Modular Manipulation: An Effective Marriage of Classical Planning, Vision, and RL

Anonymous Author(s)
Affiliation
Address
email

Abstract: Manipulation is a challenging problem in robotics since the robot has to reason about making and breaking contact, which is highly discontinuous and difficult to model, while controlling the arm accurately and precisely. Long horizon manipulation is even more difficult as it requires reasoning over many steps, making the problem exponentially more complex. While classical, sense-plan-act style approaches can perform long-horizon reasoning, they struggle to produce dexterous behavior. End-to-end approaches, on the other hand, struggle with long-horizon reasoning but excel at producing low-level control policies. In this work, we aim to combine the best of both worlds by taking a novel approach called Modular Manipulation (M&M): using learned modules for sensing (vision) and acting (control) while leveraging classical planning to sequence skills. We empirically demonstrate that M&M is capable of solving over 20 challenging single and multi-stage manipulation tasks from raw visual input, out-performing classical and end-to-end approaches. Our code will be made publicly available. Video results at modularmanipulation.github.io/.

Keywords: Reinforcement Learning, Motion Planning, Manipulation

1 Introduction

Robotic manipulation, a pivotal subject with a rich history of research [1], is essential for enabling autonomous systems to interact with the world. Despite its importance, manipulation poses a series of challenges due to the difficulty of accurately modeling discontinuous contact dynamics, and requirement to reason about the motion of external objects, particularly over long horizons. Take, for instance, a nut assembly task (Fig. 1), which demands precisely grasping the nut, aligning it with the peg, and carefully inserting it, all whilst not hitting other obstacles in the environment.

Historically, manipulation has been approached via the Sense-Plan-Act (SPA) pipeline [2, 3, 4], which requires comprehensive understanding of the environment (sense), a model of the world (plan), and low-level controller (act). Traditional approaches range from manipulation planning [5, 6], grasp analysis [7], and Task and Motion Planning (TAMP) [8], to modern variants incorporating learned vision [9, 10, 11, 12]. The SPA framework's modular approach provides a level of interpretability, enabling better understanding of each decision-making process, which is crucial for safety considerations and error analysis. Additionally, the use of planning algorithms enable long horizon decision making over complex and high-dimensional configuration spaces. Despite these strengths, these approaches often fail in scenarios with contact-rich interactions, and their performance is hindered by cascading errors across modules often originating from imperfect vision. Furthermore, these approaches require significant manual engineering and systems effort to setup for each application.

Recently, end-to-end (E2E) methods, particularly those employing imitation learning (IL) and reinforcement learning (RL), have emerged as potent alternatives to traditional manipulation strategies. IL has the advantage of transferring expert knowledge directly to the robot, and it has demonstrated excellent performance on complex tasks [13, 14, 15, 16, 17, 18]. However, it relies on the availability of demonstration data, which limits scalability and adaptation to new or unexpected scenarios. RL, on the other hand, has the potential to solve complex tasks without the need for demonstration data

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

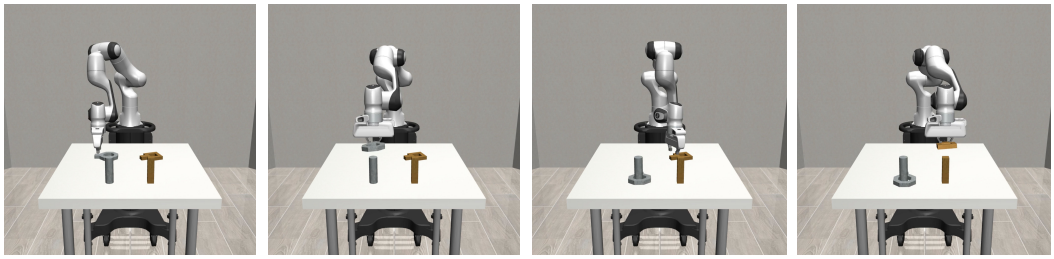

Figure 1: **Long horizon tasks visualization.** We visualize states achieved by our contact-free motion policy. The policy predicts end-effector poses near each nut and peg from RGB-D input and motion-plans to achieve them. From these states, M&M efficiently learns to solve the contact-rich portions of the task using RL.

or known models via online interaction, as demonstrated by a variety of real world manipulation studies [19, 20, 21, 22, 23, 24]. However, the challenge of exploration in RL, especially in high-dimensional spaces characteristic of robotic manipulation tasks, is nontrivial. Even with dense rewards, RL can require millions of samples to learn a single-stage task and may struggle with longer-horizon tasks in which high-level reasoning and low-level control must be learned simultaneously.

In this paper, we propose an approach that combines the strengths of SPA and E2E methods and addresses their weaknesses called Modular Manipulation (M&M). Our core intuition is that any manipulation task can be decomposed into alternating phases of **contact-free motion** and **contact-rich interaction**. The E2E approach excels at learning contact-rich interaction skills or difficult-to-model interactions, while the SPA pipeline's motion planning is adept at handling contact-free scenarios. Hence, we alternate between these two in our approach. Our method replaces the Sensing and Action stages in the SPA pipeline with learned components, retaining only the high-level Plan.

Our contributions are: A novel framework for long-horizon manipulation that tightly integrates sampling-based motion-planning with learning for low-level control and vision; An extensive experimental evaluation demonstrating that M&M can solve 10+ long-horizon manipulation tasks with up to 5 stages at success rates of over 80%, producing agents that solve long-horizon, contact-rich tasks such as NutAssembly at over 95% success rate.

2 Related Work

Planning with Modern Vision: With the advent of deep learning, classical planning methods for manipulation have been able to take advantage of significantly improved state estimation capabilities [25, 26, 27] to be able to generalize much more effectively. In particular, recent methods such as Contact Graspnet [12], DexNet [9, 10, 28, 29], NERF [30] based approaches such as DexNERF [31] and GraspNERF [32] are effective at producing accurate grasp poses, for consumption by a downstream motion-planner, even for unseen objects under challenging conditions such as transparency. Yet core limitations of classical methods persist: cascading failures due to imprecise state estimation and difficulty in handling contact-rich settings beyond grasping. We address these limitations by combining planning with learned local interaction policies.

