Appears in IEEE International Conference on Robotics and Automation (ICRA) 2023

Object Reconfiguration with Simulation-Derived Feasible Actions

Yiyuan Lee, Wil Thomason, Zachary Kingston, Lydia E. Kavraki

***Abstract*— 3D object reconfiguration encompasses common robot manipulation tasks in which a set of objects must be moved through a series of physically feasible state changes into a desired final configuration. Object reconfiguration is challenging to solve in general, as it requires efficient reasoning about environment physics that determine action validity. This information is typically manually encoded in an explicit transition system. Constructing these explicit encodings is tedious and error-prone, and is often a bottleneck for planner use. In this work, we explore embedding a physics simulator within a motion planner to implicitly discover and specify the valid actions from any state, removing the need for manual specification of action semantics. Our experiments demonstrate that the resulting simulation-based planner can effectively produce physically valid rearrangement trajectories for a range of 3D object reconfiguration problems without requiring more than an environment description and start and goal arrangements.**

I. Introduction

Robot manipulation planning is primarily a problem of finding a sequence of *valid* actions that move a set of target objects to a given goal configuration. Actions are valid if they respect the problem's constraints, which may be task-specific (e.g., keeping a glass of water upright or maintaining dynamic stability of a stack of objects) or implicit and arising from the environment (e.g., avoiding collisions).

Most approaches to manipulation planning (e.g., [1–7]) rely on explicitly specified problem constraints through formal languages like Linear Temporal Logic (LTL) [8] and the Planning Domain Definition Language (PDDL) [9, 10] or through natural language [11]. These manually created specifications identify both the valid, meaningful subsets of the state space and the valid transitions between these subsets. The resulting transition system guides a search for a sequence of valid actions that perform the given task when executed by the robot.

However, such specifications are onerous and error-prone to construct [12], and may not capture the full set of possible actions. They must not only define the valid dynamics for a problem's environment, but also be rich enough to describe a wide range of problems and goals. Furthermore, problem specifications are not unique and the choice of specification can impact planning performance [13–16]. Conversely, some manipulation planners forgo full generality for simplified problem specification and improved performance [1, 2]. These planners tend to be restricted to planar tabletop object rearrangement or similar problems [17].

Department of Computer Science, Rice University, Houston, TX, 77005, USA, {yiyuan.lee, wbthomason, zak, kavraki}@rice.edu. This work was supported in part by NSF RI #2008720 and Rice University Funds, as well as NSF Grant #2127309 to the Computing Research Association for the CIFellows Project.

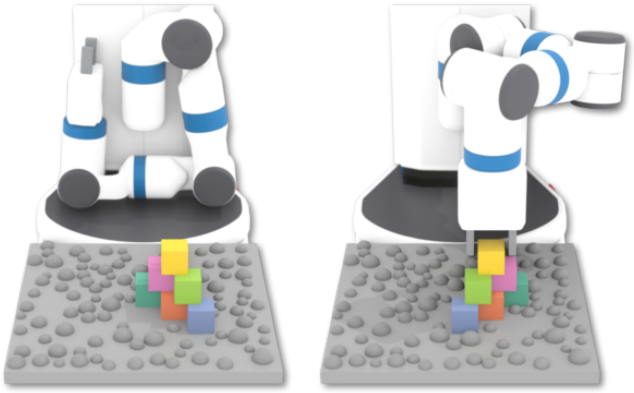


Fig. 1. To rotate the pyramid, the robot must reason about the physical validity of sequences of manipulation actions. For example, removing cubes from the bottom of the pyramid before the cubes at the top will cause the structure to topple. Placing a cube at certain positions on the bumpy surface will affect the ability to transfer the other cubes. Explicitly specifying potential intermediate arrangements and action validity for this setting is tedious. Our approach leverages a physics simulator to discover the valid actions for a given arrangement and plan to reconfigure the objects.

We propose a middle ground: planners that can solve a broad set of classes of manipulation planning problems with no more problem specification than a typical low-level motion planning problem. Our insight is that the necessary transition systems can be *implicitly defined* through an environment simulator, reducing the manual specification burden.

This paper contributes a novel perspective on manipulation problem specification and manipulation planner design. This perspective centers around embedding an environment simulation in a sampling-based planning algorithm as an implicit specification of a problem's valid transition system. In support of these ideas, we contribute (1) the *arrangement space*, a novel planning space representing object arrangements and dynamically discovered low-level robot motions moving between them, (2) *Stable Arrangement Search Trees* (SAST), an arrangement-space planner using embedded environment simulators to discover valid action sequences and the associated low-level motions (Sec. IV-A), and (3) a procedure to simplify the solutions found in the arrangement space. Concretely, we investigate the use of an embedded off-the-shelf physics simulator [18, 19] in SAST to efficiently find statically stable, collision-free states for 3D object reconfiguration problems without manually specifying action semantics. This setting is a specific instance of the broader manipulation planning paradigm that we propose.

We demonstrate that our proposed framework can efficiently solve 3D object reconfiguration problems with phys-

ical constraints without requiring more than an environment description and start/goal configurations to specify a problem. These results argue for the viability of a family of planners based upon implicitly simulator-defined transition systems.

II. Background and Related Work

This paper proposes a novel perspective on planning with embedded simulators that combines and extends earlier uses of simulation in robot planning and control (Sec. II-A). `SAST`, an example of this perspective in practice, is an efficient sampling-based planning algorithm (Sec. II-B) that builds upon ideas from tabletop rearrangement planning and integrated task and motion planning (Sec. II-C) to solve dynamically-constrained object reconfiguration problems.

