2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561221

Manipulation Planning Among Movable Obstacles Using Physics-Based Adaptive Motion Primitives

Dhruv Mauria Saxena*, Muhammad Suhail Saleem*, and Maxim Likhachev

***Abstract*— Robot manipulation in cluttered scenes often requires contact-rich interactions with objects. It can be more economical to interact via non-prehensile actions, for example, push through other objects to get to the desired grasp pose, instead of deliberate prehensile rearrangement of the scene. For each object in a scene, depending on its properties, the robot may or may not be allowed to make contact with, tilt, or topple it. To ensure that these constraints are satisfied during non-prehensile interactions, a planner can query a physics-based simulator to evaluate the complex multi-body interactions caused by robot actions. Unfortunately, it is infeasible to query the simulator for thousands of actions that need to be evaluated in a typical planning problem as each simulation is time-consuming. In this work, we show that (i) manipulation tasks (specifically pick-and-place style tasks from a tabletop or a refrigerator) can often be solved by restricting robot-object interactions to *adaptive motion primitives* in a plan, (ii) these actions can be incorporated as subgoals within a multi-heuristic search framework, and (iii) limiting interactions to these actions can help reduce the time spent querying the simulator during planning by up to $40\times$ in comparison to baseline algorithms. Our algorithm is evaluated in simulation and in the real-world on a PR2 robot using PyBullet as our physics-based simulator. Supplementary video: `https://youtu.be/ABQc7JbeJPM`.**

I. Introduction

Manipulation planning problems in domestic households, industrial manufacturing and warehouses require contact-rich interactions between a robot and the objects in the environment. As the amount of clutter in a scene increases, the likelihood of finding a completely collision-free trajectory for the manipulator decreases. This does not mean the task is impossible since we might still be able to complete it by moving the objects around. In these cases, non-prehensile interactions with objects can be much faster than deliberately rearranging the scene via a sequence of slow pick-and-place style prehensile maneuvers. In addition, each object in a cluttered scene is associated with constraints that define how a robot can manipulate it. For example, while we might be allowed to interact freely with a box of sugar, we might not be allowed to tilt or topple a cup of coffee.

We want to enable robots to grasp in clutter by using non-prehensile actions to interact with objects while satisfying any object-centric constraints. This domain of Manipulation Among Movable Obstacles (MAMO) [1] is derived from prior work on Navigation Among Movable Obstacles (NAMO) [2], [3]. Planning problems for NAMO aim to find a feasible path between start and goal states for a mobile robot navigating in an environment with reconfigurable obstacles[1]. We focus on the class of MAMO problems where the goal for the robot manipulator is a 6D pre-grasp pose of an object in $SE(3)$ without any constraints on the final poses of the movable objects.

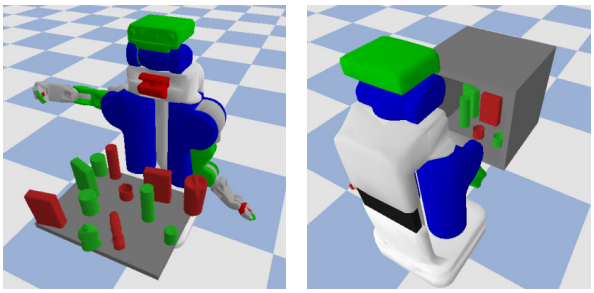

Fig. 1. Manipulation tasks in cluttered tabletop (left) or refrigerator (right) workspaces require planners to account for complex multi-body interactions between the robot and objects (green movable objects and red immovable obstacles). The goal is to pickup a randomly selected immovable obstacle.

Motion planning for MAMO is computationally costly because of two major challenges. First, we need to accurately model the dynamics of the robot-object and object-object interactions in the environment during planning. This requires the use of a high-fidelity physics-based simulator since hand-designed analytical models are hard to generalise for complex object geometries and cluttered scenes. The simulator is used to model the environment and predict the outcome of actions. The computational cost associated with running a simulator is high, which makes it infeasible to query the simulator for every action that needs to be evaluated. The second challenge is associated with the search space of the problem. Since the robot may interact with objects in the scene and reconfigure them, the search space needs to include the configuration space of all these objects. This makes the search space for a planning problem in this domain grow exponentially with the number of objects, and makes it computationally hard to find a solution.

In this work, we make an observation that many MAMO problems can be solved effectively by restricting robot-object interactions to adaptive motion primitives and show how this observation can be exploited to structure an efficient search for a contact-rich motion. *Adaptive motion primitives* (AMPs, Section IV-C) are long-range actions generated on-the-fly

The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. e-mail: {dsaxena, msaleem2, mlikhach}@andrew.cmu.edu

*Dhruv Mauria Saxena and Muhammad Suhail Saleem contributed equally to this work. This work was supported by the A2I2 program, contract W911NF-18-2-0218, funded by ARL.