End-to-end Visuomotor RL is an appealing formulation in principle as the agent can *learn* from online interaction and propagrate information throughout the entire pipeline, enabling it to avoid the cascading failures of SPA and acquire dexterous behaviors without access to any privileged information or human design biases. Recent work has greatly advanced this field, from model-free methods based on data augmentation [33, 34, 35, 36] to model-based algorithms based on variants of Dreamer [37, 38, 39, 40] that are able to train models to efficiently solve a large array of challenging visuomotor control tasks. However, in manipulation settings, these algorithms still struggle beyond the single-stage regime, often because the sample complexity to solve a given task via RL grows exponentially with the horizon. In our work, we simultaneously cut down on the learning horizon and diminish the exploration problem by imposing a general inductive decomposition of manipulation tasks into sequential phases of contact-free and contact-rich motions.

Planning and Reinforcement Learning: Recent work has begun to explore the integration of motion-planning and RL to combine the advantages of both paradigms [41, 42, 43, 44, 45, 46].

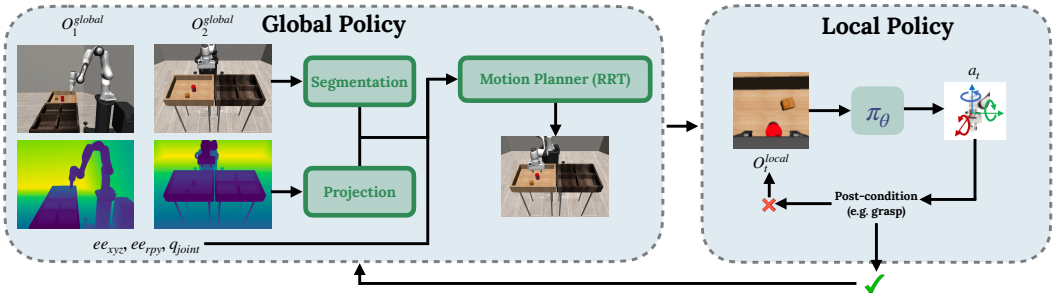

Figure 2: **Method overview.** M&M decomposes manipulation tasks into alternating phases of contact-free motion generation solved by a Global Policy (*left*) and a contact-rich interaction policy learned via RL (*right*). The Global Policy takes in two calibrated RGB-D fixed views of the scene, O_1^{global} and O_2^{global} as well as the proprioception $ee_{xyz}, ee_{rpy}, q_{joint}$ and performs visual motion planning to the target object (q_{target}). The Local Policy consumes a local wrist camera view O_t^{local}, acting until a post-condition is triggered.

GUAPO Lee et al. [41] is conceptually most similar to our method, yet their system considers the single-stage manipulation regime and is concerned with keeping the RL-agent in areas of low pose-estimator uncertainty. Our method instead considers long-horizon manipulation tasks by encouraging the RL-agent to follow a high-level plan using contact-free motion-policy. MoPA-RL [42] also bears resemblance to our method, yet it opts to learn when to use the motion-planner via RL, requiring the RL agent to discover the right decomposition of planner vs. control actions on its own. Furthermore, rollouts of trajectories using MoPA can result in the RL agent choosing to motion-plan multiple times in sequence, which is inefficient - one motion-planner action is sufficient. We instead explicitly decompose manipulation tasks into sequences of contact-free actions (motion-planner) and contact-rich (RL).

3 Modular Manipulation: A General Decomposition for Manipulation

In this section, we describe our method for addressing the long-horizon manipulation problem, M&M, outlined in Fig. 2. Given a coarse semantic plan of the task, we use motion-planning to generate contact-free motion to points from which we can efficiently train local interaction policies using RL.

Background: We consider Partially Observed Markov Decision Processes (POMDP) of the form $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, p_0, \mathcal{O}, p_O, \gamma)$. $\mathcal{S}$ is the set of environment states, $\mathcal{A}$ is the set of actions, $\mathcal{T}(s' \mid s, a)$ is the transition probability distribution, $\mathcal{R}(s, a, s')$ is the reward function, p_0 is the distribution over the initial state $s_0 \sim p_0$, $\mathcal{O}$ is the set of observations, p_O is the distribution over observations conditions on the state $O \sim p_O(O|s)$ and γ is the discount factor. In our case, the observation space is the set of all possible camera images and the reward function is assumed to defined by the environment. The agent's goal is to maximize the expected sum of rewards over the trajectory, $\mathbb{E}\left[\sum_t \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})\right]$. Additionally, we describe motion-planning, which is a core component of M&M. Given a start configuration q_0 and a goal configuration q_g, motion-planning aims to find a trajectory of way-points τ that form a collision-free path between q_0 and q_g. In our setting, q represents the joint angles of a robot arm. In such high-dimensional spaces, it is not possible to analytically plan a path through space. Hence motion-planning algorithms for manipulators leverage gradient-based optimization, search or sampling in order to find a feasible collision-free path through space. For M&M, we take advantage of sampling-based motion-planning due to its minimal setup requirements (only collision-checking) and favorable performance on planning for robot arms.

3.1 Modular Manipulation

We posit that any manipulation task can be decomposed into alternating phases of contact-free motion and contact-rich interaction. By explicitly separating tasks into sequential phases of contact-free motion to semantically meaningful points followed by contact-rich interaction with the object of interest, we can take advantage of the strengths of classical planning and learning, while simultaneously diminishing their weaknesses in order to obtain the best of both worlds. Specifically, classical planning excels at long-horizon and geometric reasoning, which we leverage to produce a global