A. Simulation in robotics

Simulation is widely used in robot control and learning. Model-predictive control (MPC) simulates control trajectories forward through time to choose optimal inputs [20]. Recent work improves MPC performance by integrating differentiable physics simulation [21, 22]. Efficient simulation for training [23, 24] has been core to learning-based methods for robot control [25–27], despite the challenge of translating controllers from simulation to the real world [28].

However, as noted by [29], simulation for planning is understudied. Prior work combining manipulation planning and simulation restricts to specific motion primitives [30, 31] or 2D settings [32]; we operate in a 3D workspace and dynamically discover the valid motions available to the robot via a bi-level search over object arrangements and robot motions. [29] studied efficient planning with simulators by selectively simulating actions. [33] improved the long-horizon planning efficiency of a Monte-Carlo Tree Search-based planner by integrating parallel simulation. [34, 35] use simulators with different precisions and interleaved simulation and execution to improve manipulation planning performance and robustness. These approaches complement `SAST`. We propose that an embedded simulator can be an effective implicit specification of a problem's constraints.

B. Sampling-based motion planning

Sampling-based motion planning (SBMP) is a family of efficient robot motion planning techniques based upon constructing approximations of a robot's high-dimensional configuration space [36] from sampled configurations [37–40]. Most SBMP algorithms operate by building up a graph [37] or tree [38] of valid configurations connected by short valid motions. `RRTConnect` [39] is among the fastest SBMP algorithms for many problems, due to its technique of growing two trees of valid motions, one from each of the start and the goal, toward each other using a local "extension" planner to control the trees' growth. `SAST` adapts the high-level planning loop of `RRTConnect` to search an expansive space of stable object arrangements (Sec. IV-A) using a simulation-based extension planner (Sec. IV-C).

C. Object reconfiguration

Object reconfiguration has been studied in contexts including manipulation planning [4, 5, 41, 42], rearrangement planning [1, 2, 43, 44], and integrated task and motion planning (TAMP) [10]. These approaches span an axis ranging from problem specialization (i.e., planar rearrangement planners [1, 2, 43]) to relative generality (i.e., full TAMP solving [3, 6, 7]). This axis also corresponds to the relative *specification effort* for each planner: a measure of the work a user must do to provide a given planner with the information it needs to operate. Planar rearrangement planners typically only specify the desired object arrangement (as well as the environment geometry), and exploit their assumption of planar problems to find solutions faster. TAMP solvers also rely on symbolic action specifications, mechanisms for discovering states that satisfy action preconditions, and more (e.g., explicit problem constraint specifications) [10]. We strike a balance: simulators still require manual effort to create, but are more broadly reusable across problems and domains than the specifications and samplers required by most TAMP solvers. Simulators can also implicitly encode a more general set of constraints than most rearrangement solvers, allowing for richer problems. Further, as progress in learning problem-specific dynamics models advances [45–48], the effort required to create simulators for planning will decrease.

`SAST`, like [2], relies on an arrangement-aware extension primitive to find valid action sequences. [32] also proposes a rearrangement planner incorporating a simplified 2D physics model to evaluate a predefined set of rearrangement actions. Similarly, [49] explores kinodynamic planning for planar rearrangement with a focus on reacting to unexpected events during rearrangement plan execution, and using a heuristic-based task specification. `SAST` uses full 3D physics, does not predefine motion primitives, and models dynamic constraints such as stability. In future work, synergistically combining `SAST` with the techniques of [32, 49] could allow `SAST` to use richer non-prehensile motions for manipulating objects.

III. Problem Formulation

We demonstrate implicit constraint definition via embedded simulation in a specific application: 3D object reconfiguration with stability constraints, using pick-and-place actions.

Consider a 3D workspace containing movable rigid-body *objects*, $o \in \mathcal{O}$, and a known set of posed static *obstacle* geometries. Objects have known 3D geometries and poses in $\mathrm{SE}(3)$. An *arrangement* assigns a pose to each object:

Definition 1: *Arrangement*
An arrangement, α, prescribes a pose, $\alpha[o] \in \mathrm{SE}(3)$, to each object in the workspace. Denote the *arrangement space*, the set of all arrangements, as $\mathcal{A}$, and let $\alpha \setminus o$ be arrangement α with object $o \in \mathcal{O}$ removed from consideration.

Arrangements may be *valid* or *invalid*. Valid arrangements are those that are both *collision-free* and *statically stable*.

Definition 2: *Valid arrangement*
Let $\texttt{CollisionFree}(\cdot)$ be a collision test for arrangements, such that $\texttt{CollisionFree}(\alpha) = \texttt{True}$ if α has

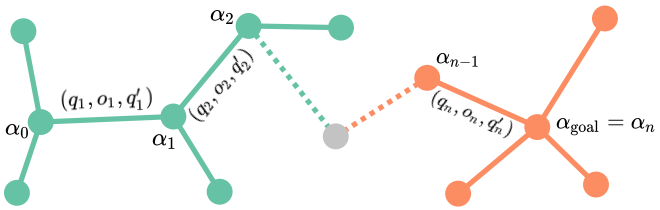


Fig. 2. Bidirectional search trees in the arrangement space. Each vertex represents a valid arrangement (Def. 2). An edge (q_i, o_i, q_i') represents a transformation between the two connected arrangements α_{i-1} and α_i. This comprises of a TRANSIT motion of the robot to configuration q_i, followed by a stable TRANSFER motion of object o_i from its pose in α_{i-1} to α_i by grasping o_i and moving the robot from q_i to q_i'. Edges are bidirectional—one can also transform arrangement α_i to α_{i-1} using a TRANSIT motion to q_i', followed by a stable TRANSFER motion of object o_i from its pose in α_i to its pose in α_{i-1} by grasping o_i and moving the robot from q_i' to q_i.

