Imitating Task and Motion Planning with Visuomotor Transformers

Anonymous Author(s)
Affiliation
Address
email

Abstract: Imitation learning is a powerful tool for training robot manipulation policies, allowing them to learn from expert demonstrations without manual programming or trial-and-error. However, common methods of data collection, such as human supervision, scale poorly, as they are time-consuming and labor-intensive. In contrast, Task and Motion Planning (TAMP) can *autonomously* generate *large-scale* datasets of *diverse* demonstrations. In this work, we show that the combination of large-scale datasets generated by TAMP supervisors and flexible Transformer models to fit them is a powerful paradigm for robot manipulation. We present a novel imitation learning system called OPTIMUS that trains large-scale visuomotor Transformer policies by imitating a TAMP agent. We conduct a thorough study of the design decisions required to imitate TAMP and demonstrate that OPTIMUS can solve a wide variety of challenging vision-based manipulation tasks with over 70 different objects, ranging from long-horizon pick-and-place tasks, to shelf and articulated object manipulation, achieving 70 to 80% success rates. Video results at optimustransformer.github.io/.

Keywords: Imitation Learning, Task and Motion Planning, Transformers

1 Introduction

Large-scale data-driven learning, powered by the Transformer architecture [1], has transformed the fields of natural language processing (NLP) and computer vision (CV). Large models at the scale of billions of parameters, trained on massive corpi [2, 3, 4] exhibit powerful capabilities such as writing coherently [2, 5], answering questions [6], and image classification [7, 8] and generation [9]. Although there is recent work applying large Transformers to robot learning [10, 11, 12], the recipe of large-scale data-driven learning and Transformers has not yet achieved the same level of widespread success in robotic manipulation. One significant bottleneck is a lack of useful data – data collection is especially challenging because it requires the robot to interact in **real-time** with the world. Furthermore, not all data is useful: the collected interactions should be **relevant** for solving manipulation tasks of interest. Finally, for learned policies to be broadly applicable, they require access to a **diverse** set of task instances, which necessitates a **scalable** data collection pipeline.

Prior work has used human teleoperation [13, 14, 15, 16, 17, 18, 19] to collect large robot manipulation datasets, enabling training large scale models [10, 20]. However, this can require significant human time and labor – RT-1 [10] required 1.5 years of data collection. Other works have used reinforcement learning (RL) – this has the potential to scale more efficiently via autonomous data collection, but it is prohibitively expensive to run in terms of robot time due to its sample inefficiency [21, 22, 23, 24], and requires significant computation time and human reward engineering [25, 26]. In this work, we consider an alternative form of supervision, Task and Motion Planning (TAMP) [27], which addresses some key limitations of prior data-collection techniques. TAMP plans a discrete sequence of objects to interact with and how to manipulate them, and continuous motions that safely and correctly facilitate these interactions. TAMP supervision is beneficial because it: 1) collects data *autonomously* and 2) *efficiently* generates demonstrations by leveraging privileged information. TAMP can generate supervision on a wide distribution of task instances, producing **task relevant**, **diverse**, **large-scale** datasets for robot-learning.

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

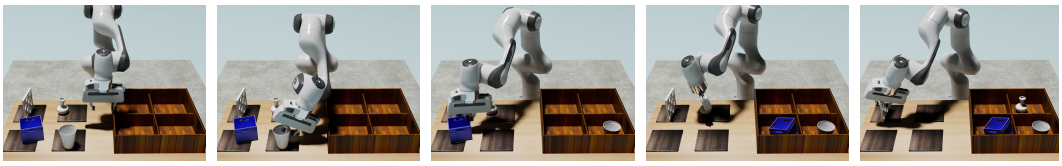


Figure 1: **Long-horizon task visualization.** We visualize the initial state and each intermediate pick state for the pick-and-place task. Note there is significant variation in geometry across each object, requiring the agent to perform a diverse series of grasps to complete the task.

However, TAMP on its own requires accurate estimation of the scene geometry and its state, is not reactive, and can spend significant time on planning. Instead, we propose to imitate TAMP across a wide range of tasks using closed-loop, visuomotor Transformer policies. As a result, we obtain **fast-to-execute**, **reactive** agents that can solve long horizon manipulation tasks **without state estimation**. Furthermore, by training on large, diverse datasets of successful trajectories, we show in our experimental evaluation that large Transformer policies have the capability of improving beyond TAMP performance. Finally, we note that while McDonald et al. [28] have also learned closed-loop policies from TAMP supervision, we perform an extensive study of the challenges in imitating TAMP, evaluate models across a wide range of tasks, and demonstrate novel capabilities including high-frequency end-to-end visuomotor control, task plan adaptation and scene generalization. Some challenges in imitating TAMP include learning from decisions made based on privileged information and multimodal demonstrations [15, 29].