[1]In our work 'objects' may be movable, but 'obstacles' are immovable.

such that they terminate in a valid goal state [4]. In our domain, they are straight lines in the configuration space of the robot, between two states whose end-effector Cartesian coordinates are within δ of each other in Euclidean norm.

We test our algorithm on random initialisations of the cluttered tabletop and refrigerator scenes from Fig. 1. Empirically our results in Section VI show that even with this restriction, our algorithm solves many MAMO planning problems and is up to $40\times$ faster than competitive baselines in our experiments.

Our main contributions include:

- a *two-stage planning approach* where we first parallely sample promising AMPs, and then systematically use them in our planning algorithm to find a solution.
- the use of these AMPs as *subgoals* within a multi-heuristic search algorithm.
- an action evaluation scheme that minimises the time spent querying a simulator during planning.

II. Related Work

The MAMO domain is closely tied to prior work in the field of NAMO [2], [3]. Past works in MAMO have taken one of two popular approaches - either solving problems via a sequence of pick-and-place style maneuvers [1], or limiting solutions to only planar robot-object interactions [5], [6], [7]. The rearrangement planning problem was extensively studied by King [8] in their thesis which focused on non-prehensile interactions.

In this work we use a physics-based simulator in-the-loop during planning to account for dynamics of robot-object and object-object interactions in the scene. Plaku et. al. [9] decompose the planning problem into a high-level discrete space, and a low-level sampling-based planner with a complex dynamics model (physics-based simulator). Zickler and Veloso [10] attempt to solve physics-based planning problems with the help of high-level, long-range robot behaviours. Dogar et. al. [11] simulate and cache multiple robot-object interactions, and use their result during planning to find feasible solutions. However, they do not allow any object-object interactions, which can be unavoidable in cluttered MAMO scenes. Similar to our work, the idea of planning till the proximity of the goal and using a more expensive specialized maneuver from within this proximity can be seen in [12] in the context of grasp planning.

Most relevant to our work in this paper are: a sampling-based algorithm KPIECE [13] and a search-based planning algorithm Selective Simulation [14]. Each uses different approaches to incorporate a simulator in-the-loop during planning. KPIECE, developed for applications with complex dynamics uses an importance function over discretized cells of the robot workspace to guide exploration. However, it queries the simulator for all action evaluations which proves to be computationally expensive for MAMO. Selective Simulation is an iterative approach which is based on the key idea that not all interactions are relevant for a given planning problem. It intelligently identifies these relevant interactions and queries the simulator to only evaluate them, thereby saving considerable computational effort. However, in the original paper, Selective Simulation was evaluated for simple constraints of contact or toppling off the table. In addition to these, we consider constraints on how far obstacles can be tilted and how much velocity can be imparted to them. Selective Simulation is also prone to repeated simulations of similar actions which is time-consuming and something we explicitly account for in our work with soft duplicate detection. Our experimental analysis also includes comparison with Selective Simulation.

III. Problem Formulation

A. Search Space

In this work, we denote the robot manipulator as $\mathcal{R}$, and $\mathcal{X}_\mathcal{R} \subset \mathbb{R}^q$ as the configuration space for a q degrees-of-freedom (dofs) manipulator. The set of objects in the scene is $\mathcal{O} = \{O_1, \ldots, O_n\}$, and the configuration of any object $\mathcal{X}_{O_i} \in SE(3)$ includes the 3D position and orientation. The robot is equipped with a set of actions $\mathcal{A}$. The search space for a planning problem in the MAMO domain is the Cartesian product of the robot and all object configuration spaces, denoted as $\mathcal{X} = \mathcal{X}_\mathcal{R} \times \mathcal{X}_{O_1} \times \cdots \times \mathcal{X}_{O_n}$.

B. Object Constraints

Each object O_i is associated with m_i constraint functions $k_i^j : \mathcal{X} \to \{0, 1\}, 1 \leq j \leq m_i$ which return 1 if constraint k_i^j is satisfied, 0 otherwise. For example, an *immovable* obstacle (an object that is not allowed to be interacted with, for example, a wall) will contain a constraint function which evaluates to 1 so long as neither the robot nor any other object makes contact with it. We denote the evaluation of all constraints of an object as $K_i(x) = \left[k_i^1(x), \ldots, k_i^{m_i}(x)\right]$. If all constraints are satisfied, $K_i(x) = \mathbf{1}_{m_i}$.