Algorithm 1: SAST

$\mathcal{T}_a \leftarrow \texttt{InitTree}(\alpha_0)$
$\mathcal{T}_b \leftarrow \texttt{InitTree}(\alpha_{\text{goal}})$
while True **do**
  $\alpha_{\text{rand}} \leftarrow \texttt{SampleArrangement}()$
  $\alpha_{\text{nearest}} \leftarrow \texttt{GetNearest}(\mathcal{T}_a, \alpha_{\text{rand}})$
  $\alpha_{\text{new}} \leftarrow \texttt{Extend}(\mathcal{T}_a, \alpha_{\text{nearest}}, \alpha_{\text{rand}}, \texttt{False})$
  if $\alpha_{\text{new}} \neq \alpha_{\text{nearest}}$ **then**
    if $\texttt{Connect}(\mathcal{T}_b, \alpha_{\text{new}}) = \alpha_{\text{new}}$ **then**
      return $\texttt{Path}(\mathcal{T}_a, \mathcal{T}_b)$
  $\texttt{Swap}(\mathcal{T}_a, \mathcal{T}_b)$

no objects in collision. Similarly, let $\texttt{Stable}(\alpha)$ be a static stability test for arrangements, such that $\texttt{Stable}(\alpha) = \texttt{True}$ if α is statically stable after a fixed duration.

An arrangement, $\alpha \in \mathcal{A}$, is valid if and only if $\texttt{CollisionFree}(\alpha) = \texttt{True}$ and $\texttt{Stable}(\alpha) = \texttt{True}$. We evaluate $\texttt{CollisionFree}(\cdot)$ via a physics simulator's collision checker. We check $\texttt{Stable}(\cdot)$ by stepping the simulator for a fixed number of time steps and verifying that all objects' displacements remain below a small heuristic threshold.

Let r be a robot arm with a static base and joint configuration space $\mathcal{Q}$ [36]. The arm is capable of two classes of motion: *Transit* motions move the empty end effector along a collision-free path between two workspace poses. *Transfer* motions grasp a target object and move it and the end effector to a new pose along a collision-free path [4].

Definition 3: *Transit motions*
A transit motion $\texttt{TRANSIT}(\alpha, q_i, q_j)$ is a continuous motion of the robot arm from initial configuration $q_i \in \mathcal{Q}$ to $q_j \in \mathcal{Q}$ that is collision-free with respect to α.

Definition 4: *Transfer motions*
A transfer motion $\texttt{TRANSFER}(\alpha_i, o, q, q', \alpha_j)$ is a continuous motion of the robot arm, holding object $o \in \mathcal{O}$, from $q \in \mathcal{Q}$ to $q' \in \mathcal{Q}$, that is collision-free with respect to $\alpha_i \setminus o$. q and q' must place object o at $\alpha_i[o]$ and $\alpha_j[o]$, respectively.

Note that these motion classes do not predefine concrete motion primitives or actions. We are now equipped to formally state the object reconfiguration problem:

Algorithm 2: $\texttt{Connect}(\mathcal{T}, \alpha')$

do
  $\alpha_{\text{nearest}} \leftarrow \texttt{GetNearest}(\mathcal{T}, \alpha')$
  $\alpha_{\text{next}} \leftarrow \texttt{Extend}(\mathcal{T}, \alpha_{\text{nearest}}, \alpha', \texttt{True})$
while $\alpha_{\text{next}} \notin \{\alpha_{\text{nearest}}, \alpha'\}$
return α_{next}

Definition 5: *Object Reconfiguration Problem*
Given an initial valid arrangement (Def. 2), $\alpha_{\text{start}} \in \mathcal{A}$, robot configuration, $q_{\text{start}} \in \mathcal{Q}$, and valid goal arrangement $\alpha_{\text{goal}} \in \mathcal{A}$, the object reconfiguration problem is to find a sequence of objects and robot configurations, $[q_1, o_1, q_1', \ldots, q_n, o_n, q_n']$ and corresponding alternating TRANSIT and TRANSFER motions such that the sequence:

$$\begin{aligned} &\texttt{TRANSIT}(\alpha_{\text{start}}, q_{\text{start}}, q_1) \\ \rightarrow &\texttt{TRANSFER}(\alpha_0, o_1, q_1, q_1', \alpha_1) \\ \rightarrow &\cdots \\ \rightarrow &\texttt{TRANSIT}(\alpha_{n-1}, q_{n-1}', q_n) \\ \rightarrow &\texttt{TRANSFER}(\alpha_{n-1}, o_n, q_n, q_n', \alpha_n) \end{aligned}$$

is valid and $\alpha_n = \alpha_{\text{goal}}$, where α_i is the arrangement after executing the i-th TRANSFER motion.

This problem formulation is similar to that of [2], but adds a 3D workspace and consideration of stability constraints.

IV. Approach

We propose to solve the reconfiguration problem with a bidirectional tree search algorithm, SAST, that operates in a given problem's arrangement space. SAST resembles RRTConnect, but operates in the arrangement space with a novel extension operator that exploits an embedded physics simulator (Sec. IV-C) to automatically discover valid actions.

A. Stable Arrangement Search Trees (SAST)

SAST initializes two trees in the arrangement space, one rooted at the start arrangement, α_{start}, and the other at the goal arrangement, α_{goal}. Vertices in these trees represent valid arrangements (Def. 2); edges represent transformations between valid arrangements. In this work, we consider pick-and-place transformations which move *exactly* one object. Given two valid arrangements α_{i-1} and α_i, a connecting edge can be described as (q_i, o_i, q_i'). This transformation corresponds to a TRANSIT motion of the robot to q_i, followed by a stable TRANSFER motion moving o_i from its pose in α_{i-1} to α_i by grasping o_i and moving the robot from q_i to q_i'. Edges are bidirectional: the reverse transformation from α_i to α_{i-1} corresponds to a TRANSIT motion to q_i', followed by a stable TRANSFER motion of o_i from its pose in α_i to α_{i-1} by grasping o_i and moving the robot from q_i' to q_i. In the arrangement space representation, a solution to a reconfiguration problem is a path of edges that connect α_{start} to α_{goal}.