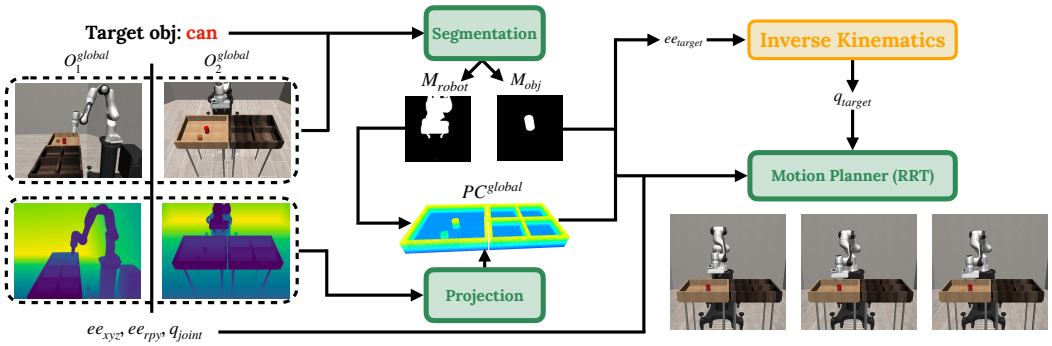

Figure 3: **Global Policy.** Inputs to the Global Policy are two calibrated RGB-D fixed views, O_1^{global} and O_2^{global}, the proprioception $ee_{xyz}, ee_{rpy}, q_{joint}$ and the target object. It performs visual motion planning to the target object by computing a scene point-cloud (PC^{global}), segmenting the target object (M_{obj}) to estimate its pose (q_{target}), segmenting the robot (M_{robot}) to remove it from PC^{global} and motion planning using RRT.

policy that addresses the long-horizon problem. Reinforcement learning (RL), on the other hand, excels at producing short-horizon dexterous behaviors which we use to learn local interaction policies that can be challenging to engineer using a classical pipeline.

In order for the global policy to know where to go and for the local policy to know when to finish executing and hand back control to the global policy, M&M requires a high-level plan to break down the task. In a TAMP pipeline, this would be accomplished by a task planner reasoning jointly with a motion-planner, requiring a defined set of action-types (primitives) and a fixed set of objects in order to produce a geometrically feasible task plan. In our setting, the global policy simply needs to know which object of interest to perform manipulation and whether it is grasping, placing, or pulling, etc. Furthermore, since we are learning low-level interaction via RL, we simply need to specify the post-condition of the interaction at each stage to know when its complete. In our experiments, we assume such a coarse, high-level plan of objects and associated post-conditions is provided as these quantities do not change based on the scene for the benchmarks we evaluate.

3.2 Global Policy

For many manipulation tasks of interest, the high-level semantics of *what* to do are straightforward to specify, yet the low-level specifics of *how* to achieve them can be challenging. The Global Policy (Fig. 3) in M&M uses such coarse high-level plans, specifically which objects/regions to perform manipulation in and in what order, to handle the contact-free motion phrase: reaching states from which control is passed to the Local Policy. To do so, the Global policy involves three steps: 1) **Projection**, which computes a scene pointcloud using multi-view RGB-D information 2) **Segmentation**, which segments the scene to obtain a segmentation mask of the target object and thereby a target pose for planning and 3) **Motion Planning** which uses the scene pointcloud to produce a collision-free path to the target pose.

The inputs to the Global Policy are O_1^{global} and O_2^{global}, which are RGB-D calibrated camera views of the scene, the current joint position q_{joint} and the target object around which the Local Policy must perform manipulation. From this information, the Global Policy must solve for a collision free path to a region near the target. This problem can be addressed by classical planning, namely motion planning. In particular, we opt to use *sampling-based* motion-planning due to its simplicity of implementation and ability to scale to 7-DOF manipulators. In order to run the motion-planner, we require a collision checker, which we implement using pointclouds. To compute the target object position, we use predicted segmentation along with calibrated depth, as opposed to a dedicated pose estimation network, primarily because state of the art segmentation models [47, 48] have significant zero-shot capabilities across objects.

Projection: In this step, we project the depth map from each global view of the scene, O_1^{global} and O_2^{global} into a point-cloud PC^{global} using their associated camera matrices K_1^{global} and K_2^{global}.

We perform the following processing steps to clean up PC^{global}: 1) cropping to remove all points outside the workspace 2) voxel down-sampling with a size of 0.005 m^3 to reduce the overall size of PC^{global} 3) outlier removal, that prunes points that are farther from their 20 neighboring points than the average in the point-cloud as shown in Fig. 3.

Segmentation: We compute masks for the robot (M_{robot}) and the target object (M_{obj}) by using a segmentation model $\mathcal{S}$ which segments the scene based on RGB input. In simulation, we opt to use simulator segmentation, as simulation visuals are drastically different from the real world. However, in the real-world, we would apply SAM [47] or Detic [48] which are capable of segmenting a wide array of objects. We reduce noise in the masks by filling holes, computing contiguous mask clusters and selecting the largest mask. We use M_{robot} to remove the robot from PC^{global}, in order to perform collision checking of the robot against the scene. Additionally, we use M_{obj} along with PC^{global} to compute the object point-cloud PC^{obj}, which we average to obtain an estimate of object position, the target end-effector position ee_{target} for the motion-planner.

Visual Motion Planning: Given the target end-effector pose ee_{target}, we use inverse kinematics (IK) to compute q_{target} and pass $q_{joint}, q_{target}, PC^{global}$ into a joint-space motion planner. To that end, we use a sampling-based motion planner, RRT-connect [49], to perform motion planning. In order to implement collision checking from vision, for a sampled joint-configuration q_{sample}, we compute the corresponding position of the robot mesh and compute the occupancy of each point in the scene point-cloud against the robot mesh. If the object is grasped, then we additionally remove the object from the scene pointcloud, compute its convex hull and use the signed distance function of the joint robot-object mesh for collision checking. As a result, the Global Policy operates entirely over visual input, and achieves a pose near the region of interest before handing off control to the local policy. We emphasize that the Global Policy *does not need to be perfect*, it just needs produce a reasonable initialization for the Local Policy to learn from.

3.3 Local Policy

We train an RL policy to perform high-frequency 6 DOF end-effector control along with gripper control from purely visual input. We train a single Local Policy across all stages of a task, sequencing between phases via the Global Policy, to optimize the overall reward function of the task. Note that this implies the value function of the Local Policy is shared across stages, enabling it to account for future decisions as well as inaccuracies in the Global Policy. To determine when a phase is complete, we check if the phase post-condition is triggered, which amounts to checking for grasping or placement, and can be estimated using vision, as we outline in the Appendix.