To address these challenges, we propose **O**ffline **P**retrained **T**AMP **Im**itation **S**ystem, or OPTIMUS, a system for training visuomotor Transformer policies via imitation learning. **Our contributions are:**
- a novel framework for training visuomotor Transformer policies for high-frequency low-level control by taking advantage of TAMP supervision
- an empirically validated data-generation pipeline and study of the insights required to imitate TAMP
- strong results demonstrating that our trained policies can solve **over 300** long-horizon manipulation tasks involving up to **8 stages** and **72** different objects, achieving success rates of **over 70%**

2 Preliminaries

Related Work: OPTIMUS builds on a rich history of work in imitation learning and TAMP for robotic manipulation. In this work, we focus on the setting of offline learning via behavior cloning [30], in which a plethora of work has leveraged human demonstrations to learn effective policies [10, 20, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39]. Our work instead relies on a TAMP supervisor, which can generate large, diverse datasets without human supervision. Furthermore, we build on recent work using Transformers for imitation [10, 12, 40, 41, 42, 43] by designing a fast to execute, visuomotor architecture operating over low-level control inputs. Finally, our system uses Task and Motion Planning [27, 44] to generate imitation data, a paradigm that has been recently explored in approaches that imitate planning [45, 46, 47] as well as TAMP directly [28]. In contrast to such prior work, our system adapts the TAMP data-generation process for improved imitation learning and uses a Transformer architecture that does not require any scene or task specific knowledge. **See Appendix E for full related work.**

Background: We address Partially Observable Markov Decision Processes (POMDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, p_0, \Omega, \mathcal{O}, \gamma \rangle$, where $\mathcal{S}$ is the set of environment states, $\mathcal{A}$ is the set of actions, $\mathcal{T}(s' \mid s, a)$ is the transition probability distribution, $\mathcal{R}(s, a)$ is the reward function, p_0 defines the distribution of the initial state $s_0 \sim p_0$, Ω is the set of observations, $\mathcal{O}(o \mid s)$ is the observation distribution, and γ is the discount factor. We consider sparse reward POMDPs where $\mathcal{R}(s, a) \equiv -\mathbb{1}_{s \notin \mathcal{S}_*}$ is zero at terminal, goal states $\mathcal{S}_* \subseteq \mathcal{S}$ and elsewhere negative one. Solutions are *policies* $\pi_\theta(o_t, h_t)$ that operate on the history $h_t = (o_1, a_1, ..., o_{t-1}, a_{t-1})$ of observations $o \in \Omega$ and actions $a \in \mathcal{A}$, outputting the next action $a_t = \pi_\theta(o_t, h_t)$. The objective is to find a policy $\pi_\theta(o_t, h_t)$ that maximizes the expected policy return $\mathbb{E}[\sum_{t=1}^{\infty} \mathcal{R}(s_t, a_t)]$. In this context, for behavior cloning, $\pi_\theta(o, h_t)$ is trained to regress a_t from (o_t, h_t) from a dataset $\mathcal{D}$ consisting of trajectories $\tau_i^n = (o_1^i, a_1^i, ..., o_{T_i}^i, a_{T_i}^i)$ produced by the expert, in which i is the i-th trajectory in the dataset, T_i is its length and n is the n-th MDP.

Task and Motion Planning: TAMP algorithms address deterministic and observable, but hybrid, control problems [27]. In order to apply them to the POMDP for data collection, we grant them

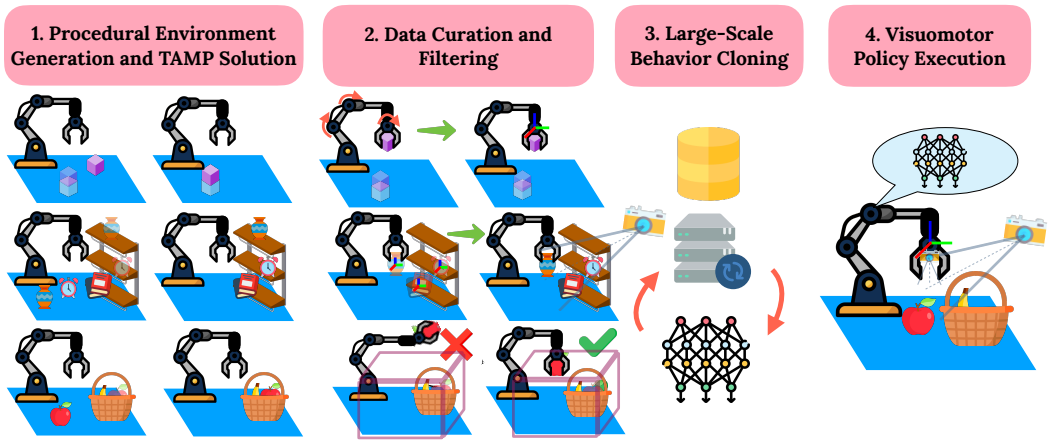


Figure 2: **OPTIMUS system**. *Column 1*: We generate a variety of tasks with differing initial configurations (*left*) and goals (*right*). *Column 2*: We transform TAMP joint space demonstrations to task space (*top*), go from privileged scene knowledge in TAMP to visual observations (*middle*) and prune TAMP demonstrations based on workspace constraints. *Columns 3 and 4*: We perform large-scale behavior cloning using a Transformer-based architecture and execute the visuomotor policies.

observability to the system state s. In simulation, this can be done through providing them access to the underlying simulator state. As a result, a TAMP policy $\pi_p(s_t)$ need only be a function of the state s_t, which is a sufficient statistic for the history $\langle h_t, o_t \rangle$. To construct this policy, we approximate the now observable POMDP with a deterministic model that can be effectively planned with [48]. Then, a TAMP algorithm uses this approximate model to plan a sequence of object interactions, the constraints present in each interaction (*e.g.* grasps and placements), and finally safe joint motions that realize them. An automated policy is built around the TAMP algorithm by tracking plans with a high-frequency feedback controller that outputs actions a and periodically replanning [48].