Planning starts from the tree rooted at α_{start}. Each iteration of the planning loop samples a random arrangement α_{rand} and finds its closest neighbor, α_{nearest} in the current tree (Alg. 1,

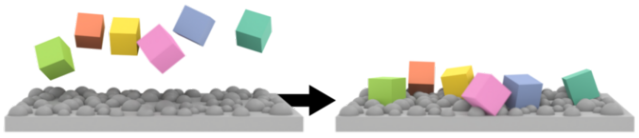


Fig. 3. Stable arrangement sampling. The pose of each object is uniformly sampled within workspace bounds; rejection sampling ensures no object intersection. The dynamics of the world are stepped until the arrangement stabilizes, that is, zero displacement over a sufficient number of steps.

lines 4 and 5). This is done via spatial lookup on a GNAT [50] with arrangement distance defined as the summed SE(3) distance[1] between the respective poses of each object in the two arrangements. SAST then attempts to Extend the tree from α_{nearest} toward α_{rand} by growing a sequence of edges according to Alg. 3. If the resulting sequence of edges is non-empty (Alg. 1, line 7), we try to Connect the other tree to the terminal vertex of the extended trajectory (Alg. 1, line 8). This is done (Alg. 2) by repeatedly extending the closest arrangement on the other tree to the terminal vertex, until either the connection succeeds or until the extension fails. If connection succeeds, SAST has found a solution and terminates. Otherwise, it swaps the trees and repeats the planning loop.

B. Sampling stable arrangements

The SampleArrangement subroutine (Fig. 3) samples a valid arrangement for use with Extend. Here, we leverage the embedded physics simulator to find stable arrangements. First, SampleArrangement picks uniform-random 3D poses for each object within the workspace bounds, using rejection sampling to ensure that the objects do not intersect. Then, it simulates the dynamics of the arrangement forward for a fixed number of small timesteps, checking at fixed intervals if the objects have maintain zero displacement since the previous interval. If so, the arrangement resulting from the applied dynamics is kinematically valid and statically stable, and is returned as a result. Otherwise, this process repeats until a valid sample is found. SampleArrangement is easy to parallelize—our implementation of SAST uses multiple threads to sample stable arrangements.

Uniform-random initial pose sampling trades off performance for ease of specification, avoiding the specialized samplers used by TAMP solvers to find states on low and zero-measure state manifolds.

C. Generating valid transformation actions

The Extend subroutine (Alg. 3) searches for a sequence of valid edges that transform a given arrangement α of k objects into a target arrangement α'. A major contribution of our work is to use an embedded physics simulator in Extend to reason about the validity of these transformations. The simulator allows us to treat the physics of the environment as an implicit specification of the valid transformation actions from any state. We also ensure that a valid transformation has a valid instantiation with the robot by motion planning for its associated TRANSIT and TRANSFER motions.

[1] SE(3) distance is the sum of the Euclidean distance of the translational components and the angular distance of the rotational components.

Algorithm 3: Extend$(\mathcal{T}, \alpha, \alpha', connectTarget)$

```
1  o_1, ..., o_k ← RandomObjectOrder()
2  α_cur ← Copy(α)
3  for i ← 1 to k do
4      α_next ← (α_cur[o_i] ← α'[o_i])
       // Sample grasp configurations.
5      q_i, q'_i ← SampleGraspConfs(α_cur[o_i], α'[o_i])
       // Perform checks.
6      if CollisionFree(α_next) = False continue
7      if Stable(α_cur \ o_i) = False continue
8      if Stable(α_next \ o_i) = False continue
9      q'_prev ← PrecedingConf(α_cur)
10     if invalid TRANSIT(α_cur, q'_prev, q_i) or invalid
         TRANSFER(α_cur, o_i, q_i, q'_i, α_next) continue
       // Connect target, or create edge.
11     if α_next = α' and connectTarget then
12         q'_next ← PrecedingConf(α')
13         if valid TRANSIT(α_next, q'_i, q'_next) then
14             T.AddEdge(α_cur, α', (q_i, o_i, q'_i))
15             return α'
16     else
17         T.AddVertex(α_next)
18         T.AddEdge(α_cur, α_next, (q_i, o_i, q'_i))
19         α_cur ← α_next
20 return α_cur
```

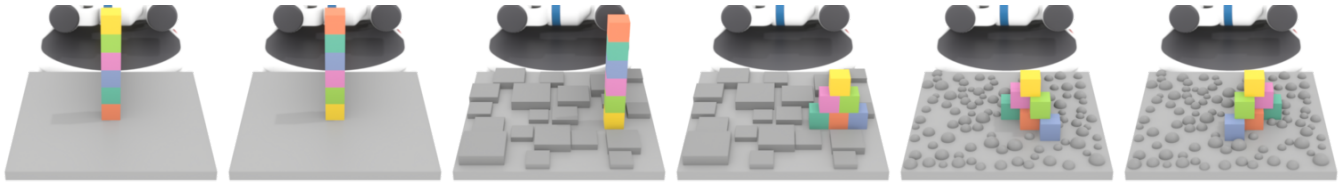
Extend starts by selecting a random order to move the objects[2] and setting the current arrangement, α_{cur} to the given start arrangement, α. It then tries to move each object in the chosen order to its target position in the given target arrangement, α', while maintaining stability of the other objects.

