Stability-Guided Exploration for Diverse Motion Generation

Eckart Cobo-Briesewitz*[1], Tilman Burghoff[1], Denis Shcherba[1], Armand Jordana[2] and Marc Toussaint[1,3]

***Abstract*— Scaling up datasets is highly effective in improving the performance of deep learning models, including in the field of robot learning. However, data collection still proves to be a bottleneck. Approaches relying on collecting human demonstrations are labor-intensive and inherently limited: they tend to be narrow, task-specific, and fail to adequately explore the full space of feasible states. Synthetic data generation could remedy this, but current techniques mostly rely on local trajectory optimization and fail to find diverse solutions. In this work, we propose a novel method capable of finding diverse long-horizon manipulations through black-box simulation. We achieve this by combining an RRT-style search with sampling-based MPC, together with a novel sampling scheme that guides the exploration toward stable configurations. Specifically, we sample from a manifold of stable states while growing a search tree directly through simulation, without restricting the planner to purely stable motions. We demonstrate the method's ability to discover diverse manipulation strategies, including pushing, grasping, pivoting, throwing, and tool use, across different robot morphologies, without task-specific guidance.**

I. Introduction

A. Motivation

Recent advances in deep learning have demonstrated impressive capabilities for controlling robots [1], [2], [3]. These developments emphasize the need for large-scale, diverse robotic datasets to enable further progress. However, compared to domains like vision and language, datasets for robotics are not as readily available. This has led to research efforts directed towards obtaining more data. For this purpose, several different approaches have been explored. One of the most prominent in the field of behavioral cloning is teleoperation [4], [5], [6], where a human operator controls a robot to solve a specific task. While this can lead to high quality real world data, it requires extensive work from human experts, which is both expensive and time consuming. For reference, current large scale robotic datasets contain around a few million demonstrations [7], whereas VLM datasets comprise billions of image-text pairs [8]. An alternative approach is to extract data from human video demonstrations [9], [10], by using computer vision to estimate the positions of humans directly from online videos [11]. The motivation for this type of method is the large amount of human videos found on the internet. However, these techniques do not produce the necessary low-level information needed for training robots. Furthermore, they only focus on human-like approaches, neglecting the diversity of robot morphologies.

*Corresponding Author: cobo-briesewitz@campus.tu-berlin.de
[1]Technische Universität Berlin
[2]LAAS-CNRS
[3]Robotics Institute Germany (RIG)