We say a state is valid if all constraints for all objects are satisfied at that state. Formally, state x is valid if $K_i(x) = \mathbf{1}_{m_i} \, \forall \, O_i \in \mathcal{O}, i \in \{1, \ldots, n\}$. We denote the space of valid states as $\mathcal{X}_V$.

C. Problem Statement

A planning problem in the MAMO domain can be defined as the tuple $\mathcal{P} = (\mathcal{X}, \mathcal{A}, x_S, \mathcal{X}_G, \mathcal{T}, c)$ where $x_S \in \mathcal{X}_V$ is the start state, $\mathcal{X}_G \subset \mathcal{X}, \mathcal{X}_G \cap \mathcal{X}_V \neq \emptyset$ is the set of goal configurations, $\mathcal{T} : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$ is a deterministic transition function, and $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a state transition cost-function. We assume $\mathcal{X}_G$ is defined using a desired end-effector pose in $SE(3)$ (a grasp location for an object), which leads to a set of possible manipulator configurations, some of which must be valid ($\mathcal{X}_G \cap \mathcal{X}_V \neq \emptyset$).

A path π of length T is a sequence of states $\{x_1, \ldots, x_T\}$ and has cost $C(\pi) = \sum_{i=1}^{T-1} c(x_i, x_{i+1})$ where $x_{i+1} = \mathcal{T}(x_i, a_i)$. The goal for a MAMO planning problem is to find a valid path from start to goal, i.e. a path made up of a sequence of valid states. Formally, we can write this as an

optimisation problem:

$$\text{find } \pi^* = \arg\min_{\pi} C(\pi)$$

$$\text{s.t. } x \in \mathcal{X}_V, \forall x \in \pi \qquad \text{(path of valid states)}$$

$$x_1 = x_S, x_T \in \mathcal{X}_G \qquad \text{(start, goal constraints)}$$

$$x_{i+1} = \mathcal{T}(x_i, a_i), a_i \in \mathcal{A}, \forall x_i, x_{i+1} \in \pi \qquad \text{(transition dynamics)}$$

IV. Approach

A. Graph Representation

We solve MAMO planning problems using a search-based planning algorithm in $\mathcal{X}$. Our graph representation contains two types of actions in $\mathcal{A}$ - *simple primitives* and *adaptive motion primitives* (AMPs). Each simple primitive changes one joint angle of a robot by a small amount. In comparison, an AMP is computed on the fly and can change all coordinates in $\mathcal{X}_\mathcal{R}$. A consequence of our core assumption (Section IV-D) is that for simple primitives $a_\text{s} \in \mathcal{A}$, a valid transition $x' = \mathcal{T}(x, a_\text{s})$ implies that the state x and the successor state x' only differ in the robot configuration (in $\mathcal{X}_\mathcal{R}$). For AMPs $a_\text{AMP} \in \mathcal{A}$, a valid transition $x' = \mathcal{T}(x, a_\text{AMP})$ can lead to differences in object configurations $(\mathcal{X}_{O_1} \times \cdots \times \mathcal{X}_{O_n})$ in addition to a difference in $\mathcal{X}_\mathcal{R}$.

B. Action Evaluation

Following the ideas outlined in [14], we decompose our action evaluation scheme into a relatively fast collision checking routine and a slower physics-based simulation.

The set of objects $\mathcal{O}$ in a scene can be separated into two subsets - *movable* objects $\mathcal{O}_M$ that the robot is allowed to interact with, and *immovable* obstacles $\mathcal{O}_I = \mathcal{O} \setminus \mathcal{O}_M$. We assume that this separation is known a priori.

Definition 1 (Phase 1 validity)**.** We say an action $a \in \mathcal{A}$ from state $x \in \mathcal{X}_V$ is *Phase 1 valid* if it does not make contact with any immovable obstacle $O \in \mathcal{O}_I$.

Definition 2 (Phase 2 validity)**.** We say an action $a \in \mathcal{A}$ from state $x \in \mathcal{X}_V$ is *Phase 2 valid* if it is Phase 1 valid and it does not result in any object constraint violations.

The fast collision checking routine is used to determine Phase 1 validity of an action. This can also determine Phase 2 validity if there is no collision with any object. In the case when an action is Phase 1 valid, but also collides with some movable object(s), Phase 2 validity can only be determined after simulating it. This is because we need to account for the complex multi-body interactions that might result upon executing the action. These interactions might violate object constraints due to a movable object-immovable obstacle contact, or the robot violating other movable object constraints such as tilting or toppling.