For each object, o_i, Extend creates a new arrangement α_{next} equal to α_{cur} with o_i at its pose in α' and samples collision free robot configurations grasping o_i's pose in α_{cur} and at α_{next}, using the same grasp. Then, it checks that: (1) α_{next} is collision-free, (2) $\alpha_{\text{cur}} \setminus o_i$ is stable, allowing o_i to be moved, and (3) $\alpha_{\text{next}} \setminus o_i$ is also stable, allowing o_i to be moved in the *reverse* transformation. If these conditions are met, Extend attempts to find a valid TRANSIT motion between the preceding configuration of α_{cur}[3] and the sampled grasp for o_i's pose in α_{cur}, and a valid TRANSFER motion between the sampled grasps for o_i's pose in α_{cur} and α_{next}, respectively, using a standard motion planner (Alg. 3, lines 9 and 10). These motions are considered infeasible if the subplanner fails to find a solution within a predefined timeout.

If Extend finds the requisite TRANSIT and TRANSFER motions, then it either adds the discovered edge to the *current* tree (Alg. 3, lines 17 to 19) and continues with the next object and $\alpha_{\text{cur}} = \alpha_{\text{next}}$, or attempts to connect the newly-reached

[2] We choose a random order for simplicity, but could substitute a more sophisticated permutation selector for performance.

[3] If $\alpha_{\text{cur}} = \alpha_{\text{start}}$, we select q_0 as q'_{prev}. If $\alpha_{\text{cur}} = \alpha_{\text{goal}}$, we skip this check since there is no constraint on the robot configuration at the goal.



(a) REVERSE Start (b) REVERSE Goal (c) TRANSFORM Start (d) TRANSFORM Goal (e) ROTATE Start (f) ROTATE Goal

Fig. 4. Test problems used in our experiments. (a-b) REVERSE: The robot starts with a stack of 6 cubes and must re-stack them on the same base location in reversed order. (c-d) TRANSFORM: The robot must transform a stack of 6 cubes into a pyramid, centered at the same location. The table is covered in tiles of random height and size, which constrain the feasible intermediate arrangements. (e-f) ROTATE: The robot is given a *diagonal* pyramid of cubes and must manipulate the cubes into another pyramid with cubes stacked cross-diagonally. The table is covered in bumps of random size and location, which prevent the cubes from being placed flat in intermediate arrangements and make it more difficult to find stable arrangements. In all problems, the robot starts with its arm tucked in (Fig. 1). Across runs, the x and y positions of the structures to reconfigure, as well as the tiles and bumps, are randomized.

arrangement to the *other* tree (Alg. 3, lines 11 to 14) if α_{next} is the target arrangement and a connection is desired. In the latter case, `Extend` returns the target arrangement (Alg. 3, line 15); otherwise, it iterates through the remaining objects and returns the last reached arrangement.

D. Solution simplification

Although unnecessary for completeness, `SAST` applies heuristic simplifications to solutions to improve their quality.

If an object o has been moved twice along a solution trajectory, one of these motions may be unnecessary. We can remove the first motion by altering the second motion to move o starting from the first motion's starting pose. Similarly, we can remove the second motion by altering the first motion to move o to the second motion's ending pose. Both cases modify the pose of o in the arrangements between the first and second motions. This requires recomputing the grasps and planning motions for these intermediate arrangements to validate the altered arrangement trajectory. In a third case, motions may also be removed if the pickup and placement locations of the object are exactly the same.

`SAST` iterates through these three simplification cases on solutions, rechecking for stability and recomputing the `TRANSIT` and `TRANSFER` motions after each modification to ensure that the solution remains feasible. This simplification process continues until no potentially redundant actions remain. Note that this heuristic set is non-exhaustive and does not guarantee optimal motions.

V. EXPERIMENTS

We evaluate `SAST` on a set of 3D tabletop rearrangement problems (Fig. 4)—REVERSE (a-b), TRANSFORM (c-d), and ROTATE (e-f). These problems involve using a single-arm manipulator to reconfigure cubes from one 3D structure to another. They require reasoning about the physical constraints between the cubes, as well as with the environment. The solutions are non-trivial, in that the robot must choose and move the objects through intermediate arrangements to achieve its goal. In addition, some problems contain obstacles such as tiles and bumps which complicate the validity of actions. Grounding these details in order to apply contemporary approaches would be tedious and challenging.

	REVERSE	TRANSFORM	ROTATE
Success Rate	0.96 (0.03)	0.96 (0.03)	1.00 (0)
Solve Time (s)	16.5 (0.9)	55.0 (5.6)	61.5 (6.6)
Solution Length	25.6 (0.7)	23.7 (0.8)	15.2 (0.5)
Num. Nodes	44.1 (2.4)	86.2 (9.6)	57.4 (6.0)

Table I. Run-time metrics of `SAST` across the test problems. *Success rate* refers to the proportion of runs where `SAST` finds a solution within the given time limit. For successful runs, *Solve Time* refers to the time taken to find a solution; *Solution Length* refers to the solution length, in terms of the number of objects moved; *Num. Nodes* refer to the total number of tree vertices created by `SAST` during search. Mean values are shown, with standard error shown in parentheses.

A. Implementation

We use DART [19] as our embedded physics simulator and plan `TRANSIT` and `TRANSFER` motions via Robowflex [51] with the Open Motion Planning Library (OMPL) [52]. All experiments ran on an AMD 5900x desktop CPU with 12 cores at 4.8 GHz, using 6 parallel threads for sampling stable arrangements and inverse kinematics.

B. Planning performance

We applied `SAST` to each test problem for 50 trials with a maximum timeout of 300 seconds per trial. In each trial, we randomize the start and goal positions of the structure to rearrange, together with the the obstacle positions.

Table I shows that `SAST` was almost always able to find a solution within the stipulated time limit. Solution times were also reasonable, taking not more than a minute per successful run despite having to invoke the simulator repeatedly for collision and static stability checking, and having to integrate the low-level motion planning of the `TRANSIT` and `TRANSFER` motions. The sizes of the search trees, in terms of the number of nodes, were also small, indicating that a sparse coverage of arrangements was sufficient to identify a solution.