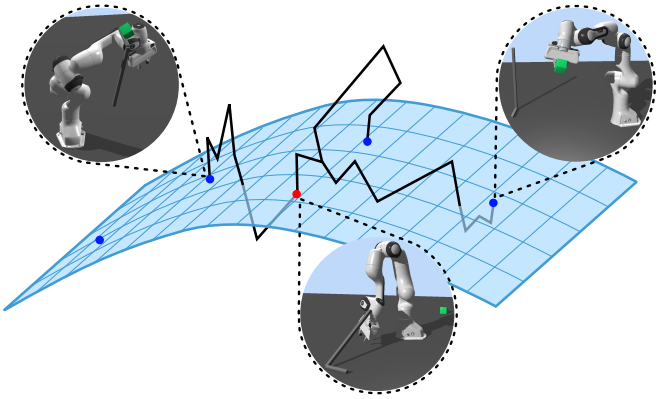


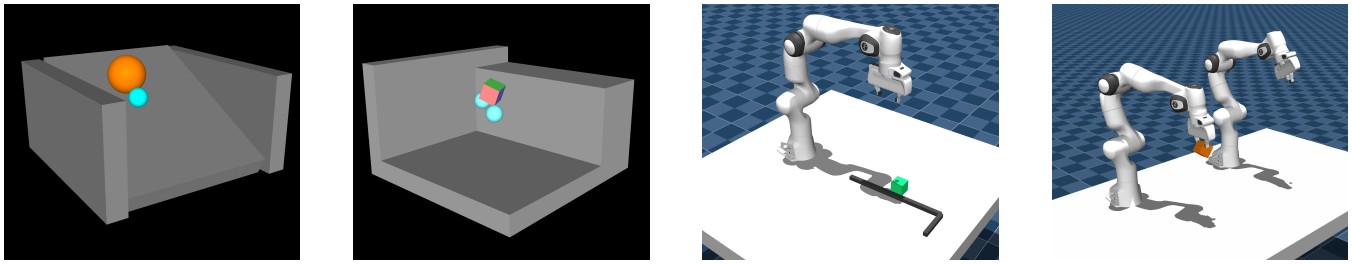
Fig. 1. Schematic of the paths found by our method. For each start node (red), we grow a tree *guided* by the manifold of stable states, but not constrained by it.

Overall, human generated data is inherently limited, as it may not be optimal and can fail to capture the full range of solutions accessible to robots. In contrast, algorithmically generated synthetic data can provide a broader set of valid solutions. Consequently, in this work, we are interested in the use of simulators to directly create high-quality and diverse data on tasks involving complex robot-object interactions.

Sampling-based Model Predictive Control (MPC) has recently shown promising results in generating dynamic motions by directly interacting with the simulator, both in locomotion [12] and manipulation [13]. However, these techniques mostly rely on local exploration in the control space [14], and are therefore prone to local minima. In contrast, sampling-based planners [15], [16] have a long history of relying on state space sampling to explore globally. We aim to combine both approaches to generate diverse, dynamic and contact-rich manipulations.

B. Related Work

a) Sampling-based MPC: MPC defines a controller as the solution to an optimal control problem. While MPC in robotics has historically relied on gradient-based optimization techniques [17], [18], recent advances in parallel computing have enabled the successful deployment of gradient-free techniques in MPC on robots [12], [13]. In practice, most techniques currently used in robotics rely on some form of iterative procedure where samples following a Gaussian distribution are used to evaluate the function around the current guess in order to find a descent direction [14]. For instance, Predictive Sampling simply picks the best sample to be the new guess around which the next Gaussian samples are drawn [19]. MPPI uses a softmax weighting of the samples



(a) SpheresRamp (b) SpheresCube (c) PandaHook (d) PandasCube

Fig. 2. The four environments used in our experiments. For the first two scenes the blue balls represent robots containing a translational joint in 3D space. The complexity of (a) lies in the high likelihood of landing in local minima, as once the object (orange) falls off the ramp it is impossible to recover. Environment (b) tests how well the algorithm can find diverse paths using two possible robot contact points as well as having a state where the rotation of the object is highly relevant. In (c) the method is tested for the ability of finding paths containing tool use. And environment (d) test the method on a high dimensional space where two robots can cooperate with each other to manipulate an object.

to compute an update direction [20]. CMA-ES relies on a weighted average of the k-best samples (and also iteratively updates the covariance matrix) [21].

The main strength of these techniques is that they can find interesting robot motions simply by interacting with the simulator. However, these approaches are inherently local. In many MPC and trajectory optimization methods, sampling is performed in the control space, often using single-shooting formulations, as handling constraints directly in gradient-free optimization is difficult. In contrast, graph-based motion planning methods sample in the state space and can offer theoretical guarantees such as probabilistic completeness.

b) Sampling Based Motion Planning: Sampling-based motion planners solve motion planning problems using probabilistic methods. They are commonly divided into two categories: *single-query motion planning*, with the most prominent approach being Rapidly-exploring Random Trees (RRTs) [15], and *multi-query motion planning*, which is typically addressed using probabilistic roadmaps (PRMs) [22]. RRTs incrementally build a tree by sampling random states and extending the nearest node in the tree toward each sample. In contrast, PRMs build a graph by sampling points in the configuration space and attempting to connect nearby samples. This roadmap can then be reused to efficiently answer motion planning queries online. These algorithms are probabilistically complete under mild assumptions [23], meaning that they find a solution, if it exists, as the number of samples tends to infinity.