C. Adaptive Motion Primitives

Adaptive motion primitives (AMPs) are IK-based motion primitives that are generated on-the-fly as part of our algorithm [4] and included in $\mathcal{A}$. For any robot configuration $x \in \mathcal{X}_\mathcal{R}$, if the robot's 3D Cartesian end-effector pose is within $\delta = 0.2\text{m}$ of the goal end-effector pose $\mathcal{X}_G$, we generate an AMP that tries to connect x to $\mathcal{X}_G$. This is done by obtaining an IK solution $x_G \in \mathcal{X}_\mathcal{R}$ for $\mathcal{X}_G$, and linearly interpolating between x and x_G. The validity of an AMP (Phase 1 or Phase 2) is dependent on checking all interpolated states between x and x_G. We choose the value of δ based on basic domain knowledge like the size of our workspaces and obstacles therein. Fig. 2 shows our action evaluation strategy from Section IV-B for AMPs.

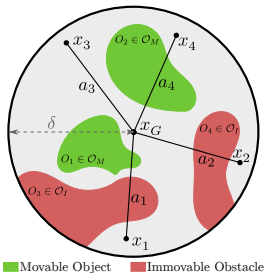

Fig. 2. For a particular goal state configuration $x_G \in \mathcal{X}_\mathcal{R}$, we can generate AMPs from several states x_i within distance δ from it. This δ-sphere in configuration space $\mathcal{X}_\mathcal{R}$ might be occupied by both movable (green) objects and immovable (red) obstacles. This can lead to invalid actions a_1, a_2, valid action a_3, and Phase 1 valid action a_4 whose Phase 2 validity will be determined after simulation.

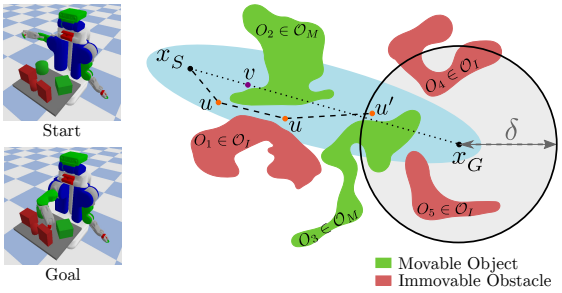

Fig. 3. Consider planning between start and goal configurations shown on the left. A "greedy" shortest-path search algorithm in the MAMO domain would proceed along the dotted path, exploring states in the light blue region, and spend a lot of time simulating interactions with object O_2 near the purple state v. Due to the assumptions we make, our search algorithm SPAMP proceeds along the dashed path via orange states u, and only starts simulating AMPs from states beyond u' in the δ-sphere around goal x_G.

D. Assumptions

We make one assumption for solving MAMO planning problems in this work which is closely related to AMPs and their use in our search algorithm.

Assumption 1. *We only need to simulate AMPs to find a valid solution for a MAMO planning problem.*

For grasping and reaching in cluttered scenes like Fig. 1, there is a large volume of object-free space between the start and goal. Interactions with objects are necessary when the robot is in a region with a high degree of clutter, and clutter near the goal is often the most pertinent for finding a feasible plan. Based on this observation we delay interacting with objects until the robot reaches a configuration near the goal.

For a preset value of δ, we restrict robot-object interactions until the end-effector is within δ of the goal pose. Since

AMPs are long-range actions contained inside this δ-sphere, interactions that are vital to the success of a plan are often the terminal AMPs in a plan (in comparison to the short-range simple primitives which do not lead to meaningful interactions). This leads us to Assumption 1. Consequently, since AMPs are terminal actions, the valid solution paths we find only contain a single action which interacts with obstacles. It is important to note that restricting interactions to a single action does not limit the number of objects the robot can interact with. We illustrate the effect of Assumption 1 in Fig. 3 by comparing our algorithm against a naive search algorithm.

This assumption restricts the space of MAMO planning problems solvable by our algorithm to those that require at most one AMP to interact with objects near the goal configuration. Our success rates from Section VI suggest that this assumption is not restrictive for the high-clutter scenes we consider in this work (Fig. 1). Assumption 1 helps us deal with the two major computational challenges for MAMO:

1) The number of simulator calls go down significantly.
2) The search space for planning from x_S to an AMP reduces from $\mathcal{X} = \mathcal{X}_\mathcal{R} \times \mathcal{X}_{O_1} \times \cdots \times \mathcal{X}_{O_n}$ to $\mathcal{X}_\mathcal{R}$.

E. Subgoals

We solve MAMO planning problems using a search-based planning algorithm. The performance of these algorithms is dependent on the quality of the heuristic functions used. The use of multiple heuristics can guide the search along multiple promising directions, which can help overcome local minima associated with any one heuristic.