Across each problem and trial, we only had to specify the geometry and positions of the obstacles (steps and bumps) and the start and goal arrangement poses of the objects. This highlights the strength of our approach in using the physics simulator to automatically derive action validity without requiring any manual, explicit specification.

Solution lengths, however, often require about twice the optimal number of steps. This is because `SAST`, like `RRTConnect`, is non-optimizing.

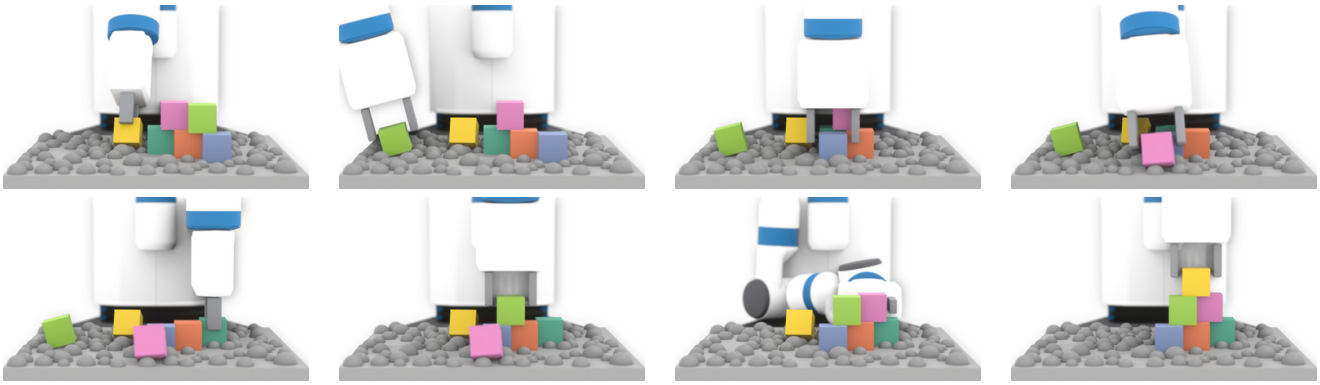


Fig. 5. Sequence of actions in a computed solution after simplification for the ROTATE problem. The robot is able to identify actions that remove the blocks on top before removing those at the bottom. This ensures that blocks are not removed from the bottom that will cause those stacked on top to topple. The sequence shown achieves the minimum possible length for the given problem.

	REVERSE	TRANSFORM	ROTATE
Simplify Time (s)	26.5 (1.4)	41.8 (2.7)	28.0 (2.0)
Simplified Solution Length	12.1 (0.1)	12.0 (0.1)	8.0 (0.1)
Improvement (%)	51.5 (1.2)	46.8 (1.7)	44.9 (1.7)

Table II. Solution simplification results. *Simplify Time* is the time taken for the simplification procedure to terminate. *Simplified Solution Length* is the length of the simplified solutions, in terms of the number of objects moved. *Improvement* is the percentage of the original solution length that simplification eliminated. Mean values are shown, with standard error in parentheses.

	REVERSE	TRANSFORM	ROTATE
w/ Motion Checks	1.00 (0)	1.00 (0)	1.00 (0)
w/o Motion Checks	0.60 (0.07)	0.42 (0.07)	0.56 (0.07)

Table III. Solution feasibility with and without low-level motion checking during the arrangement tree search. We show the mean proportion of solutions with feasible low-level motions; the standard error is in parentheses.

C. Simplification performance

The results in Table I do not use the solution simplification heuristics of Sec. IV-D. Table II shows the results of applying these heuristics to the solutions found by each successful run, indicating that solution simplification usually terminated within 40 seconds. Most of the additional time comes from rechecking for stability and replanning for the low-level TRANSIT and TRANSFER motions, required whenever two actions moving the same object merge. This is done up to $\mathcal{O}(n^3)$ times in the number of actions in the initial solution.

Simplification usually decreased solution length by roughly half, reaching or coming close to the optimal solution length. Fig. 5 shows an example of a simplified ROTATE solution.

D. How important is integrating motion planning?

SAST verifies the feasibility of each TRANSIT and TRANSFER motion by planning a valid trajectory in the robot's configuration space for the motion. To investigate the impact of this integrated verification, we conducted an ablation experiment by removing these low-level feasibility checks. We find solutions in terms of sequences of object arrangements and object grasps, assuming that the transformations between arrangements are feasible. After finding a full solution, we attempt to compute a motion plan for each of the associated low-level motions to check solution validity.

Table III shows a substantial drop in solution feasibility when low-level motion checks are skipped. Indeed, TRANSIT motions require checking that an object is reachable with a given grasp without the manipulator colliding with the other objects; TRANSFER motions require checking that an object can be pulled away without the object or the manipulator intersecting with the other nearby objects. The problems we consider have environment obstacles (table, tiles, and bumps) that do not interfere much with the robot's motion—in more constrained environments, such as in a cupboard or drawer, we could expect feasibility to worsen.

VI. DISCUSSION

This work contributes a novel perspective on manipulation planning that embeds a simulator to implicitly encode the valid actions and states for a problem domain. We demonstrate this perspective for 3D object reconfiguration planning, where we are able to efficiently find statically stable object configurations that would otherwise be onerous to specify.

SAST currently uses random sampling and extension to grow the arrangement space graph, but informed approaches like Expansive Space Trees [53] or Monte Carlo Tree Search [33] may discover solutions faster. SBMP advances such as biased samplers [54, 55] and optimizing planners [56–59] may also complement embedded-simulator planning.