Consider an example TAMP problem in which the goal is to place a **cup** on a **shelf** (*i.e.* the **Shelf** task). The TAMP model has the following parameterized actions: $\texttt{move}(q_1, \tau, q_2)$ moves the robot from configuration q_1 to configuration q_2 via trajectory τ, $\texttt{pick}(o, g, p, q)$ picks object o at placement pose p with grasp pose g when the robot is at configuration q, and $\texttt{place}(o, g, p, q, o_2)$ places object o at placement pose p on object o_2 with grasp pose g when the robot is at configuration q. An example TAMP plan p for the Shelf task is: $p = [\texttt{move}(\boldsymbol{q_0}, \tau_1, q_1), \texttt{pick}(\textbf{cup}, g, \boldsymbol{p_0}, q_1), \texttt{move}(q_1, \tau_2, q_2), \texttt{place}(\textbf{cup}, g, p, q_2, \textbf{shelf})]$ The values in bold, the initial configuration $\boldsymbol{q_0}$ and cup placement $\boldsymbol{p_0}$, are constants. The other values are free parameters. A TAMP algorithm searches to find both the *plan skeleton*, the sequence of parameterized actions, as well as values for grasp g, placement p, configurations q_1, q_2, and trajectories τ_1, τ_2 that satisfy grasp, stability, kinematic, and collision constraints.

3 Designing a TAMP Imitation System

In this section, we motivate and describe our TAMP imitation system, OPTIMUS. We distill a privileged TAMP policy into a neural network in order to obtain policies that do not require access to state information, are fast to execute, and react instantaneously. To design OPTIMUS, we apply a TAMP supervisor to a procedural problem generator to produce demonstrations across a diverse range of tasks. However, trajectories produced by TAMP are not necessarily straightforward for an agent to imitate, especially when the agent must learn without access to privileged state information. Consequently, we carefully create a data curation pipeline and couple it with agent design decisions that maximize its ability to learn from TAMP trajectories and solve challenging manipulation tasks.

We consider tasks with significant variation across objects, poses, and configurations. We design four environments: 1) block stacking, 2) single and multi-step pick and place, 3) shelf pick and place, and 4) articulated object manipulation with microwaves. To obtain object diversity, we load objects from the ShapeNet dataset [49]. With a TAMP supervisor and diverse task distribution in place, we now describe the data collection pipeline and how we use it for policy learning.

3.1 Cost-Minimizing TAMP

We use the PDDLStream planning framework [50] to model the TAMP domain and the *adaptive* algorithm, a sampling-based algorithm, to plan. Our formulation makes use of samplers for grasp generation, placement sampling, inverse kinematics, and motion planning. The samplers can produce a large, if not infinitely large, set of diverse values. We implement the grasp generator using the ACRONYM grasp dataset [51] for ShapeNet objects. We use TRAC-IK [52] for inverse kinematics (IK), and bidirectional Rapidly-Exploring Random Trees (BiRRT) [53] for motion planning.

When using TAMP solutions for imitation learning, it is essential to train on high-quality plan traces. Behavior cloning techniques typically are adverse to multi-modal policy behavior, so a TAMP demonstrator that takes several different actions at a particular state produces data is challenging to imitate. One way to reduce TAMP policy variability is to optimize for low-cost plans. Although a TAMP problem is not guaranteed to have a unique minimum cost solution, this strategy biases solutions to a consistent family of low-cost plans.

We propose a two-stage approach to producing low-cost TAMP solutions. First, we use cost-sensitive PDDLStream planning that minimizes the joint-space distance traveled. Specifically, we define costs for $\texttt{move}(q_1, \tau, q_2)$ actions that limit ∞-norm (max) of the distance $||q_1 - q_2||_\infty$ between configurations q_1, q_2. The straight-line distance between two configurations is a lower bound on the length of the shortest collision-free path between them. We optimize this lower bound before performing motion planning which is computationally expensive due to continuous collision checking. This PDDLStream algorithm is asymptotically optimal [50, 54], but it might take arbitrary long to find a plan below a target cost bound. In practice, we run the planner in an anytime mode with a computation budget of five seconds and return the best plan identified. In the second stage, we perform motion planning using BiRRT; however, it can produce motions that are jagged and locally sub-optimal. To smooth these trajectories, we post-process them using cubic spline short cutting with velocity and acceleration limits [55], which converges to a locally time-optimal trajectory.

Finally, we aim to limit the variability in IK solutions. This also advantageous for task-space control, which lacks the control authority to reach all IK solutions. We seed TRAC-IK's optimization-based IK from a single configuration seed, the initial configuration, and optimize for the closest solution to the initial configuration within a 10 millisecond timeout. This also biases TAMP toward plans that stay near the initial configuration, typically accelerating the search for low-cost plans. By intentionally not exploiting the redundancy to explore diverse IK solutions, we limit the completeness of the TAMP algorithm for the benefit of downstream learning.

3.2 Generating Imitation Data from TAMP

Directly training on datasets collected by TAMP is a challenge for imitation learning, as the TAMP system operates with access to information unavailable to the learner, controls the robot in joint space, which can be difficult to learn in, and generates demonstrations that may not necessarily take the shortest path in task-space. To address these issues, we highlight design decisions regarding the observations and actions we produce from the TAMP data-generation process as well as how we select which demonstrations to train on.