Instead of training the policy from a fixed view input, we use wrist camera observations only for the local policy. This design choice affords several unique properties that we validate in our experimental analysis: 1) improved learning efficiency and speed, 2) ease of chaining pre-trained policies. Local policies are uniquely capable of leveraging wrist only views because of the decomposition in M&M: the local policy simply has to learn interaction behaviors in a small region near the object, it has no need for a global view of the scene, in contrast to the standard end-to-end RL agent that would need to see a global view of the scene to know where to go to solve a task. For additional details in regarding the structure and training process of the Local Policy, see Appendix Section **??**.

4 Experimental Setup

Tasks: We conduct experiments on single and multi-stage manipulation tasks across four simulated environment suites (**Metaworld**, **Obstructed Suite**, **Kitchen** and **Robosuite**) which contain obstructed settings, contact-rich setups, and sparse rewards (Fig. B.1). See Appendix Section B) for additional environment details.

We consider the **Metaworld** benchmark [50], which contains a diverse array of manipulation tasks from which we select five challenging environments: `MW-Disassemble` (removing a nut from a peg), `MW-BinPick` (picking and placing a cube), `MW-Assembly` (picking and placing a nut on peg), and `MW-Hammer` (picking up a hammer and hitting a nail). The **ObstructedSuite** set of tasks, introduced

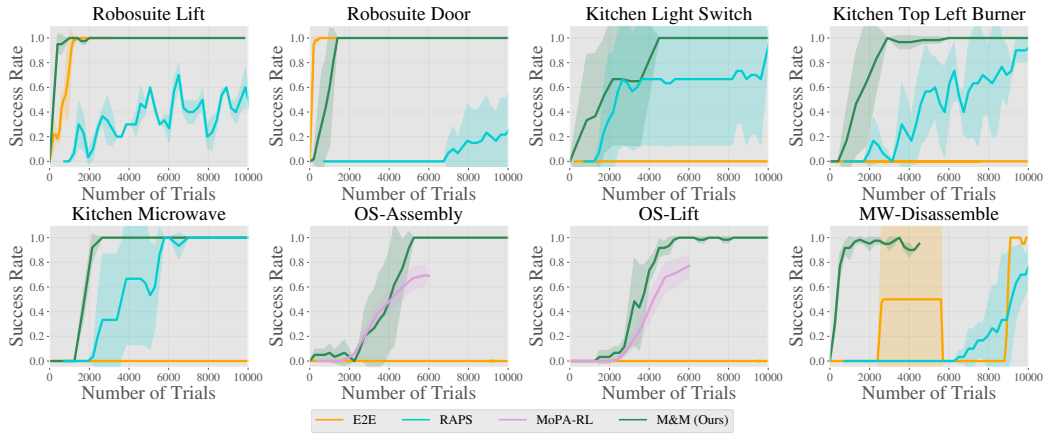

Figure 4: **Single Stage Results.** We plot task success rate as a function of the number of trials. M&M improves on the efficiency of the baselines across single-stage tasks (*plan length of 1*) in Robosuite, Kitchen, Metaworld, and Obstructed Suite, **achieving an asymptotic success rate of 100% on all 8 tasks**.

by Yamada et al. [42], consists of three manipulation tasks with significant obstructions `OS-Lift` (lift a cube in a tall box), `OS-Push` (push a block surrounded by walls), and `OS-Assembly` (avoiding obstacles to place table leg at target). These tasks almost necessitate motion-planning due to obstacles that block the target object from the agent.

The **Kitchen** [51, 52] environments test two aspects of our agent: its ability to handle sparse terminal rewards and its long-horizon manipulation capabilities. The single-stage kitchen tasks include `K-Slide` (push slide cabinet to the right), `K-Kettle` (place kettle on back stove), `K-Burner` (turn burner knob), `K-Light` (flick light switch to "on"), and `K-Microwave` (open microwave door). The multi-stage Kitchen tasks denote the number of stages in the name and include combinations of the aforementioned single tasks. Lastly, we evaluate the **Robosuite** benchmark [53], a series of manipulation tasks ranging from single stage (`RS-Lift` - lift a cube, `RS-Door` - open a door) to multi-stage (`RS-NutRound`,`RS-NutSquare`, `RS-NutAssembly` - pick-place nut(s) onto target peg(s) and `RS-Bread`, `RS-Cereal`, `RS-Milk`, `RS-Can`, `RS-CerealMilk`, `RS-CanBread` - pick-place object(s) into appropriate bin(s)). Unlike the other environment suites, which simplify aspects of the controller, Robosuite emphasizes realism and fidelity to real-world control, enabling us to highlight the potential of our method to be applied to real systems.

Baselines: The simplest baseline is end-to-end RL (**E2E**) using DRQ-v2 [33], a SOTA model-free visual RL algorithm also used to train our local policy. Next, we compare against RAPS [54], a hierarchical RL method that modifies the action space of the agent with engineered subroutines (primitives). RAPS demonstrates strong performance across a wide range of manipulation tasks, but is limited in expressivity due to its action space, unlike M&M. We evaluate **MoPA-RL** [42], which similar to M&M in its integration of motion-planning and RL but differs primarily in that it allows the RL agent to decide when and where to call the motion-planner. In initial experiments, we found that MoPA-RL performs poorly with visual input; we instead use reported numbers from the paper from experiments using state information on the Obstructed Suite of tasks. Finally, we consider a classical baseline based on sense-plan-act (**SPA**) which leverages our system for pose estimation and planning but operates low-level control through engineered primitives.

Experiment details: We evaluate all methods using visual input. SPA and M&M methods use $O_1^{global}, O_2^{global}$ and O^{local}. For E2E and RAPS, we provide the learner access to a single fixed view observation O_1^{global} for simplicity and speed of execution, as we did not find meaningful performance improvement in these baselines by incorporating additional camera views. We measure performance in terms of task success rate with respect to the number of trials (episodes). We do so to provide a fair metric for evaluating a variety of different low-level control implementations across M&M, RAPS, and E2E. Each method is trained for 10K episodes total. We train on each task using the default reward function without modification. For each method, we run 3 seeds on every task and average across 10 evaluations. For tabular results, we include a measure of standard deviation as well.