Given the fact that simulations are the computational bottleneck in our domain, and the assumption that we only simulate AMPs, it is helpful to guide a search-based planning algorithm to regions of the search space where a valid AMP likely exists. Our two-stage planning approach discussed in Section V first finds AMPs that are Phase 1 or Phase 2 valid, and generates heuristic functions that guide the search to the beginning of these actions. For an AMP from state $x_{T-1} \in \mathcal{X}_V$, the corresponding heuristic function is a simple Euclidean distance from x_{T-1} in $\mathcal{X}_\mathcal{R}$. This first stage is run in parallel across multiple simulator instances, one per AMP sampled. The second stage runs a multi-heuristic search to find a path from start state x_S to a goal state in $\mathcal{X}_G$.

The beginning of the AMP x_{T-1} can be thought of as a *subgoal* for the planner since we guide the search towards it. While it is not necessary to reach the subgoal, exploring the search space near it can help find a solution. If an AMP a_{AMP} is Phase 2 valid, and the subgoal x_{T-1} is reachable without making contact with any object, we do not need to simulate actions on the way to x_{T-1}. It suffices to find a collision-free path in $\mathcal{X}_\mathcal{R}$ from start x_S to x_{T-1}, and append $x_T = \mathcal{T}(x_{T-1}, a_{\mathsf{AMP}})$ for a valid MAMO solution.

F. Soft Duplicate Detection for Action Evaluation

Assumption 1 states that we only simulate Phase 1 valid AMPs during planning. However, since the number of such AMPs in cluttered environments can be very large, we further optimise the calls to the simulator by employing a soft duplicate action detection scheme [15] to avoid simulating actions likely to violate constraints [16].

We maintain a list of states L from which an AMP has been simulated and found to be invalid. For any new state $x \in \mathcal{X}_V$ from which we find a Phase 1 valid AMP, we first compute its Euclidean distances in $\mathcal{X}_\mathcal{R}$ to states in L. Given a preset threshold β, if $\|x - x_l\|_2 < \beta$ for any $x_l \in L$ we postpone the simulation of the AMP from x by artificially inflating the heuristic value of x and re-inserting it into the priority queue it was expanded from.

Algorithm 1 Simulation-based Planning with AMPs (SPAMP)

Input: Planning problem $\mathcal{P}$, number of AMP subgoals N, number of AMP samples M, planning timeout $t_{\max}$
Output: solution path π

1: **procedure** SPAMP($\mathcal{P}, N, M, t_{\max}$)
2: $H \leftarrow$ `GetValidSubgoals`(N, M)
3: ▷ Phase 1 or Phase 2 valid subgoals.
4: $OPEN \leftarrow$ `InitialiseHeuristics`(H)
5: $\pi \leftarrow$ PLAN$(\mathcal{P}, H, OPEN, t_{\max})$
6: **return** π

V. ALGORITHM

Algorithm 1 is a high-level overview of our two-stage planning pipeline. We call our algorithm Simulation-based Planning with AMPs (SPAMP). The GETVALIDSUBGOALS subroutine in Line 2 selects subgoals via rejection sampling. M Phase 1 valid AMPs are randomly sampled and simulated in parallel. N Phase 2 valid subgoals are returned (if available), else a combination of Phase 1 and Phase 2 valid make up the N returned subgoals (first stage).

A. Multi-Heuristic Framework for MAMO

We use Multi-Heuristic A* (MHA*) [17] as our search algorithm in this work. MHA* with its round robin priority queue selection was originally developed under the assumption that all action evaluations take roughly the same amount of time. However, our action evaluations have varying time complexity (collision-checking for Phase 1 vs. simulating for Phase 2 validity). A simple round robin strategy would lead to an uneven distribution of computational resources across the queues. For this reason, we prioritise state expansions from queues that the search has spent the least time expanding states from thus far. This time-based prioritisation of queues in MHA* leads to a much more equitable allocation of computational resources for MAMO.

B. Planning Algorithm

Algorithm 2 contains details from the preceding sections to provide a more in-depth look at our planning algorithm. $OPEN$ is a set of priority queues, each of which is defined by a heuristic function. Following standard MHA* terminology, our *anchor* heuristic is a 3D Breadth-First Search (BFS) heuristic computed from the specified Cartesian goal end-effector position [18] taking into account the immovable obstacles in the scene. Every other heuristic is defined by a