Imitating a Privileged Expert: TAMP operates over a privileged view of the world. It has access to information that is difficult to obtain from a perception system, such as environment geometry and object state. To address these issues, OPTIMUS operates over image observations by using multiple camera views in each task (1-2 fixed cameras and 1 wrist-mounted camera). We find that multiple views, in particular the wrist camera, help the agent to better perceive scene geometry [56] and align its actions with the privileged expert. By training over multi-view RGB observations, we provide the network with an observation space that is invariant to object symmetry, encodes 3D information, is efficient to train over, and enables simplicity of the architecture.

Learning from TAMP Generated Actions: The TAMP system plans arm motions in configuration space, in which it can fully control each robot degree of freedom. However, training vision-based policies in joint-space is difficult due to the challenge of learning the camera projection from pixels to poses and then the redundant inverse kinematics mapping from pixels to joint angles [29, 57, 58]. Additionally, for robots with more than six degrees of freedom, joint space is higher dimensional than task space. Thus, in OPTIMUS, we instead use task-space control. We generate task space trajectories by performing forward kinematics on joint-space way-points given by the TAMP planner,

then execute an operational-space (task-space) controller [59] to achieve them. Sec. 4.2 conducts an experiment comparing policy the trained policy success rate with joint-space actions versus task-space actions. Fig. 6 shows that task-space actions enable higher success rates.

Filtering Demonstrations: Since there is variance in run-time due to random sampling and the TAMP system is not guaranteed to converge in plan cost within the fixed time limit, some plans and thus behaviors may be sub-optimal. This data can often hamper policy learning by operating outside of the space of nominal solution trajectories. Training on this data from the TAMP system increases the likelihood of the agent leaving areas of high state space coverage, which produces policies that exhibit heightened compounding error. To ease the burden on the policy, we curate the data using several trajectory pruning rules. During data collection, we employ joint-space path smoothing. However, straight-line paths through joint-space are non-linear in task space, resulting in longer motions in the learner's action space. Therefore, we propose **two data pruning rules** (Fig. 2 *column 2*) to filter TAMP demonstrations. First, we remove outlier trajectories that have task-space length greater than two standard deviations away from the mean trajectory length, which can be viewed as randomly restarting TAMP episodes to reduce plan variance. Second, we impose a containment constraint in the form of a bounding box in visible workspace and prune out trajectories in which the end-effector pose exits the box. Sec. 4.2 and Fig. 6 illustrate that the combination of these rules does improve performance by comparing the trained policy success rate with and without filtering.

3.3 Training Imitation Policies at Scale

We now describe the imitation pipeline in OPTIMUS. Given large, diverse datasets from TAMP, we perform offline behavior cloning to distill the TAMP expert into a visuomotor policy.

OPTIMUS Architecture: Our policy must operate over a history of multiple camera views and proprioception, output low-level task space actions, and execute in real time. To that end, we design our policy π, visualized in Fig. 3, as a Transformer operating over a history of observations h, in which each token corresponds to a single observation time-step. As a result, the Transformer can efficiently attend to all observations as the Transformer context length is set to h. To produce a single input token for a time-step t, we first embed each input, images from cameras $1, ..., N$ (I_1^t through I_N^t) as well as proprioception p_t, into fixed dimensional vector spaces. For embedding images, we use the vision backbone from Mandlekar et al. [29]: ResNet-18 [60] with a spatial softmax [61] output activation. We then fuse the inputs for a single time-step to produce z_t, a vector matching the Transformer embedding dimension, by concatenating and performing an MLP projection. The Transformer attends to each token z_t and outputs a distribution over action a_t corresponding to the current time-step.

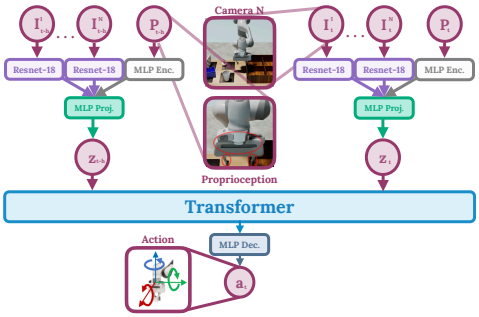
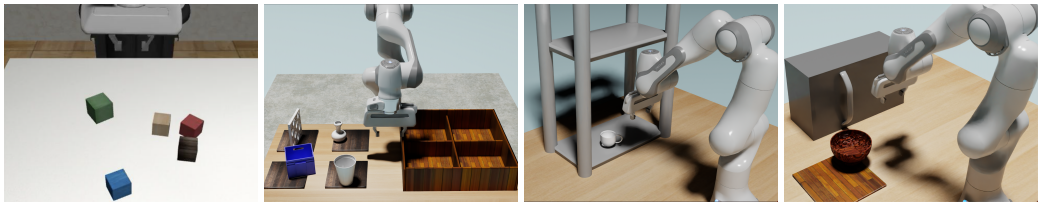


Figure 3: **OPTIMUS policy architecture**. The model takes as input multiple images and proprioception information per time-step, with a context of h. We encode the input using Resnet-18 for images and a MLP for the low-dimensional observations. We concatenate the embeddings, project them into the Transformer embedding dimension and pass them to the Transformer, which predicts an embedding that is decoded into an action.

The data distribution outputted by the TAMP supervisor is heavily multi-modal, from the diversity in planned paths to the variety of grasps and placements per object. As a result, we use a Gaussian Model Mixture (GMM) output distribution with $K = 5$ components for the policy from Mandlekar et al. [29] and train the model using log likelihood. As in [29], we find that this loss function provides a significant improvement over the standard MSE loss, which produces a unimodal policy.