These methods have proven effective in many domains and have led to various extensions being developed [24], [25], [26]. In the context of our work, an important extension is manifold RRTs [27], [28]. When constraints imposed on the system define a manifold of lower dimension than the ambient space, the probability of finding feasible points through uniform sampling in the ambient space approaches zero. Manifold RRTs address this issue by projecting sampled points onto the manifold and modifying the extend step to ensure that the tree grows along it.

In our work, we use the manifold of stable configurations to guide the search. Similarly to manifold RRT, we consider points projected onto a manifold as targets for extension; however, we do not restrict the paths to remain within the manifold. In that sense, our method is guided by a manifold of stable states while still being able to explore unstable states outside of it. This enables us to find diverse non-prehensile manipulations.

c) Kinodynamic planning: Kinodynamic planning aims to solve problems involving both kinematics and dynamic constraints. The most direct approach is to use trajectory optimization [29]; however, similarly to sampling-based MPC, it is subject to local minima and therefore cannot solve tasks involving long horizons. Another approach is to use Task and Motion Planning (TAMP) [30]; however, this implies a tedious manual definition of predicates. Alternatively, one could use sampling-based planners. For instance, RRTs can naturally be used for kino-dynamic planning by modifying the extend step to select a control input, typically by sampling multiple candidates and keeping the one that yields a next state closest to the target [16]. Some works [31], [32] combined trajectory optimization and RRTs; however, they rely on gradient based optimization, which limits their application to problems involving complex contact interactions. In the context of manipulation, [33], [34] combined RRT and motion primitives to perform non-prehensile manipulation. In [35] contacts are used for guiding motion planning, but is restricted to quasidynamic manipulation.

Conceptually similar to our work, [36] uses stability to guide a kino-dynamic RRT; however, they only consider non-prehensile manipulations on a 2-dimensional plane. Furthermore, the authors enforce that the system ends up in a stable state after each action, while our approach allows the robots to execute multiple actions without intermittent stability.

C. Contributions

Our contributions are as follows:

(i) We introduce *StaGE*, a novel algorithm to find complex and diverse long-horizon manipulations without motion priors. We achieve this by using our novel sampling scheme, *Stability-Guidance*, to guide a kino-dynamic RRT that directly interacts with black-box simulation. In addition, we provide a set of extensions to kino-