Algorithm 2 SPAMP PLANNER

```
1: procedure PLAN(P, H, OPEN, t_max)
2:     Insert(OPEN, x_S)                          ▷ Add to all queues.
3:     while OPEN is not empty do
4:         h ← BestQueue(OPEN)                    ▷ Time-based selection.
5:         x ← BestState(h)                       ▷ Pop from all queues.
6:         if x ∈ X_G then
7:             return ExtractPath(x)
8:         if ∃ Phase 2 valid AMP from x ∈ H then
9:             a_AMP ← Phase 2 valid AMP from x ∈ H
10:            return ExtractPath(x) ∪ {T(x, a_AMP)}
11:        EXPAND(x, OPEN)
12: procedure EXPAND(x, OPEN)
13:    for a_s ∈ A do                             ▷ Simple primitives only.
14:        x' ← T(x, a).
15:        if x' ∈ X_V then
16:            Insert(OPEN, x')
17:    if ||FK(x) − X_G||_2 ≤ δ then              ▷ 3D end-effector poses.
18:        if x has soft duplicate then
19:            InflateHeuristic(x)
20:            Insert(OPEN, x)
21:            return
22:        x_G ← IK(X_G)                          ▷ Inverse kinematics.
23:        a_AMP ← GenerateAMP(x, x_G)
24:        if IsPhase1Valid(a_AMP) then
25:            if IsInteraction(a_AMP) then
26:                x' ← Simulate(a_AMP)
27:                if x' ∈ X_V then
28:                    Insert(OPEN, x')
29:                else
30:                    Insert(L, x)
31:            else
32:                x' ← T(x, a_AMP).
33:                Insert(OPEN, x')
```

corresponding AMP subgoal as per Section IV-E. The action set $\mathcal{A}$ is made up of the simple primitives and AMPs via the function `GenerateAMP`.

SPAMP terminates if the next-best state to expand x is in the goal set or we have already found a Phase 2 valid AMP from it. For every other x, the EXPAND function generates and evaluates all possible successor states of x. These necessarily include successors $x' = \mathcal{T}(x, a_s) \, \forall \, a_s \in \mathcal{A}$ (Line 13). In addition, we may generate and evaluate an AMP from x to $x_G \in \mathcal{X}_\mathcal{R}$, an inverse-kinematics solution for $\mathcal{X}_G$ (Line 22). This evaluation (Line 23 onwards) occurs if x passes the end-effector distance check (Line 17) and soft duplicate check (Line 18). The `IsInteraction` function returns true if a_{AMP} 'collides' with a movable object during the Phase 1 validity check (Line 24).

VI. EXPERIMENTAL RESULTS

We run all our experiments on the PR2 robot and use PyBullet [19] as our physics-based simulator. We run experiments in two different workspaces - a tabletop and a refrigerator. The objects in a scene are divided into immovable and movable subsets prior to planning. The robot is allowed to interact with the movable objects but cannot tilt them excessively, cause them to fall over or outside the workspace, or impart high velocities. Neither the robot nor any movable object can make contact with immovable obstacles.

A. Comparative Quantitative Evaluation in Simulation

We compare the performance of our algorithm Simulation-based Planning with AMPs (SPAMP) against relevant state-of-the-art baseline algorithms that can be applied to the MAMO domain - KPIECE [13] and Selective Simulation (SS) [14]. Three variants of KPIECE were tested. The first two use the KPIECE implementation from OMPL [20], but differ in how goal biasing is implemented. K1 precomputes a set of 3 valid goal configurations by running IK before planning. K2 runs IK online (with random seeds) every time the search tree is grown towards the goal. K3 is our implementation of KPIECE[2]. The projective space for all KPIECE algorithms is the 3D Cartesian coordinate for the end-effector. SS2 is a modified version of Selective Simulation (SS2) with soft duplicate detection.

A planning problem is initialised with objects selected at random from the YCB Object Dataset [21], placed at random poses in the robot workspace. The goal for the PR2 is to reach a pre-grasp pose for an immovable obstacle. We run all planners on 180 randomly initialised planning problems in both workspaces. Table I shows quantitative results for all planners. We use 12 objects on the tabletop (6 movable and 6 immovable), and 5 objects in the refrigerator setting (3 movable and 2 immovable)[3]. Sample initialisations of these workspaces are shown in Fig. 1. SPAMP uses $N = 3$ subgoals from $M = 8$ samples in GETVALIDSUBGOALS. All planners were given a maximum planning time of 1800s. We divide all planning problems into two scenarios based on how long it takes a NAIVE planner (a vanilla MHA* algorithm with only the 3D BFS heuristic to the goal) to solve them. Problems solved by NAIVE in less than 100s are 'Easy', and the rest are 'Difficult'.