4 Experimental Evaluation

In our experimental evaluation of OPTIMUS, we aim to answer the following questions: 1) Can imitating TAMP enable end-to-end policies to acquire long-horizon behaviors? 2) Does TAMP allow networks to solve complex manipulation tasks involving 3D obstacles and articulated objects? 3) Does diverse environment generation along with TAMP data-collection enable large-scale behavior



(a) StackFive (b) PickPlaceFour (c) Shelf (d) Microwave

Figure 4: **Environment Visualizations**. We evaluate OPTIMUS on long-horizon block stacking (a), multi-step pick-place (b), shelf object manipulation (c), and articulated object manipulation (d).

Dataset	BC-MLP	BC-RNN	BeT	OPTIMUS
Stack	100	100	100	100
StackThree	98	88	76	**100**
StackFour	83	77	61	**96**
StackFive	57	57	45	**70**

Dataset	BC-MLP	BC-RNN	BeT	OPTIMUS
PickPlaceTwo	96	**98**	80	96
PickPlaceThree	62	81	46	**91**
PickPlaceFour	33	38	22	**60**

Figure 5: **Long Horizon Manipulation Results.** (left) Performance is shown in terms of task success rate. While all methods are able to solve single-step block stacking, only OPTIMUS is able to solve longer-horizon variants. (right) For long-horizon manipulation, while the baselines are competitive with OPTIMUS on PickPlaceTwo, OPTIMUS demonstrates significant improvement in success rate as the number of objects increases.

learning? We begin by describing the datasets, tasks, and protocols that we use to evaluate OPTIMUS. We then proceed to experimentally evaluate OPTIMUS.

Datasets and Tasks: We evaluate OPTIMUS across block stacking, pick and place, shelf manipulation and articulated object manipulation. (Fig. 4). Our block stacking tasks have two (Stack), three (StackThree), four (StackFour) or five (StackFive) blocks, with 1K, 2K, 5K, and 7K demonstrations respectively. For pick and place, we have: PickPlace-1, pick-place with a single object using 1K demos, and pick-place with two (PickPlaceTwo), three (PickPlaceThree), and four objects (PickPlaceFour) into separate bins. Finally we have two tasks in which the goal is for the agent to move the object to the target location while maneuvering in tight spaces (Shelf-1) or first pulling open a microwave door (Microwave-1), for which we generate 1K demonstrations each. For PickPlace, Shelf, and Microwave, we additionally evaluate two multi-task variants, in which we sample a set of 19 and 72 objects from ShapeNet. We collect a 1K demonstrations per object, with 19K and 72K total trajectories, resulting in the following datasets: Pickplace-19, PickPlace-72, Shelf-19, Shelf-72, Microwave-19, Microwave-72. **See Appendix Sec. C for complete task descriptions and details.**

Evaluation Protocol: We evaluate BC-MLP [62] and BC-RNN [29], which consist of a Resnet-18 backbone followed by an MLP and an LSTM [63]. Additionally, we compare against Behavior Transformer (BeT) [43], which discretizes the dataset into clusters using K-Means and uses a Transformer model to predict a cluster center and an offset, in order to handle multi-modal data. Each method uses on the order of magnitude of 30M parameters, uses the same architecture as OPTIMUS for the vision-backbone, and leverages the data-pipeline we propose in Sec. 3. For each task, we evaluate on a dataset of unseen initial environment states. For single-task results, we evaluate using 50 problems and average across 3 random seeds per run. For multi-task results, we evaluate using 10 problems per task, with a single seed per run. We use task success rate as our evaluation metric, which is 1 if all objects are in a goal arrangement and 0 otherwise.

4.1 Learning Results

We first show that OPTIMUS can imitate the TAMP system to high fidelity on simple, shorter horizon tasks. We then extend our evaluation to the long-horizon regime in which the task complexity grows significantly with the number of objects. Next, we move beyond the table-top manipulation setting and train policies to solve tasks involving a shelf and microwave. Finally, we demonstrate that OPTIMUS can enable multi-task policies that can manipulate a wide range of objects.

OPTIMUS imitates TAMP to high fidelity on simple pick-and-place tasks. On Stack (Fig. 5), we find that OPTIMUS and the baselines are all able to achieve 100% performance on the task. On the other hand, on PickPlace-1, (Fig. 5), while the baselines achieve high success rates of up to 97%, only our method is able to solve the task at 100% success rate. These results demonstrate that on simple

tasks, OPTIMUS can fit well to the output of the TAMP system, even though OPTIMUS does not have access to any privileged information. We note that even with significant tuning, BeT struggles to fit to TAMP data on most of our tasks. We hypothesize that this may be due to the difficulty of fitting K-Means as the dataset size increases, especially as TAMP generated datasets contain on the order of 1-100K trajectories depending on the task. As a result, the cluster centers can be highly inaccurate, increasing the burden on the transformer to fit appropriate offsets.