Embedded-simulator planning is broadly applicable outside object reconfiguration, which poses several directions for future work. How can we use simulators to encode constraints beyond stability, such as orientation or contact dynamics? Similarly, what are the precise requirements for an embedded simulator? For some problems, precise physics simulation may be unnecessary; for others, non-standard physics can encode problem constraints. Further, how well do plans found via embedded simulation transfer to the real world?

Finally, we wish to explore richer uses of the embedded simulator, including combining differentiable simulation with optimization techniques, to broaden the manipulation problem classes that we can efficiently solve.

REFERENCES

[1] A. Krontiris and K. E. Bekris. "Dealing with Difficult Instances of Object Rearrangement". In: Robotics: Science and Systems. July 13, 2015.

[2] A. Krontiris and K. E. Bekris. "Efficiently Solving General Rearrangement Tasks: A Fast Extension Primitive for an Incremental Sampling-Based Planner". In: IEEE International Conference on Robotics and Automation. May 2016, pp. 3924–3931.

[3] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki. "An Incremental Constraint-Based Framework for Task and Motion Planning". In: *The International Journal of Robotics Research* 37.10 (Sept. 2018), pp. 1134–1151.

[4] R. Alami, T. Siméon, and J.-P. Laumond. "A Geometrical Approach to Planning Manipulation Tasks". In: International Symposium on Robotics Research. 1989, pp. 113–119.

[5] J. Barry, K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez. "Manipulation with Multiple Action Types". In: International Symposium on Experimental Robotics. Vol. 88. 2013, pp. 531–545.

[6] M. Toussaint. "Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning". In: International Joint Conference on Artificial Intelligence. 2015, p. 7.

[7] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. "PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning". In: International Conference on Automated Planning and Scheduling. 2020. arXiv: 1802.08705 [cs].

[8] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi. "Towards Manipulation Planning with Temporal Logic Specifications". In: IEEE International Conference on Robotics and Automation. May 2015, pp. 346–352.

[9] D. McDermott. "PDDL—The Planning Domain Definition Language". In: AIPS Planning Competition. 1998.

[10] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. "Integrated Task and Motion Planning". In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (2021), pp. 265–293.

[11] D. K. Misra, J. Sung, K. Lee, and A. Saxena. "Tell Me Dave: Context-Sensitive Grounding of Natural Language to Manipulation Instructions". In: *International Journal of Robotics Research* 35.1-3 (2016), pp. 281–300.

[12] W. Thomason and H. Kress-Gazit. *Counterexample-Guided Repair for Symbolic-Geometric Action Abstractions*. May 13, 2021. arXiv: 2105.06537.

[13] M. Vallati and I. Serina. "A General Approach for Configuring PDDL Problem Models". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 28. 1. 2018.

[14] W. Vega-Brown and N. Roy. "Admissible Abstractions for Near-optimal Task and Motion Planning". In: International Joint Conference on Artificial Intelligence. 2018, pp. 4852–4859.

[15] E. Dahlman and A. E. Howe. "A Critical Assessment of Benchmark Comparison in Planning". In: *Journal of Artificial Intelligence Research* 17 (July 2, 2002), pp. 1–33. arXiv: 1106.1804.

[16] V. Xia, Z. Wang, and L. P. Kaelbling. "Learning Sparse Relational Transition Models". In: *International Conference on Learning Representations*. 2019.

[17] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour. "Manipulation Planning Among Movable Obstacles". In: IEEE International Conference on Robotics and Automation. Apr. 2007, pp. 3327–3332.

[18] E. Coumans et al. *Bullet Physics Library*. 2013.

[19] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu. "Dart: Dynamic Animation and Robotics Toolkit". In: *Journal of Open Source Software* 3.22 (2018), p. 500.

[20] C. E. García, D. M. Prett, and M. Morari. "Model Predictive Control: Theory and Practice—A Survey". In: *Automatica* 25.3 (May 1, 1989), pp. 335–348.

[21] E. Heiden, D. Millard, H. Zhang, and G. S. Sukhatme. *Interactive Differentiable Simulation*. May 18, 2020. arXiv: 1905.10706 [cs, stat].

[22] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme. "NeuralSim: Augmenting Differentiable Simulators with Neural Networks". In: IEEE International Conference on Robotics and Automation. 2021.

[23] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox. "GPU-Accelerated Robotic Simulation for Distributed Reinforcement Learning". In: Conference on Robot Learning. Oct. 24, 2018, pp. 270–282.

[24] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning". In: Neural Information Processing Systems: Datasets and Benchmarks Track. 2021.

[25] O. Kroemer, S. Niekum, and G. Konidaris. "A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms". In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 1395–1476.

[26] J. Kober, J. A. Bagnell, and J. Peters. "Reinforcement Learning in Robotics: A Survey". In: *The International Journal of Robotics Research* 32.11 (Sept. 1, 2013), pp. 1238–1274.

[27] B. Shen, Z. Jiang, C. Choy, L. J. Guibas, S. Savarese, A. Anandkumar, and Y. Zhu. "ACID: Action-Conditional Implicit Visual Dynamics for Deformable Object Manipulation". In: Robotics: Science and Systems. Aug. 5, 2022.

[28] W. Zhao, J. P. Queralta, and T. Westerlund. "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey". In: *IEEE Symposium Series on Computational Intelligence*. Dec. 1, 2020, pp. 737–744.

[29] M. S. Saleem and M. Likhachev. "Planning with Selective Physics-based Simulation for Manipulation Among Movable Objects". In: IEEE International Conference on Robotics and Automation. May 2020, pp. 6752–6758.

[30] S. Zickler and M. Veloso. "Efficient Physics-Based Planning: Sampling Search Via Non-Deterministic Tactics and Skills". In: International Conference on Autonomous Agents and Multiagent Systems. 2009, p. 7.