	RS-Bread	RS-Can	RS-Milk	RS-Cereal	RS-NutRound	RS-NutSquare	MW-BinPick	MW-Assembly	MW-Hammer
E2E	0 / 0	0.5 / .7	0 / 0	.05 / 0.07	.1 / .1	0 / 0	1.0 / 0	1.0 / 0	1.0 / 0
RAPS	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0.0 / 0	1.0 / 0	1.0 / 0
SPA	0.93 / 0.09	1.0 / 0	0.9 / 0.05	0.63 / 0.09	0.56/0.25	0.27/.21	1.0 / 0	0.5 / 0.08	1.0 / 0
M&M (ours)	1 / 0	1 / 0	1 / 0	1 / 0	.98 / .04	.97 / .02	1.0 / 0	1.0 / 0.0	1.0 / 0

Table 1: **Two Stage Results.** Performance is measured in terms of mean/std success rate on two-stage (*2 planner actions*) manipulation tasks. SPA is competitive with M&M on pick-place style tasks, but its performance drops considerably (100% to 56%/27%) on contact-rich tasks involving assembling nuts. E2E and RAPS are competitive with M&M on the Metaworld suite but makes little progress on the long-horizon tasks in Robosuite. M&M is able to solve each task with at least 97% success rate.

5 Results

M&M accelerates learning efficiency on a wide array of manipulation tasks. For single-stage manipulation, the Global Policy motion-plans to the target object, then hands off control to the Local Policy. In this setting, we observe improvements in learning efficiency (with respect to number of trials) as well as final performance in comparison to the learning baselines E2E, RAPS and MoPA-RL, across 11 tasks in Robosuite, Metaworld, Kitchen and ObstructedSuite (Fig. 4). M&M specially performs well on sparse reward tasks, such as in Kitchen, for which a key challenge is in figuring out which object to manipulate and where it is. Additionally, we observe qualitatively meaningful behavior using M&M: In Kitchen, M&M learns to use the gripper to grasp and turn the burner knob, unlike E2E or RAPS which end up using other joints flick the burner to the right position.

M&M efficiently solves tasks with obstructions by leveraging motion-planning. We now consider three tasks from the Obstructed Suite in order to highlight M&M's effectiveness at performing manipulation in the presence of obstacles. As we observe in Fig. 4, M&M is able to do so efficiently, solving each task within 5K episodes, while E2E fails to make progress. M&M is able to do so because the Global Policy handles the obstacle avoidance implicitly via motion-planning and initializes the Local Policy in an advantageous region near the target object. In contrast, E2E spends a significant amount of time attempting to reach the object in spite of the obstacles, failing to learn the task. While MoPA-RL is also able to solve the tasks, it requires more trials than M&M even though it operates over *privileged* state input, as the agent must simultaneously learn *when* and *where* to motion plan as well as *how* to manipulate the object.

M&M enables visuomotor policies to learn pick-and-place behaviors with up to 5 stages. For multi-stage manipulation, M&M is provided a high-level plan of target objects and associated post-conditions, the Global Policy achieves each target pose in sequence and the Local Policy interacts with the target object according to the overall reward function of the task. Two-stage manipulation results for pick and place style tasks across Robosuite and Metaworld are shown in Table 1. On the Robosuite tasks, E2E and RAPS generally fail to make progress: while they eventually learn to reach the object, they fail to consistently grasp it, let alone learn to place it in the target location. On the MetaWorld tasks, the baselines perform well on most tasks due to the heavily shaped rewards. SPA is able to achieve high performance across each Robosuite task (93%, 100%, 90%, 63% for Bread, Can, Milk, and Cereal), as the manipulation skills required are fairly simple: a top grasp and placement. However, only M&M is able to achieve perfect performance across each task, avoiding cascading failures from SPA by learning from online interaction.

On multi-stage manipulation tasks (involving 3-5 stages), we find that SPA performance drops significantly in comparison to M&M (43.16% vs. 90% on average). For multiple stages, the cascading failure problem becomes all the more problematic, causing SPA to fail at intermediate stages, while M&M is able to learn to adapt to imperfect Global Policy behavior via RL. See Table 2 for a detailed breakdown of the results.

M&M solves contact-rich, long-horizon manipulation tasks such as NutAssembly. In these experiments, we show that M&M can learn to solve contact-rich tasks (`RS-NutRound`, `RS-NutSquare`, `RS-NutAssembly`) that pose significant challenges for SPA due to the difficulty of designing manipulation behaviors under continuous contact. By learning a local interaction policy whose purpose is to produce locally correct contact-rich behavior, we find that M&M is effective at learning contact-rich

	RS-CerealMilk	RS-CanBread	RS-NutAssembly	K-Multistage-3	Multistage-4	K-Multistage-5
Stages	4	4	4	3	4	5
E2E	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
RAPS	0 / 0	0 / 0	0 / 0	.89 / .1	.3 / .15	0 / 0
SPA	0.73 / 0.05	.63 / .21	0.23 / .21	**1 / 0**	0 / 0	0 / 0
M&M	**.85 / .21**	**0.9 / 0.2**	**0.96 / 0.08**	**1 / 0**	**.89 / .13**	**.80 / .22**

Table 2: **Multistage (Long-horizon) results.** Performance is measured in terms of mean/std task success rate at convergence. M&M is the consistently solves each task, **outperforming SPA by over 70%** on challenging contact-rich tasks such as NutAssembly.

	$\sigma = 0$	$\sigma = 0.01$	$\sigma = 0.025$	$\sigma = 0.1$	$\sigma = 0.5$
SPA	1 / 0	.93 / .05	.27 / .12	0 / 0	0 / 0
M&M	1 / 0	1 / 0	1 / 0	0.75 / 0.07	0 / 0

Table 3: **Noisy Pose Ablation Results.** We add Gaussian noise to the pose estimates and evaluate SPA and M&M. M&M is able to handle noisy poses, only observing performance degradation beyond $\sigma = 0.1$.

manipulation over long horizons (Table 1, Table 2), outperforming SPA by a wide margin (98%, 97%, 96%, vs. 56%, 27%, 23%). Our decomposition into contact-free motion generation and contact-rich interaction decouples the *what* (target nut) and *where* (peg) from the *how* (precision grasp and contact-rich place), allowing the RL agent to simply focus on the aspect of the problem that is challenging to estimate a-priori: how to interact with the objects in the appropriate manner.