OPTIMUS enables visuomotor policies to solve manipulation tasks with up to 8 stages. We first evaluate on long-horizon block stacking, a task that is difficult because the stack of blocks becomes more unstable as its height grows. We train visuomotor policies across StackThree, StackFour, and StackFive, and visualize the results in Fig. 5. OPTIMUS outperforms the baseline methods while achieving near-perfect performance across each task. Multi-step pick-place is even more difficult as the network must learn to fit a variety of different grasps for each object. We plot the results for the multi-step pickplace tasks in Fig. 5. We find that while BC-RNN outperforms OPTIMUS on PickPlaceTwo, OPTIMUS exhibits a large performance improvement on PickPlaceThree and Four. These results demonstrate that with either primitive or general-purpose rigid objects, it is possible to train policies to perform long-horizon behaviors consisting of up to 8 pick and place operations or 40 TAMP high-level actions, with high success rates of **70%** and **60%** respectively. An important take-away from these results is that for longer-horizon tasks, the Transformer policy architecture we develop in OPTIMUS greatly outperforms MLPs and RNNs.

Guided TAMP	OPTIMUS
88	**90**

Table 1: **Comparison against Guided TAMP.** Results are in terms of task success rate.

We additionally compare against prior work on imitating TAMP [28] on the Robosuite [58] PickPlace task, which involves picking and placing four fixed objects: a milk carton, a soda can, a cereal box and a piece of bread, into separate bins. In contrast to PickPlaceFour, Robosuite PickPlace can be solved with top-down, axis-aligned grasps due to the simplicity of the object geometry. However, the initial configurations are more challenging as all the objects are placed together in the same bin. We generate 25K demonstrations of the task using our system. As we show in Table 1, OPTIMUS achieves favorable results to Guided TAMP (90% vs. 88%) without requiring access to privileged state information, a fixed set of ground actions or online supervision.

Dataset	BC-MLP	BC-RNN	BeT	OPTIMUS
PickPlace-1	94	97	85	**100**
PickPlace-19	61	58	50	**85**
PickPlace-72	50	49	41	**75**
Shelf-1	**91**	88	70	**91**
Shelf-19	48	31	26	**66**
Shelf-72	30	36	13	**48**
Microwave-1	73	77	51	**86**
Microwave-19	24	41	31	**61**
Microwave-72	23	29	16	**47**

Table 2: **Single and Multitask Results across PickPlace, Shelf, Microwave.** Performance is shown in terms of task success rate. While the baselines are competitive with OPTIMUS on the single task variants of each task, OPTIMUS greatly outperforms the baselines as the number of objects increases across all tasks.

OPTIMUS can also solve tasks requiring obstacle awareness and skills beyond pick-and-place. On Shelf-1, OPTIMUS is able to grasp then place the object in the middle rung of the shelf at high success rates of 91% shown in Table 2. On Microwave-1 OPTIMUS outperforms the baselines by nearly 10%, achieving 86% success rate overall. This is likely because OPTIMUS is able to better fit the data in the multi-step manipulation regime, as noted in the prior section. The results on the Shelf and Microwave tasks demonstrate that OPTIMUS can learn to solve difficult manipulation tasks that require obstacle awareness and the ability to manipulate articulated objects.

OPTIMUS can learn to adapt its behavior based on the scene configuration. As we describe in the Appendix, OPTIMUS is able to learn to adapt its task plan to produce additional stacking operations (StackAdapt) or clear the area in front of the microwave (MicrowaveAdapt) achieving 96% and 75% success. OPTIMUS is able to generalize to unseen receptacle sizes, achieving 80% and 70% success rate on held out shelves and microwaves.

We next evaluate the ability of our TAMP generation pipeline to collect diverse datasets in order to train large-scale policies. We add variety in the form of objects with differing geometries, requiring a single network to learn a range of manipulation behaviors end-to-end.

OPTIMUS achieves high success rates on vision-based manipulation tasks with up to 72 objects. For each task: PickPlace, Shelf, and Microwave, we evaluate on their 19 and 72 object variants (Table 2). On the 19 object tasks, OPTIMUS achieves 85%, 66%, and 61% in greatly outperforming the best baseline for each task: 61%, 48%, and 41%. Similarly on the 72 object tasks, we find that

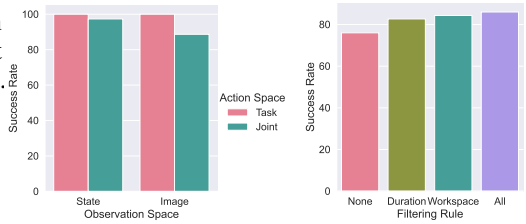
OPTIMUS obtains 75%, 48% and 47% success rates, in comparison to 50%, 36% and 29% for the best baseline. From these results, we note two important points: 1) Transformer-based architectures such as OPTIMUS are highly effective for multi-task imitation learning: they greatly outperform MLPs and RNNs. 2) While the single task variants of these tasks are solved at high success rates, performance drops significantly in the multi-task case, particularly for more challenging manipulation tasks such as Shelf and Microwave, indicating further work remains to bridge that gap.

Finally, we highlight three advantages of OPTIMUS over the TAMP system: 1) success rate improvement over the TAMP supervisor, 2) faster run-time, 3) operation from image instead of state input. We evaluate TAMP and OPTIMUS on all of the single task datasets and find that on average, **OPTIMUS almost doubles the performance of the TAMP supervisor** (87% vs. 52%). Additionally, we evaluate the run-time of OPTIMUS against TAMP by computing the average time per step for both systems across 100 trials. Overall, **OPTIMUS is 5-7.5x faster than TAMP** (.021/.031s vs. .15s per action). **See Appendix Sec. B for analysis of scene adaptation and TAMP comparison results.**

4.2 Ablations

We now analyze critical design decisions in OPTIMUS, varying each component of interest while holding the rest of OPTIMUS constant.