SPAMP achieves the highest success rate across all algorithms which shows that our assumption from Section IV-D is not too restrictive for the MAMO domain. In terms of planning times, SPAMP is $30 - 50\times$ faster than KPIECE and KPIECE2, and $2 - 8\times$ faster than SELSIM and SELSIM2. SELSIM is most competitive in terms of planning times, but it is still $2 - 7\times$ slower than SPAMP for difficult problems. KPIECE mostly performs random point-to-point exploration which is not useful for MAMO problems, and by default simulates all actions in the search tree which makes planning times grow quickly.

B. Runs on a Physical Robot

We setup the tabletop workspace experiment with the PR2 robot in our laboratory (Fig. 4). We selected 6 objects at random to initialise our scene and used PERCH 2.0 [22] to detect the object poses for simulator initialisation. SPAMP was given a 30s planning timeout, and objects were instantiated in the simulator as movable with 75% probability.

[2]We implement KPIECE as per the algorithm in [13], along with the recommended multiple levels of discretisation

[3]The tabletop is $0.6\text{m} \times 0.8\text{m}$, and refrigerator is $0.6\text{m} \times 0.6\text{m} \times 0.6\text{m}$.

TABLE I

SIMULATION STUDY OF VARIOUS PLANNERS IN THE MAMO DOMAIN.

		Planning Algorithms											
		Tabletop Workspace (6 movable, 6 immovable)						Refrigerator Workspace (3 movable, 2 immovable)					
Metrics	**Scenario**	**SPAMP**	K1	K2	K3	SS	SS2	**SPAMP**	K1	K2	K3	SS	SS2
Success %	Overall	**99%**	91%	92%	85%	87%	91%	93%	51%	38%	**94%**	87%	91%
Planning Time (s)	Easy	**5 ± 5**	267 ± 301	339 ± 351	211 ± 369	6 ± 20	8 ± 26	**3 ± 3**	116 ± 293	210 ± 443	167 ± 222	7 ± 22	14 ± 57
	Difficult	**11 ± 16**	318 ± 310	464 ± 416	210 ± 327	22 ± 41	38 ± 55	**8 ± 13**	58 ± 119	128 ± 335	249 ± 425	57 ± 126	63 ± 157
Simulation Time (s)	Easy	**0 ± 0**	267 ± 301	339 ± 351	42 ± 88	4 ± 18	4 ± 14	**0 ± 0**	116 ± 293	208 ± 441	50 ± 89	3 ± 18	8 ± 35
	Difficult	**1 ± 7**	318 ± 310	463 ± 415	90 ± 131	4 ± 14	16 ± 31	**1 ± 6**	57 ± 118	137 ± 334	44 ± 92	20 ± 82	24 ± 62

Fig. 4. MAMO experiment setup for a PR2 robot.

TABLE II

QUANTITATIVE PERFORMANCE FOR REAL-WORLD EXPERIMENTS

	Metrics		
Algorithm	Success Rate	Planning Time (s)	Simulation Time (s)
SPAMP	82%	2 ± 4	0.8 ± 0.6

The quantitative data from execution of 39 plans on the physical robot is shown in Table II. An obstacle constraint in the real-world was violated in 7 out of 39 executions. This is due to a mismatch between the simulator and the real-world and due to either object modeling errors, robot execution errors, or perception errors in object localisation.

C. In-Depth Analysis of SPAMP in Simulation

To get a better understanding of the quantitative performance of SPAMP, we conducted experiments to highlight the effect of various components. For our first experiment, we highlight the effect of subgoals and soft duplicate detection. We consider four different planning algorithms - NAIVE is a vanilla MHA* algorithm with only the 3D BFS heuristic to the goal; NAIVE+DD uses soft duplicate detection on top of NAIVE; SUBG uses one randomly sampled Phase 1 valid AMP as a subgoal in MHA*; and SUBG+DD uses soft duplicate detection on top of the SUBG planner. Problems in this experiment are initialised with 8 movable objects on the tabletop. Table III shows the quantitative benefits of subgoals and soft duplicate detection individually, and that in tandem they can greatly improve performance over the NAIVE planner.

TABLE III

EFFECT OF SUBGOALS AND SOFT DUPLICATE DETECTION

Metrics	**Scenario**	**Planning Algorithms**			
		NAIVE	NAIVE+DD	SUBG	SUBG+DD
Success Rate	Overall	90%	92%	95%	**96%**
Planning Time (s)	Easy	13 ± 22	9 ± 21	7 ± 14	**6 ± 10**
	Difficult	433 ± 388	188 ± 264	58 ± 108	**39 ± 97**

TABLE IV

QUANTITAVE PERFORMANCE OF SPAMP VARIANTS (TABLETOP)