[31] C. Zito, R. Stolkin, M. Kopicki, and J. L. Wyatt. "Two-Level RRT Planning for Robotic Push Manipulation". In: IEEE/RSJ International Conference on Intelligent Robots and Systems. 2012, pp. 678–685.

[32] J. A. Haustein, J. King, S. S. Srinivasa, and T. Asfour. "Kinodynamic Randomized Rearrangement Planning via Dynamic Transitions between Statically Stable States". In: IEEE International Conference on Robotics and Automation. 2015, pp. 3075–3082.

[33] B. Huang, A. Boularias, and J. Yu. "Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning". In: IEEE/RSJ International Conference on Intelligent Robots and Systems. 2022.

[34] W. C. Agboh and M. R. Dogar. *Robust Physics-Based Manipulation by Interleaving Open and Closed-Loop Execution*. June 23, 2021. arXiv: 2105.08325 [cs].

[35] W. C. Agboh, D. Ruprecht, and M. R. Dogar. "Combining Coarse and Fine Physics for Manipulation Using Parallel-in-Time Integration". In: International Symposium on Robotics Research. Vol. 20. 2019, pp. 725–740.

[36] T. Lozano-Pérez. "Spatial Planning: A Configuration Space Approach". In: *Autonomous Robot Vehicles*. Springer New York, 1990, pp. 259–271.

[37] L. E. Kavraki, P. Svestka, J. Latombe, and M. H. Overmars. "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces". In: *IEEE Transactions on Robotics and Automation* 12.4 (Aug. 1996), pp. 566–580.

[38] S. M. LaValle. *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.

[39] J. Kuffner and S. LaValle. "RRT-Connect: An Efficient Approach to Single-Query Path Planning". In: IEEE International Conference on Robotics and Automation. Vol. 2. 2000, pp. 995–1001.

[40] D. Hsu, J.-C. Latombe, and H. Kurniawati. "On the Probabilistic Foundations of Probabilistic Roadmap Planning". In: *The International Journal of Robotics Research* 25.7 (July 2006), pp. 627–643.

[41] S. Cambon, R. Alami, and F. Gravot. "A Hybrid Approach to Intricate Motion, Manipulation and Task Planning". In: *International Journal of Robotics Research* 28.1 (Jan. 1, 2009), pp. 104–126.

[42] D. Berenson, S. S. Srinivasa, and J. Kuffner. "Task Space Regions: A Framework for Pose-Constrained Manipulation Planning". In: *International Journal of Robotics Research* 30.12 (2011), pp. 1435–1460.

[43] S. Han, N. Stiffler, A. Krontiris, K. Bekris, and J. Yu. "High-Quality Tabletop Rearrangement with Overhand Grasps: Hardness Results and Fast Methods". In: Robotics: Science and Systems. 2017.

[44] R. Shome and K. E. Bekris. "Synchronized Multi-Arm Rearrangement Guided by Mode Graphs with Capacity Constraints". In: Workshop on the Algorithmic Foundations of Robotics. 2020, p. 16.

[45] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia. "Graph Networks as Learnable Physics Engines for Inference and Control". In: International Conference on Machine Learning. June 4, 2018, pp. 4470–4479.

[46] M. Chang, T. Ullman, A. Torralba, and J. Tenenbaum. "A Compositional Object-Based Approach to Learning Physical Dynamics". In: International Conference on Learning Representations. 2017.

[47] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu. "Interaction Networks for Learning about Objects, Relations and Physics". In: Neural Information Processing Systems. Dec. 1, 2016.

[48] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum. "Simulation as an Engine of Physical Scene Understanding". In: *Proceedings of the National Academy of Sciences* 110.45 (Nov. 5, 2013), pp. 18327–18332.

[49] K. Ren, L. E. Kavraki, and K. Hang. "Rearrangement-Based Manipulation via Kinodynamic Planning and Dynamic Planning Horizons". In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Aug. 3, 2022.

[50] S. Brin. "Near Neighbor Search in Large Metric Spaces". In: International Conference on Very Large Data Bases. 1995, pp. 574–584.

[51] Z. Kingston and L. E. Kavraki. "Robowflex: Robot Motion Planning with MoveIt Made Easy". In: IEEE/RSJ International Conference on Intelligent Robots and Systems. 2022.

[52] I. A. Sucan, M. Moll, and L. E. Kavraki. "The open motion planning library". In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 72–82.

[53] D. Hsu, J.-C. Latombe, and R. Motwani. "Path Planning in Expansive Configuration Spaces". In: IEEE International Conference on Robotics and Automation. Vol. 3. Apr. 1997, pp. 2719–2726.

[54] D. Hsu, T. Jiang, J. Reif, and Z. Sun. "The Bridge Test for Sampling Narrow Passages with Probabilistic Roadmap Planners". In: IEEE International Conference on Robotics and Automation. Vol. 3. Sept. 2003, pp. 4420–4426.

[55] Y. Lee, C. Chamzas, and L. E. Kavraki. "Adaptive Experience Sampling for Motion Planning Using the Generator-Critic Framework". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 9437–9444.

[56] S. Karaman and E. Frazzoli. "Sampling-Based Algorithms for Optimal Motion Planning". In: *The International Journal of Robotics Research* 30.7 (June 2011), pp. 846–894.

[57] J. D. Gammell and M. P. Strub. "Asymptotically Optimal Sampling-Based Motion Planning Methods". In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (May 3, 2021), pp. 295–318.

[58] M. P. Strub and J. D. Gammell. "AIT* and EIT*: Asymmetric Bidirectional Sampling-Based Path Planning". In: *The International Journal of Robotics Research* (2022).

[59] L. Janson, E. Schmerling, A. Clark, and M. Pavone. "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions". In: *The International journal of robotics research* 34.7 (2015), pp. 883–921.