Task-space control greatly improves visuomotor learning performance. In the state-based regime, we find that the choice of action space makes little difference, both control schemes achieve high performance (98% for joint space vs. 100% for task space). However, when training with visual observations, we find that there is a large gap (86% vs. 100%) in performance between joint control and task-space control. We hypothesize that this may be due to the challenge of learning an inverse kinematics mapping from visual input, *i.e.* mapping 2D observed pixel locations to 7DOF joint angles.



(a) Arm Action Space (b) Data Curation

Figure 6: **Effect of Arm Action Space Choice and Data Filtering Rules.** (a) OPTIMUS task success rate improves with task-space over joint-space actions when using image observations. Image observation-space policies perform comparably to the privileged state-based policies when using task-space actions. (b) Performance is improved by filtering TAMP success trajectories based on the visible workspace and their duration.

Data filtration results in a significant improvement in policy success rates. On the Microwave-1 task, we train four policies with different filtration schemes: 1) no filtering (None), 2) filtering based on trajectory length (Duration) 3) filtering based on visible workspace limits (Workspace), and 4) Duration and Workspace combined (Both). We find (Fig. 6) that policies trained on unfiltered data perform worse when compared to those trained on filtered data. Specifically, workspace filtering has a greater impact than Duration. Combining both forms of filtering results in the greatest performance improvement of 10% and demonstrates that filtering TAMP trajectories is crucial to obtaining high success rates for learned policies.

5 Limitations and Future Work

In this work, we propose an approach for distilling privileged TAMP experts into large-scale visuomotor policies. We generate large, diverse datasets and train high-capacity Transformer models to solve challenging, long-horizon manipulation tasks without task, state, or environment knowledge. Even so, there are limitations to OPTIMUS and scope for future work. First, for OPTIMUS to be able to solve a task, TAMP needs to be capable of solving it at training time, which could prevent OPTIMUS from being applied to tasks challenging for TAMP, such as deformable object manipulation. However, recent work has shown that TAMP can operate in regimes beyond pick-and-place by leveraging skill learning [64, 65], which OPTIMUS could distill as well. Second even with OPTIMUS, there is a significant drop in success rate with increasing task difficulty and number of objects (Sec. 4.1), which suggests further work on multi-task learning is necessary to bridge that gap. Finally, we describe two possible approaches to applying OPTIMUS to the real world: 1) Collect TAMP data in a controlled setting at training time using a pose estimation system (e.g. AR tags) and distill using OPTIMUS. At test time, OPTIMUS would not require pose estimation, while providing significantly faster execution and superior task performance. 2) Transfer imitation policies trained in simulation either zero-shot or via fine-tuning to real data [66, 67, 68, 69]. We leave this extension of OPTIMUS to future work.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[4] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

[5] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[6] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[8] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[9] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

[10] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2204.01691*, 2022.

[11] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

[12] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[13] Peter F Hokayem and Mark W Spong. Bilateral teleoperation: An historical survey. *Automatica*, 42(12):2035–2057, 2006.

[14] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[15] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[16] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on robot learning*, pages 1113–1132. PMLR, 2020.

[17] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*, 2020.

[18] Zichen Jeff Cui, Yibin Wang, Nur Muhammad, Lerrel Pinto, et al. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

[19] Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task-agnostic offline reinforcement learning. *arXiv preprint arXiv:2209.08959*, 2022.

[20] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[21] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

[22] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[23] Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real-world robotic reinforcement learning. *arXiv preprint arXiv:2004.12570*, 2020.

[24] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

[25] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[26] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. *arXiv preprint arXiv:2210.13702*, 2022.

[27] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tom´as Lozano-P´erez. Integrated Task and Motion Planning. *Annual review of control, robotics, and autonomous systems*, 4, 2021.

[28] Michael James McDonald and Dylan Hadfield-Menell. Guided imitation of task and motion planning. In *Conference on Robot Learning*, pages 630–640. PMLR, 2022.

[29] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[30] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[31] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. *Proceedings 2002 IEEE International Conference on Robotics and Automation*, 2:1398–1403 vol.2, 2002.

[32] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference in Robot Learning*, volume abs/1709.04905, 2017.

[33] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. Robot programming by demonstration. In *Springer Handbook of Robotics*, 2008.

[34] Sylvain Calinon, Florent D'halluin, Eric L. Sauser, Darwin G. Caldwell, and Aude Billard. Learning and reproduction of gestures by imitation. *IEEE Robotics and Automation Magazine*, 17:44–54, 2010.

[35] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *arXiv preprint arXiv:1710.04615*, 2017.

[36] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations. *arXiv preprint arXiv:2003.06085*, 2020.

[37] Chen Wang, Rui Wang, Danfei Xu, Ajay Mandlekar, Li Fei-Fei, and Silvio Savarese. Generalization through hand-eye coordination: An action space for learning spatially-invariant visuomotor control. *arXiv preprint arXiv:2103.00375*, 2021.

[38] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[39] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[40] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[41] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv: Arxiv-2210.03094*, 2022.

[42] Sudeep Dasari and Abhinav Gupta. Transformers for one-shot visual imitation. In *Conference on Robot Learning*, pages 2071–2084. PMLR, 2021.

[43] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *arXiv preprint arXiv:2206.11251*, 2022.

[44] Marc A Toussaint, Kelsey Rebecca Allen, Kevin A Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. 2018.