Ablation: M&M produces policies that are robust to noisy pose estimates. In this experiment we evaluate a core advantage of M&M: the ability to handle noisy/inaccurate poses by leveraging online interaction via RL. On the `RS-Can` task, we add zero-mean Gaussian noise to the pose, with $\sigma \in 0.01, 0.025, .1, .5$ and report our results in Table. 3. While SPA struggles to handle $\sigma > 0.01$, M&M is able to learn with noisy poses at $\sigma = .1$, at the cost of slower learning performance. Neither method performs well at $\sigma = 0.5$, however at that point the poses are not near the object and the effect is similar to a random robot pose reset.

Analysis: M&M leverages task plans to improve learning performance While the target object sequence is necessary for M&M to plan to the right location at the right time, in this experiment we evaluate if knowledge of the post-conditions is necessary. Specifically, on the `RS-Can` task, we evaluate the use of post-condition checks (M&M) to trigger the next step in the plan versus simply using a timeout of 25 steps. We find that it is in fact critical to use post-condition checks to enable the agent to effectively sequence the plan; use of a timeout results in dithering behavior which slows down learning. After 10K episodes we observe a performance improvement of 31% (100% with checks, 69% without) by including plan post-conditions in our pipeline.

6 Limitations and Future Work

In this work, we proposed M&M, a method that seamlessly integrates the long-horizon reasoning capabilities of classical planning and the dexterity of learned policies. At the heart of our method lies the decomposition of manipulation tasks into sequential phases of contact-free motion-generation and contact-rich interaction. We solve these phases using sampling-based motion planning and model-free RL respectively, an approach which we validate via an extensive experimental evaluation. However, our work has several limitations which leave scope for future work. First, we assume access to a coarse high-level plan, which could be addressed by introducing a task planner, e.g. STRIPS [55], and specifying explicit pre-/post-conditions for each learn-able action sequence for the local policy. Second, M&M is susceptible to errors in segmentation. One solution is to learn a residual policy over the predicted target for the motion planner using RL, to adapt the targets based on segmentation noise or failure. Finally, we note that M&M inherits the standard limitations of RL, namely dependence on reward definition, requirement of resets and concerns regarding safety. However, we argue that our method is highly amenable to sim2real transfer, which would bypass these issues, as M&M trains *local* visuomotor policies using only a wrist-camera, thereby reducing the sim2real gap. We leave this extension to future work.

References

[1] M. T. Mason. Toward robotic manipulation. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:1–28, 2018.

[2] R. P. Paul. *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. Richard Paul, 1981.

[3] D. E. Whitney. The mathematics of coordinated control of prosthetic arms and manipulators. 1972.

[4] M. Vukobratović and V. Potkonjak. *Dynamics of manipulation robots: theory and application*. Springer, 1982.

[5] T. Lozano-Perez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1):3–24, 1984.

[6] R. H. Taylor, M. T. Mason, and K. Y. Goldberg. Sensor-based manipulation planning as a game with nature. In *Fourth International Symposium on Robotics Research*, pages 421–429, 1987.

[7] A. T. Miller and P. K. Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004.

[8] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-P´erez. Integrated Task and Motion Planning. *Annual review of control, robotics, and autonomous systems*, 4, 2021.

[9] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1957–1964. IEEE, 2016.

[10] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. 2017.

[11] A. Mousavian, C. Eppner, and D. Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2901–2910, 2019.

[12] 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)*, pages 13438–13444. IEEE, 2021.

[13] A. J. Ijspeert, J. Nakanishi, and S. 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.

[14] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In *Springer Handbook of Robotics*, 2008.

[15] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2204.01691*, 2022.

[16] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[17] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[18] M. Dalal, A. Mandlekar, C. Garrett, A. Handa, R. Salakhutdinov, and D. Fox. Imitating task and motion planning with visuomotor transformers. 2023.

[19] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[20] A. Herzog*, K. Rao*, K. Hausman*, Y. Lu*, P. Wohlhart*, M. Yan, J. Lin, M. G. Arenas, T. Xiao, D. Kappler, D. Ho, J. Rettinghouse, Y. Chebotar, K.-H. Lee, K. Gopalakrishnan, R. Julian, A. Li, C. K. Fu, B. Wei, S. Ramesh, K. Holden, K. Kleiven, D. Rendleman, S. Kirmani, J. Bingham, J. Weisz, Y. Xu, W. Lu, M. Bennice, C. Fong, D. Do, J. Lam, N. Brown, M. Kalakrishnan, J. Ibarz, P. Pastor, and S. Levine. Deep rl at scale: Sorting waste in office buildings with a fleet of mobile manipulators. In *arXiv preprint arXiv:2305.03270*, 2023.

[21] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. *arXiv preprint arXiv:2210.13702*, 2022.

[22] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

[23] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[24] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal. Visual dexterity: In-hand dexterous manipulation from depth. *arXiv preprint arXiv:2211.11744*, 2022.

[25] Y. Labbé, L. Manuelli, A. Mousavian, S. Tyree, S. Birchfield, J. Tremblay, J. Carpentier, M. Aubry, D. Fox, and J. Sivic. Megapose: 6d pose estimation of novel objects via render & compare. *arXiv preprint arXiv:2212.06870*, 2022.

[26] 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.

[27] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. kpam: keypoint affordances for category-level robotic manipulation. arxiv. *arXiv preprint arXiv:1903.06684*, 2019.

[28] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg. Dex-net 3.0: Computing robust robot suction grasp targets in point clouds using a new analytic model and deep learning. *arXiv preprint arXiv:1709.06670*, 2017.

[29] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019.

[30] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[31] J. Ichnowski, Y. Avigal, J. Kerr, and K. Goldberg. Dex-nerf: Using a neural radiance field to grasp transparent objects. *arXiv preprint arXiv:2110.14217*, 2021.

[32] Q. Dai, Y. Zhu, Y. Geng, C. Ruan, J. Zhang, and H. Wang. Graspnerf: Multiview-based 6-dof grasp detection for transparent and specular objects using generalizable nerf. *arXiv preprint arXiv:2210.06575*, 2022.