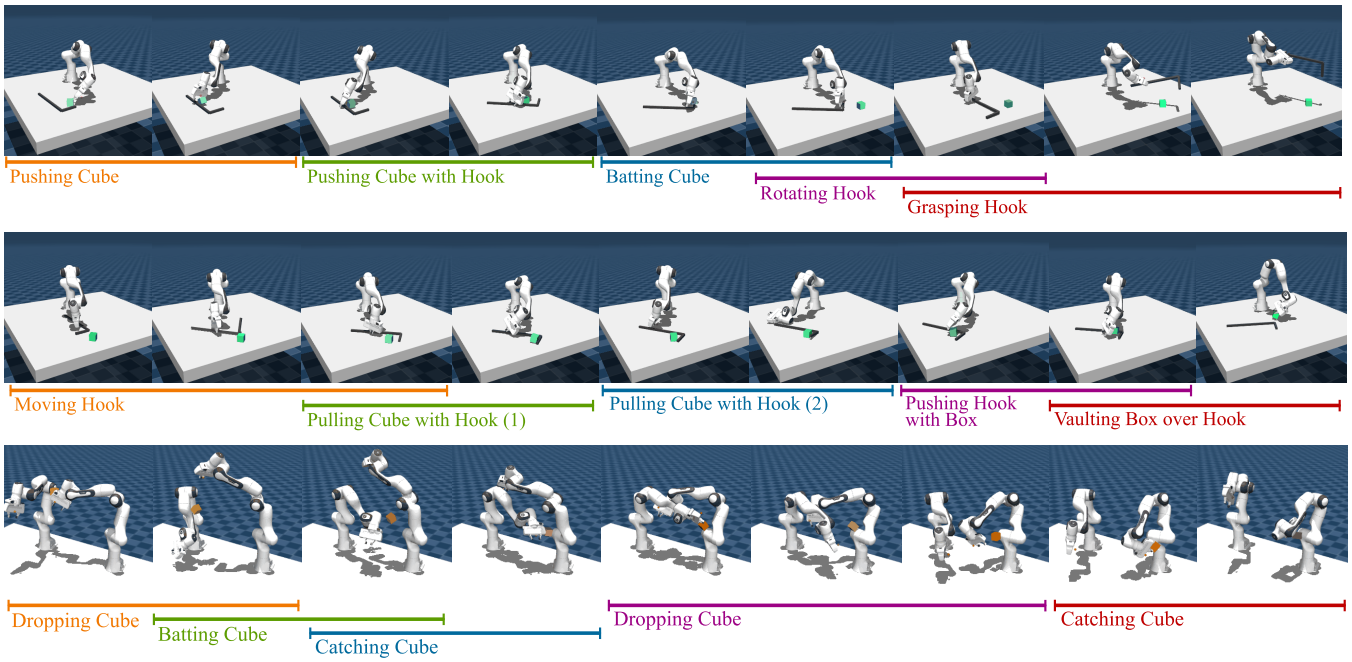


Fig. 3. Examples of trajectories generated by *StaGE*. The first trajectory shows diverse manipulations in the *PandaHook* environment, while the second one demonstrates tool use (pulling the cube with the hook). The third trajectory shows a transfer of the cube from the left panda to the right, by throwing and catching the cube multiple times.

dynamic RRT to encourage the generation of diverse behaviors.

(ii) Evaluations in multiple challenging environments with different robot morphologies qualitatively demonstrate the nature of the manipulations found by our approach.

Importantly, the proposed method is task-agnostic. All the obtained motions naturally emerge and do not rely on manually tuned cost functions. To the best of our knowledge, this is the first generic method applying RRT with black-box simulation to non-prehensile manipulation without the use of hand-crafted motion primitives or analytical constraints.

II. Method

The aim of our method is to find diverse, contact-rich manipulations in any scene, independent of any task. To that end, we consider a hierarchy of subspaces $\mathcal{C}_{stable} \subset \mathcal{C}_{feasible}$, where $\mathcal{C}_{feasible}$ describes the space of all reachable states in a given black-box simulation, and $\mathcal{C}_{stable}$ additionally imposes that the state is stable, i.e., all objects are in equilibrium. Our method is divided into two stages. First, we sample states from $\mathcal{C}_{stable}$ using a diverse constraint solver. In the second stage, we use an RRT-style planner to find diverse paths between states. Importantly, the sampled random states are only used to guide the search; that is to say, the RRT-style planner is free to evolve through non-stable regions to enable dynamic manipulation. We provide a visual intuition for this approach in Fig. 1. The pseudo-code for *StaGE* can be seen in Alg. 1.

A. Sampling Physically Stable States

In the first stage of our algorithm, we sample a set of fixed stable states $\mathcal{C}_s \subset \mathcal{C}_{stable}$ to guide the search later on. To that end, we follow the constrained sampling method introduced in [37], which generates states by solving a non-linear program. We briefly explain the method here and refer the reader to [37] for more details.

The method first samples contact variables $c_{ij} \in \{0, 1\}$, indicating whether the i-th and j-th frames in our scene are in contact. For scenes with one underactuated object (scenes (a), (b), and (d) in Fig. 2), we sample 1 to 3 support frames. If the scene contains multiple objects (scene (c) in Fig. 2), we sample uniformly from the set of up to three contacts, where each contact includes at least one underactuated object. Then, for each active contact, we add a point of attack p_{ij}, constrained to lie on both surfaces, as well as a force f_{ij} acting on the frames, constrained to be within a predefined friction cone [38]. Each underactuated object is constrained to be in a quasi-static equilibrium, i.e. the sum of all forces and moments acting on it through the contacts and gravity is zero. Lastly, we constrain the state to be collision-free.