[45] Mohak Bhardwaj, Sanjiban Choudhury, and Sebastian Scherer. Learning heuristic search via imitation. In *Conference on Robot Learning*, pages 271–280. PMLR, 2017.

[46] Ahmed H Qureshi, Anthony Simeonov, Mayur J Bency, and Michael C Yip. Motion planning networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2118–2124. IEEE, 2019.

[47] Adam Fishman, Adithyavairan Murali, Clemens Eppner, Bryan Peele, Byron Boots, and Dieter Fox. Motion policy networks. *arXiv preprint arXiv:2210.12209*, 2022.

[48] Caelan Reed Garrett, Chris Paxton, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Dieter Fox. Online replanning in belief space for partially observable task and motion problems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5678–5684. IEEE, 2020.

[49] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[50] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 440–448, 2020.

[51] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. Acronym: A large-scale grasp dataset based on simulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6222–6227. IEEE, 2021.

[52] Patrick Beeson and Barrett Ames. {TRAC-IK}: An open-source library for improved solving of generic inverse kinematics. 11 2015.

[53] James J Kuffner Jr. and Steven M LaValle. RRT-Connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[54] William Vega-Brown and Nicholas Roy. Asymptotically optimal planning under piecewise-analytic constraints. 2016. URL http://www.wafr.org/papers/WAFR_2016_paper_11.pdf.

[55] Kris Hauser and Victor Ng-Thow-Hing. Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. pages 2493–2498, 2010.

[56] Kyle Hsu, Moo Jin Kim, Rafael Rafailov, Jiajun Wu, and Chelsea Finn. Vision-based manipulators need to also see from their hands. *arXiv preprint arXiv:2203.12677*, 2022.

[57] Roberto Martín-Martín, Michelle A Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh Garg. Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1010–1017. IEEE, 2019.

[58] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

[59] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[61] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[62] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5628–5635. IEEE, 2018.

[63] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[64] Zi Wang, Caelan Reed Garrett, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning compositional models of robot skills for task and motion planning. *The International Journal of Robotics Research*, 40(6-7):866–894, 2021.

[65] Aidan Curtis, Xiaolin Fang, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Caelan Reed Garrett. Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1940–1946. IEEE, 2022.

[66] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. 2022.

[67] Pierre-Louis Guhur, Shizhe Chen, Ricardo Garcia, Makarand Tapaswi, Ivan Laptev, and Cordelia Schmid. Instruction-driven history-aware policies for robotic manipulations. 2022.

[68] Mohi Khansari, Daniel Ho, Yuqing Du, Armando Fuentes, Matthew Bennice, Nicolas Sievers, Sean Kirmani, Yunfei Bai, and Eric Jang. Practical imitation learning in the real world via task consistency loss. *arXiv preprint arXiv:2202.01862*, 2022.

[69] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

[70] Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135*, 2022.

[71] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[72] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014.

[73] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.

[74] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[75] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[76] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[77] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, 25 (5):581–603, 2011.

[78] Micha Hersch, Florent Guenter, Sylvain Calinon, and Aude Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics*, 24(6): 1463–1467, 2008.

[79] Scott Niekum, Sachin Chitta, Andrew G Barto, Bhaskara Marthi, and Sarah Osentoski. Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems*, volume 9, pages 10–15607. Berlin, Germany, 2013.

[80] Baris Akgun, Maya Cakmak, Jae Wook Yoo, and Andrea Lockerd Thomaz. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 391–398, 2012.

[81] Ajay Mandlekar, Jonathan Booher, Max Spero, Albert Tung, Anchit Gupta, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei. Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity. *arXiv preprint arXiv:1911.04052*, 2019.

[82] Albert Tung, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Learning multi-arm manipulation through collaborative teleoperation. *arXiv preprint arXiv:2012.06738*, 2020.

[83] Josiah Wong, Albert Tung, Andrey Kurenkov, Ajay Mandlekar, Li Fei-Fei, Silvio Savarese, and Roberto Martín-Martín. Error-aware imitation learning from teleoperation data for mobile manipulation. In *Conference on Robot Learning*, pages 1367–1378. PMLR, 2022.

[84] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.

[85] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Human-in-the-loop imitation learning using remote teleoperation. *arXiv preprint arXiv:2012.06733*, 2020.

[86] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016.

[87] Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. *arXiv preprint arXiv:2109.08273*, 2021.

[88] Ryan Hoque, Ashwin Balakrishna, Carl Putterman, Michael Luo, Daniel S Brown, Daniel Seita, Brijen Thananjeyan, Ellen Novoseller, and Ken Goldberg. Lazydagger: Reducing context switching in interactive imitation learning. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 502–509. IEEE, 2021.

[89] Shivin Dass, Karl Pertsch, Hejia Zhang, Youngwoon Lee, Joseph J Lim, and Stefanos Nikolaidis. Pato: Policy assisted teleoperation for scalable robot data collection. *arXiv preprint arXiv:2212.04708*, 2022.

[90] Ahmed H Qureshi, Arsalan Mousavian, Chris Paxton, Michael C Yip, and Dieter Fox. Nerp: Neural rearrangement planning for unknown objects. *arXiv preprint arXiv:2106.01352*, 2021.

[91] Danny Driess, Jung-Su Ha, Marc Toussaint, and Russ Tedrake. Learning models as functionals of signed-distance fields for manipulation planning. In *Conference on Robot Learning*, pages 245–255. PMLR, 2022.