Metrics	**Scenario**	**Planning Algorithms**			
		NAIVE	NAIVE+DD	PHASE1	**SPAMP**
Success Rate	Overall	85%	92%	97%	**99%**
Planning Time (s)	Easy	16 ± 22	14 ± 37	13 ± 101	**5 ± 5**
	Difficult	524 ± 439	379 ± 461	48 ± 218	**11 ± 16**
Simulation Time (s)	Easy	7 ± 13	4 ± 7	5 ± 11	**0 ± 0**
	Difficult	130 ± 292	63 ± 136	18 ± 62	**1 ± 7**

In a second experiment, we compare the performance of SPAMP against three related variants on the tabletop workspace experiment from Section VI-A. We compare against NAIVE, NAIVE+DD, and a PHASE1 planner, which randomly samples $N = 3$ Phase 1 valid AMPs for use as subgoals (without simulating them). All three baselines are allowed to simulate AMPs from within the δ-sphere of a goal configuration from the outset. Table IV shows that while simply using Phase 1 valid subgoals has benefits over not using them, the necessary reliance on simulating all Phase 1 valid AMPs leads to higher simulation times, and thereby higher planning times, as compared to SPAMP.

VII. CONCLUSION

In this work we present SPAMP, an algorithm for Simulation-based Planning with Adaptive Motion Primitives for the MAMO domain. We use AMPs as subgoals within a multi-heuristic search framework to solve manipulation planning problems in cluttered scenes. SPAMP improves planning times by up to $40 - 80\times$ over KPIECE, and up to $2 - 8\times$ over Selective Simulation, two state-of-the-art baselines for the MAMO domain. We show that our assumption of restricting robot-object interactions to terminal AMPs in a plan is not restrictive since we solve $93 - 99\%$ of all problems.

REFERENCES

[1] M. Stilman, J. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *2007 IEEE International Conference on Robotics and Automation, ICRA*. IEEE, 2007.

[2] G. T. Wilfong, "Motion planning in the presence of movable obstacles," *Ann. Math. Artif. Intell.*, 1991.

[3] M. Stilman and J. J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," *Int. J. Humanoid Robotics*, 2005.

[4] B. J. Cohen, G. Subramania, S. Chitta, and M. Likhachev, "Planning for manipulation with adaptive motion primitives," in *IEEE International Conference on Robotics and Automation, ICRA 2011*. IEEE, 2011.

[5] J. P. van den Berg, M. Stilman, J. Kuffner, M. C. Lin, and D. Manocha, "Path planning among movable obstacles: A probabilistically complete approach," in *Eighth International Workshop on the Algorithmic Foundations of Robotics, WAFR 2008*.

[6] M. R. Dogar and S. S. Srinivasa, "A planning framework for non-prehensile manipulation under clutter and uncertainty," *Auton. Robots*, 2012.

[7] J. E. King, M. Cognetti, and S. S. Srinivasa, "Rearrangement planning using object-centric and robot-centric action spaces," in *2016 IEEE International Conference on Robotics and Automation, ICRA*. IEEE, 2016.

[8] J. E. King, "Robust rearrangement planning using nonprehensile interaction," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, December 2016.

[9] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Trans. Robotics*, 2010.

[10] S. Zickler and M. M. Veloso, "Efficient physics-based planning: sampling search via non-deterministic tactics and skills," in *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*. IFAAMAS, 2009.

[11] M. R. Dogar, K. Hsiao, M. T. Ciocarlie, and S. S. Srinivasa, "Physics-based grasp planning through clutter," in *Robotics: Science and Systems VIII, 2012*, 2012.

[12] N. Vahrenkamp, M. Do, T. Asfour, and R. Dillmann, "Integrated grasp and motion planning," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2883–2888.

[13] I. A. Sucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Trans. Robotics*, 2012.

[14] M. Suhail Saleem and M. Likhachev, "Planning with selective physics-based simulation for manipulation among movable objects," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[15] W. Du, S. Kim, O. Salzman, and M. Likhachev, "Escaping local minima in search-based planning using soft duplicate detection," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*. IEEE, 2019.

[16] J. D. Hernández, M. Moll, and L. E. Kavraki, "Lazy evaluation of goal specifications guided by motion planning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 944–950.

[17] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic a," *The International Journal of Robotics Research*, 2016.

[18] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *IEEE International Conference on Robotics and Automation, ICRA 2010*. IEEE, 2010.

[19] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[20] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.

[21] B. Çalli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-cmu-berkeley dataset for robotic manipulation research," *Int. J. Robotics Res.*

[22] A. Agarwal, Y. Han, and M. Likhachev, "Perch 2.0 : Fast and accurate gpu-based perception via search for object pose estimation," in *IROS*, 2020.