We then try to find a state x satisfying these constraints by uniformly sampling a random initial state $\bar{x}$ formulating the optimization problem

$$\min_x \|x - \bar{x}\|^2$$
$$\textit{subject to} \left\{ \begin{array}{c} \text{collision} \\ \text{contact} \\ \text{force} \end{array} \right\} \text{-constraints},$$

which we solve using an augmented Lagrangian method [39]. This formulation can be understood as a random projection onto the manifold C_{stable}. Some examples of the resulting states can be seen in Fig. 1.

While this method works well for the simple contact dynamics in the first two scenes in Fig. 2, it can struggle with the more complex meshes of the robotic arm introduced in scenes (c) and (d). That is to say, the optimizer may not find a solution that satisfies the constraint. If so, we sample a random initial state $\bar{x}$ again and try to solve it until success or a given number of trials is reached. In practice, in the *PandaHook* scene, we achieve a 100% success rate in finding a stable state within 100 attempts when both objects remain in contact with the table. However, imposing a grasp constraint (i.e., enforcing contact between an object and both finger frames of the gripper) reduces performance. The success rate drops to 15.1% when attempting to grasp the cube and to only 1.4% for grasping the hook. To remedy that, we first sort the contacts into high-level abstractions (namely contact with table, with panda or grasps) and then filter our stable states to have roughly equal numbers for each class. We note that another way to circumvent this problem would be to handle these interactions by using specialized (grasp) samplers [40], [41]. However, this is beyond the scope of this work.

B. Connecting States

To achieve our goal of finding diverse interactions, we build upon kinodynamic RRT [16]. In its most basic version, kinodynamic RRT grows a tree rooted at a starting state by uniformly sampling a state and extending the closest node of the tree towards it. The extension works by selecting an action and simulating its effect for a fixed time-step. The end-state of that roll-out is then added as a new leaf of the tree. One possible scheme to select the action is to evaluate random actions through a physical engine and take the action that results in a state minimizing the distance to the sampled target state, as proposed in [36]. In this work, we refer to this approach as RRT-sim.

While being generic, this approach is insufficient for our use-case. Intuitively, sampling directly in the configuration space will lead to too many uninteresting or irrelevant configurations (e.g., the tool flying in the air). To circumvent this issue, we propose *stability-guidance*, which means that instead of uniformly sampling from the space $C_{feasible}$, we only sample from the embedded manifold C_{stable}. Sampling from this manifold is expensive due to the complex, contact-rich interactions. Therefore, we first generate a set of fixed stable states C_s, as described in section II-A, from which we draw a point to extend the tree towards. To compute the state to extend from and the action selected for extension, we use the weighted mean squared error as a metric, weighing the position error of non-actuated objects higher than the joint-state error of the robot.

Furthermore, our goal is to find many *diverse* paths between multiple nodes in a dynamic, underactuated system with complex non-prehensile interactions. To improve exploration, we propose three additional extensions:

(i) **Sampling from the K-Nearest Neighbors** Instead of taking the node closest to the sampled stable state, we uniformly choose one of its k-nearest neighbors. This enables us to grow the tree (and consequently find more paths), even if the closest node is already within the minimum distance to the target state. To efficiently compute the k-nearest nodes for each stable state, we exploit the fact that the stable states are fixed: each stable state keeps a record of its k-nearest nodes, which we update each time we add a new node to our tree. This allows us to execute all the k-nearest neighbor searches and updates during the $N_{\max}$ iterations of the algorithm with $m = |C_s|$ stable states in $O(N_{max} m \log k)$ instead of $O(N_{max}(\log N_{max} + k))$ when using approximate nearest neighbors.[1] Since $N_{max} \gg m$, we use our (exact) implementation.

(ii) **N-Best Actions** Selecting the n-best actions that reduce the distance to the sampled target state, instead of only choosing the single best one, results in greater diversity in the paths found by our method.