[33] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[34] M. Laskin, A. Srinivas, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[35] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[36] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.

[37] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[38] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[39] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[40] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, pages 2226–2240. PMLR, 2023.

[41] M. A. Lee, C. Florensa, J. Tremblay, N. Ratliff, A. Garg, F. Ramos, and D. Fox. Guided uncertainty-aware policy optimization: Combining learning and model-based strategies for sample-efficient policy learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7505–7512. IEEE, 2020.

[42] J. Yamada, Y. Lee, G. Salhotra, K. Pertsch, M. Pflueger, G. Sukhatme, J. Lim, and P. Englert. Motion planner augmented reinforcement learning for robot manipulation in obstructed environments. In *Conference on Robot Learning*, pages 589–603. PMLR, 2021.

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

[44] F. Xia, C. Li, R. Martín-Martín, O. Litany, A. Toshev, and S. Savarese. Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation. *arXiv preprint arXiv:2008.07792*, 2020.

[45] S. James and A. J. Davison. Q-attention: Enabling efficient learning for vision-based robotic manipulation. *IEEE Robotics and Automation Letters*, 7(2):1612–1619, 2022.

[46] S. James, K. Wada, T. Laidlow, and A. J. Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13739–13748, 2022.

[47] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[48] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra. Detecting twenty-thousand classes using image-level supervision. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IX*, pages 350–368. Springer, 2022.

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

[50] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[51] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

[52] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[53] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

[54] M. Dalal, D. Pathak, and R. R. Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. *Advances in Neural Information Processing Systems*, 34: 21847–21859, 2021.

[55] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[56] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[57] O. 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.

Appendix

A M&M Implementation Details

Algorithm 1 M&M

Require: segmentation model $\mathcal{S}$, high-level plan $\mathcal{P}$, Motion Planner MP, RGB-D global camera, RGB wrist camera, Camera Matrix K^{global}

initialize RL: π_θ, replay buffer $\mathcal{R}$
for episode $1 \ldots N$ **do**
  for $p \in \mathcal{P}$ **do**
    ▷ Global Policy
    object, post-condition: $\mathcal{P}[p]$
    PC^{global} = Projection(O_1^{global}, O_2^{global}, K^{global})
    M_{robot}, M_{obj} = Segmentation(O_1^{global}, O_2^{global}, robot, object)
    PC^{robot} , PC^{object} = $M_{robot} * PC^{global}$, $M_{obj} * PC^{scene}$
    $PC^{scene} = PC^{global} - PC^{robot}$
    tgt = mean(PC^{obj})
    q_{target} = IK(tgt)
    MotionPlan(MP, q_{target}, PC^{scene})
    ▷ Local Policy
    for $i = 1, ..., h$ low-level steps **do**
      Get action $a_t \sim \pi_\theta(O_t^{local})$
      Get next state $O_{t+1}^{local} \sim p(|s_t, a_t)$.
      Store $(O_t^{local}, a_t, O_{t+1}^{local}, r)$ into $\mathcal{R}$
      Sample $(O_k^{local}, a_t, O_{k+1}^{local}, r) \sim \mathcal{R}$ ▷ k = random index
      update π_θ using RL
      if post-condition **then**
        break

As described in Section 3, we use post-conditions to determine when the Local Policy should hand control back to the Global Policy. For the tasks we consider, these post-conditions amount to checking for a grasp or placement for the target object in the stage. For example, for `RS-NutRound`, the plan for the first stage is (grasp, nut) and the plan for the second stage is (place, peg). Placements are straightforward to check: simply evaluate if the object being manipulated is within a small region near the target object. This can be computed using the estimated pose of the two objects (current and target). Grasps are more challenging to estimate and we employ a two stage pipeline to detecting a grasp. First, we estimate the object pose and then evaluate if the z value has increased from when the stage began. Second, in order to ensure the object is not simply tossed in the air, we check if the robot's gripper is tightly caging the object. We do so by collision checking the object pointcloud against the gripper mesh. We use the same collision checking procedure as outlined in Sec 3 for checking collision between the scene pointcloud and robot mesh.

Moving to training details, we implement the Local Policy using DRQ-v2 [33], a state-of-the-art visual model-free RL algorithm. For all tasks, we use the reward function defined by the environment, which may be dense or sparse depending on the task. We train the local policy only using the observations and actions produced by itself, as opposed to taking advantage of actions produced during motion-planning, as done in MoPA-RL [42]. We find that for M&M, it is crucial to use an action-repeat of 1, in general we found that increasing this harmed performance, in contrast to the end-to-end baseline which performs best with an action repeat of 2. For training policies using DRQ-v2, we use the default hyper-parameters from the paper, held constant across all tasks. We use the "medium" difficult exploration schedule defined in [33], which anneals the exploration σ from 1.0 to 0.1 over the course of 500K environment steps. Due to memory concerns, instead of using a replay buffer size of 1M, ours is of size 750K across each task. Finally, for path length, we use the standard benchmark path length for E2E and MoPA-RL, 5 per stage for RAPS following Dalal et al. [54], and 25 per stage for M&M. For an overall description of M&M, see Algorithm 1.

