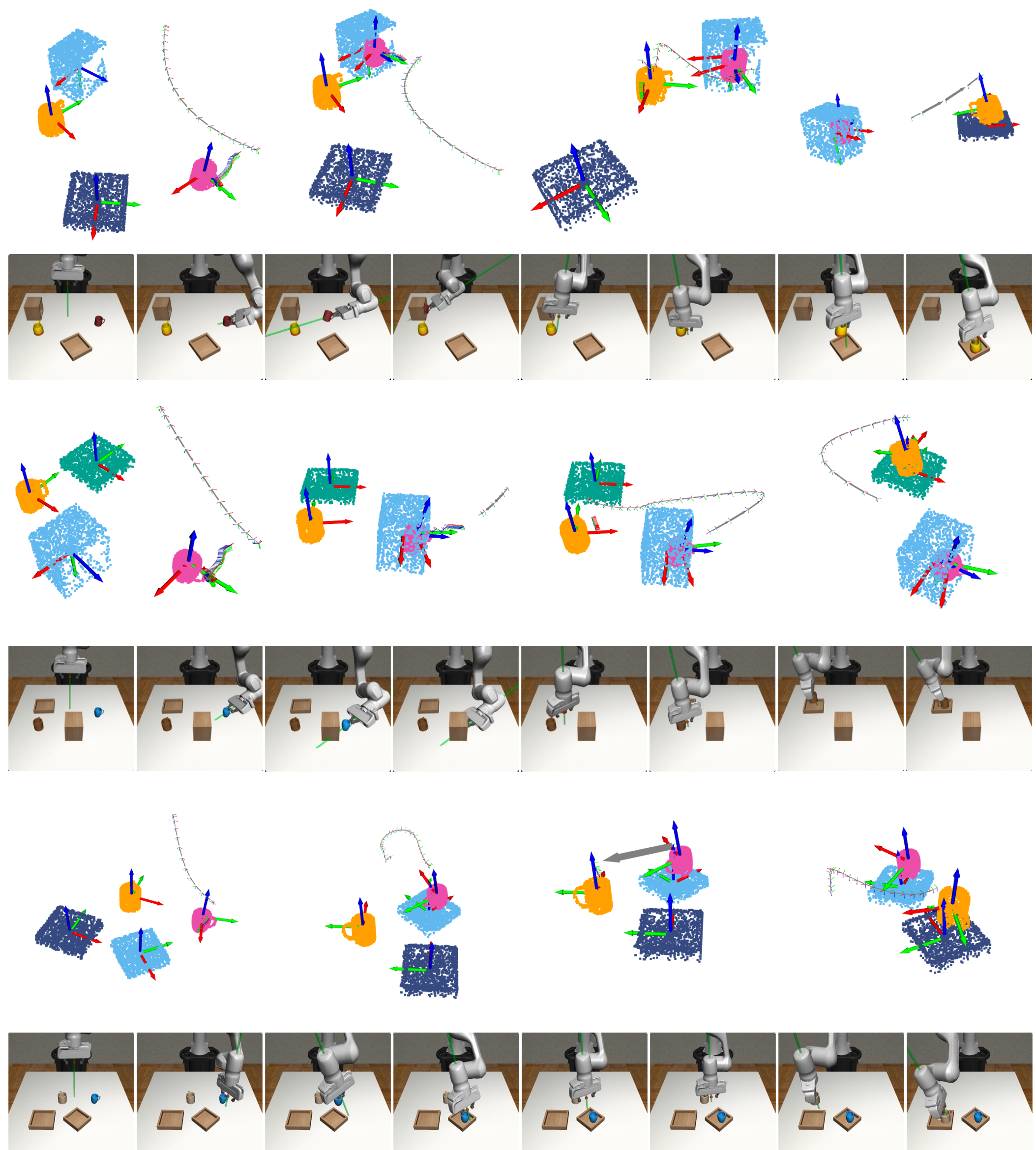

Figure 8: **Qualitative Visualization.** Trajectories generated by our system at different stages, the planned scene represented as point cloud, and snapshots of the execution process.