(iii) **Node Rejection** If a node fails to expand the tree towards any of the target stable states, we consider it to have a high likelihood of being a dead-end and therefore do not try to expand it further. Our environments contain unrecoverable states. For example, if the sphere falls off the ramp in scene (a) from Fig 2 or if the cube is pushed out of reach in (d). We believe that dead-ends act as a task-agnostic proxy for such states.

C. Extracting Paths

We create paths from the tree grown in the second stage by selecting all nodes x that fall within a certain minimum distance ε to any of the stable states and extracting the path from x_0 to x.

Afterwards, we filter out redundant paths for each goal state by iterating through the paths in random order and greedily taking a path if its distance to all previously taken paths is greater than a minimum threshold. We compute the distance between paths (given as sets P, Q) using the (undirected) Hausdorff-distance

$$d_H(P,Q) = \max\{\hat{d}_H(P,Q), \hat{d}_H(Q,P)\} \quad \text{for} \quad \hat{d}_H(P,Q) = \max_{p\in P} \min_{q\in Q} d(p,q)\,. \tag{1}$$

III. Experiments

We evaluate our method across four challenging environments. We provide baselines and perform ablations of all major algorithmic contributions.

A. Environments

We use four distinct environments in our experiments, each using different robot morphologies and highlighting different challenges. The environments are depicted in Fig. 2.

[1] in our implementation, the selection step is constant, while the update step costs $O(m \log k)$ when using a max-heap to save the nearest nodes for each iteration. Using approximate nearest neighbors [42], [43] one could achieve an expected runtime of $O(\log i + k)$ in the i-th iteration, leading to the overall runtime $O(N_{max}(\log N_{max} + k))$.

Algorithm 1: StaGE

Input: Maximum number of nodes $N_{\max}$, number of stable configurations m, nearest neighbors k, number of best actions n, minimal path difference d_{min}

Output: Set of feasible paths $\mathcal{P}$

$C_s \leftarrow \textsc{SampleStableStates}(m)$;
$x_0 \sim C_s$;
// -- Tree Construction --
$\mathcal{T} \leftarrow \textsc{InitializeTree}(x_0)$;
for $i \leftarrow 1$ **to** $N_{\max}$ **do**
  $x_{\text{target}} \sim C_s$;
  $\mathcal{X}_{\text{near}} \leftarrow \text{KNEAREST}(\mathcal{T}, x_{\text{target}}, k)$;
  $x_{\text{near}} \sim \mathcal{X}_{\text{near}}$;
  $(\mathcal{A}_n, \mathcal{X}_n) \leftarrow \textsc{OptimizeActions}(x_{\text{near}}, x_{\text{target}}, n)$;
  if $\textit{ReducesDistance}(\mathcal{X}_n, C_s)$ **then**
    $\mathcal{T}.\textsc{Expand}(x_{\text{near}}, \mathcal{A}_n, \mathcal{X}_n)$;
  else
    $\mathcal{T}.\textsc{Disable}(x_{\text{near}})$;
// -- Path Extraction --
$\mathcal{P} \leftarrow \emptyset$;
foreach $x \in \mathcal{T}$ **do**
  foreach $x_s \in C_s \setminus \{x_0\}$ **do**
    if $\textit{Distance}(x, x_s) < \varepsilon$ **then**
      $p \leftarrow \textsc{PathToRoot}(x)$;
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{p\}$;
$\mathcal{P} \leftarrow \textsc{RemoveRedundant}(\mathcal{P}, d_{min})$;
return $\mathcal{P}$;

(i) **SpheresRamp** The environment consists of a single robot (blue) with a three-dimensional translational joint, manipulating an object (orange) on a ramp with two walls on the sides; a task similar to [44], but on a sloped surface. This environment enforces non-prehensile manipulation, since grasps are not possible. Furthermore, it contains non-recoverable states when the ball falls off the ramp.

(ii) **SpheresCube** This environment contains two robots (blue), each with three-dimensional translational joints, manipulating a cube. The environment contains two walls. This allows for diverse manipulations, where the cube can be pushed, grasped, thrown, or pivoted using the walls and floor. Additionally, the environment also forces the method to find manipulations that change the orientation of the cube.