B Tasks

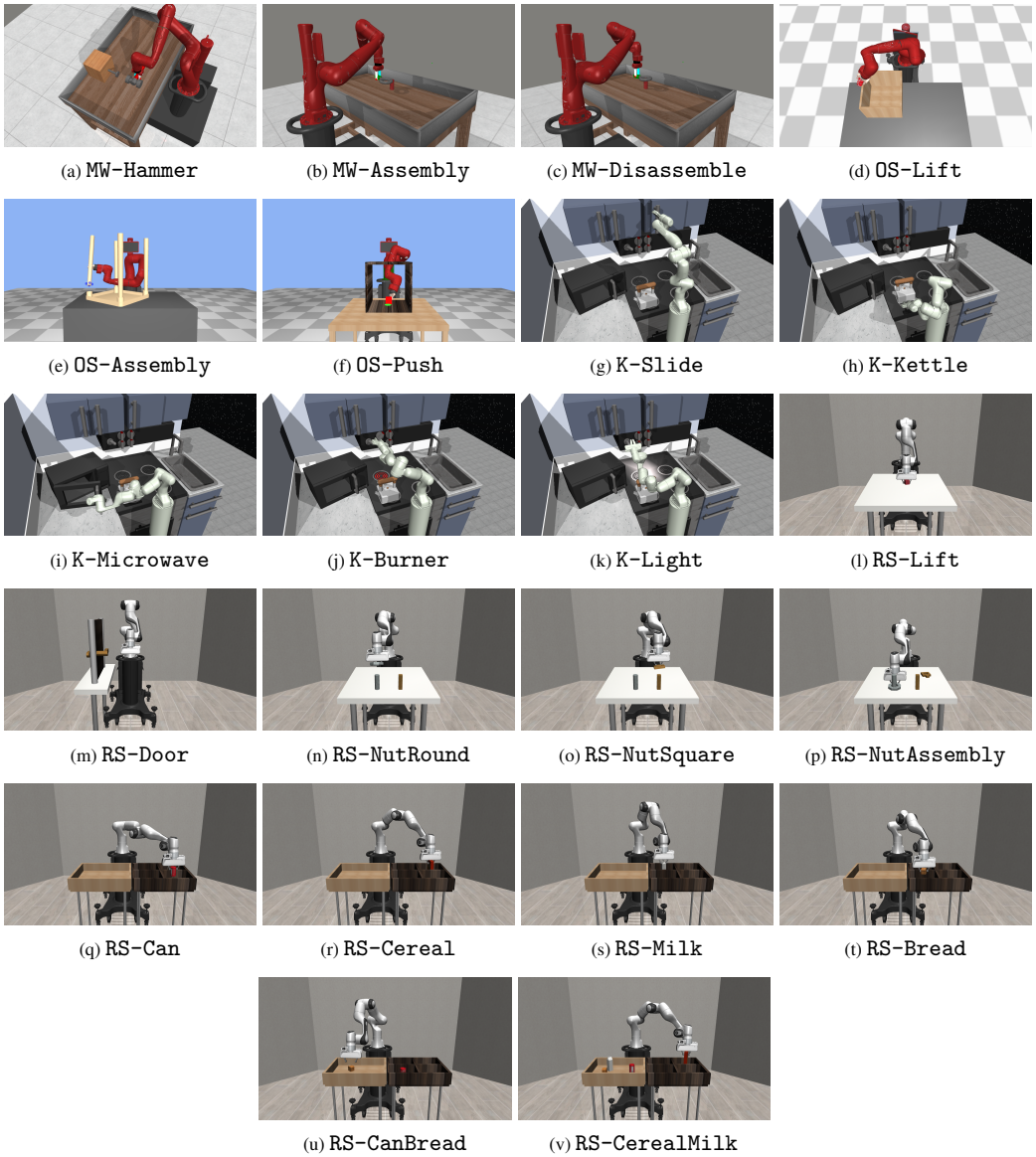

Figure B.1: **Task Visualizations**. M&M is able to solve all tasks with at least 80% success rate from purely visual input.

We discuss each of the environment suites that we evaluate using M&M. All environments are simulated using the MuJoCo simulator [56].

1. **Metaworld** (Row 1 of Fig. B.1). Metaworld, introduced by Yu et al. [50], aims to offer a standardized suite for multi-task and meta-learning methods. The benchmark consists of 50 separate manipulation tasks with a Sawyer robot, well-shaped reward functions, involve manipulating a single object to a randomized goal position, or multiple objects to a deterministic goal position. We evaluate on the single-task, multi-goal, v2 variants of the Metaworld environments. All environments use end-effector position control - a 3DOF arm action space along with gripper control - orientation is fixed. In our evaluation we use the default environment task rewards, a fixed camera view for the baselines and a wrist camera for our local policies. We refer the reader to the Metaworld paper for additional details regarding the environment suite.
2. **Obstructed Suite** (Rows 1-2 of Fig. B.1). The Obstructed Suite of tasks introduced by Yamada et al. [42] are a challenging set of tasks requiring a Sawyer arm to perform obstacle avoidance while solving the task. The `OS-Lift` task requires the agent to pick up a can that is inside a tall box, requiring it to reach over the walls to grab the object and then lift it without making contact with the edges of the bin. The `OS-Push` environment tasks the agent with push a block to the goal in the present of a bin that forces the agent to adjust its motion in order to avoid being blocked by its upper joints. Finally, the `OS-Assembly` task involves moving the robot arm to a precise placement location while avoiding obstacles, then performing the table leg placement. Note that we evaluate our method on these environments from visual input, a more challenging setting than the one considered by Yamada et al. [42].
3. **Kitchen** (Rows 2-3 of Fig. B.1). The Kitchen manipulation suite introduced in the Relay Policy Learning Paper [51] and maintained in D4RL [52] is a set of challenging, sparse reward, joint-controlled manipulation tasks in a single kitchen. The tasks require the ability to explore efficiently whilst also being able to chain skills across long temporal horizons, to achieve behaviors such as opening the microwave, moving the kettle, flicking the light switch, turning the burner, and finally sliding the cabinet door (`K-Multistage-5`). Aside from the single-stage tasks described in Section 4, we evaluate on three multi-stage tasks which require chaining the single-stage tasks in a particular order. `K-Multistage-3` involves moving the kettle, flicking the light switch and turning the burner, while `K-Multistage-4` is the same as `K-Multistage-3`, but the agent must first open the microwave door then execute the rest of the tasks.
4. **Robosuite** (Rows 3-6 of Fig. B.1). The Robosuite benchmark from Zhu et al. [53] contains challenging, long-horizon manipulation tasks involving pick-place and nut assembly, as well as simpler tasks that involve lifting a cube and opening a door. The rewards are coarsely defined in terms of distances to targets as well as grasp/placement conditions, which, in fact, are straightforward to implement in the real world as well using pose estimation. This stands in contrast to Metaworld which spends considerable engineering effort defining well-shaped dense rewards often by taking advantage of object geometry. As a result, learning-based methods struggle to make any progress on Robosuite tasks that involve more than a single-stage - optimizing the reward function tends to leave the agent a local minima. The suite also contains a well-tuned, realistic Operation Space Control [57] implementation that we leverage to train policies in end-effector space.