(iii) **PandaHook** The environment contains a single Franka Emika Panda robotic arm, as provided by [45], together with a cuboid and a hook. This environment enables more complex interaction where, for example, the robot can use the hook to reach and manipulate the cube. This environment motivates tool use and demonstrates sequential manipulation planning, where long horizon multi-step manipulations are needed.

(iv) **PandasCube** The environment contains two panda robot arms in a bi-manual setup with a cuboid. This environment encourages collaboration, like throwing the cuboid from one arm to the other.

B. Evaluation Metrics

We evaluate our baselines and method on four different metrics intended to capture the quantity and diversity of the resulting data.

(i) **Path Count** The total number of diverse paths returned by our method.

(ii) **Coverage** The percentage of stable states sampled at the initial stage, which were reached by at least one node.

(iii) **Entropy** The entropy of the states visited by the paths, calculated using the Kozachenko–Leonenko estimator [46]. This estimator approximates the entropy based on the distance of the k-th nearest neighbor. We sample 100 states[2] from all visited states and calculate the entropy using $k = 10$. We repeat this process 10 times and report the average.

(iv) **Average Hausdorff** The average pairwise (undirected) Hausdorff distance (1) between paths with the same stable state as the endpoint, averaged over all stable end states with at least two paths connecting to it.

For evaluation, we fix the set of stable states for each environment. We evaluate all baselines and ablations using 10 randomly drawn starting states and average the results. We only report the entropy if the paths returned by the method add up to at least 100 nodes, and we only report the average Hausdorff-distance if a stable state is reached by at least two paths.

C. Baselines

We compare our method against kinodynamic RRT through black-box simulation (RRT-sim) introduced in sec. II-B. We modify it by adding a bias towards sampling stable states with 20% probability, which corresponds to *goal-bias* in kinodynamic RRTs [16].

Additionally, we evaluate the impact of our proposed extensions with ablations. For these, we take the number of selected actions $n \in \{1, 16\}$ and the number of nearest neighbors $k \in \{1, 16\}$. We also perform an ablation with only a 20% chance of sampling stable states, the rest of the samples being uniform. We give a budget of 2,500 tree expansions for the *SpheresRamp* and *SpheresCube* environments and a budget of 10,000 tree expansions for the *PandaHook* and *PandasCube* environments.

Furthermore, we also consider as a baseline sampling-based MPC with predictive sampling. We rely on the implementation provided in Hydrax [47] on the first (simplest) environment.

[2]We use a fixed number of states, since the KL-estimator depends on it and the number of visited states varies by multiple orders of magnitude between the methods we compare.

TABLE I

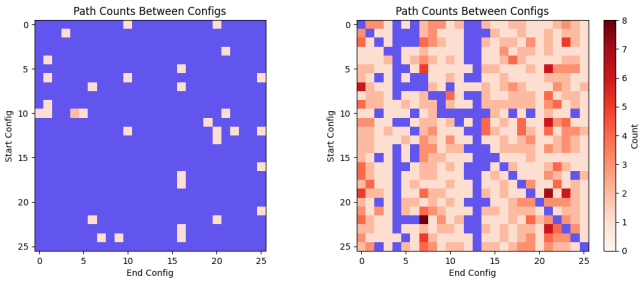
EXPERIMENTAL RESULTS ACROSS FOUR METRICS.

Method	Count	Cov. [%]	Entropy [nats]	Avg. Haus. [$\times 10^{-2}$]
SpheresRamp				
Predictive Sampling	14.3	57.2	—	—
RRT-sim	2.5	10	—	—
StaGE-uniform-80%	30.4	61.6	−27.21	11.27
StaGE w/o node rejection	50	76.8	−28.38	14.64
StaGE w/o n-best actions	14.8	37.2	−39.11	17.27
StaGE w/o k-NN	47	61.2	−30.96	**18.13**
StaGE	**68.8**	**85.2**	**−26.84**	12.52
SpheresCube				
RRT-sim	0.1	0.1	—	—
StaGE-uniform-80%	16.1	2.83	−18.66	14.05
StaGE w/o node rejection	110.8	12.42	**−3.37**	14.95
StaGE w/o n-best actions	0.3	0.2	—	**24.72**
StaGE w/o k-NN	39.3	7.07	−6.91	19
StaGE	**134.2**	**16.97**	−13.83	15.25
PandaHook				
RRT-sim	0	0	—	—
StaGE-uniform-80%	4.8	0.36	18.67	31.72
StaGE w/o node rejection	40.9	1.01	30.3	39.6
StaGE w/o n-best actions	0	0	—	—
StaGE w/o k-NN	45.4	0.65	31	**50.25**
StaGE	**48.7**	**1.44**	**37.48**	40.22
PandasCube				
RRT-sim	3	0.87	—	54.22
StaGE-uniform-80%	24.7	4.45	32.13	55.35
StaGE w/o node rejection	**198.8**	**24.68**	51.27	58.86
StaGE w/o n-best actions	7.4	1.34	—	53.7
StaGE w/o k-NN	40	6.02	35.21	**66.14**
StaGE	170	21.34	**52.12**	58.4

IV. RESULTS

Table I shows the results of our experiments. Our method performs the best in terms of coverage and paths found for all environments except *PandasCube*. We hypothesize that the improvement of *StaGE w/o node rejection* over *StaGE* is due to the high dimension of the action space (two 8-DOF robotic arms). Since many actions (e.g. null-space movements or movements of the arm not in contact with the box) will not improve the state cost, it is reasonable to assume that the size of the set of actions improving our state is small. Therefore, even on a previously unsuccessful node, actions improving the distance can be sampled in subsequent iterations. Future work could investigate alternative sampling schemes for sampling actions, in order to improve their the chances of success. Fig. 4 more clearly shows the ability of our method to explore the space of feasible actions in the *SpheresRamp* environment compared to the *RRT-sim* baseline. Fig. 3 shows some examples of the trajectories found by *StaGE*.

Generating the trajectories on an AMD Ryzen 9 9900X CPU takes 4 seconds per trajectory for the *SpheresRamp* scenario, 1 minute per trajectory for *SpheresCube*, 8 minutes per trajectory on *PandaHook*, and 30 seconds per trajectory on *PandasCube*. Significant speedups are possible by parallelizing the simulation using a GPU, which is beyond the scope of this work.



(a) RRT-sim (b) StaGE

Fig. 4. Adjacency matrices for trajectories found between a set of 26 stable states for the *SpheresRamp* environment. Blue indicates no connections were found between two states, the count value on the right refers to the amount of diverse paths found for each pair of stable states.

V. CONCLUSION

This paper introduced *StaGE*, a novel method for diverse motion generation in complex non-prehensile manipulation scenarios. We demonstrated that our approach generalizes to a wide range of novel environments without requiring task-specific guidance, showing that pure exploration alone can yield long-horizon sequential manipulation behaviors. Even in the absence of motion priors, the method discovers highly complex skills such as throwing, grasping, pivoting, pushing, handovers, and tool use, relying solely on stable states as guidance.

ACKNOWLEDGMENTS

This work has received support from the German Federal Ministry of Research, Technology and Space (BMFTR) under the Robotics Institute Germany (RIG).

REFERENCES

[1] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Proceedings of The 7th Conference on Robot Learning*, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, Nov 2023, pp. 2165–2183. [Online]. Available: https://proceedings.mlr.press/v229/zitkovich23a.html

[2] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter *et al.*, "PaLM-e: An embodied multimodal language model," in *Proceedings of the 40th International Conference on Machine Learning*, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, Jul 2023, pp. 8469–8488. [Online]. Available: https://proceedings.mlr.press/v202/driess23a.html

[3] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail *et al.*, "$\pi_{0.5}$: a vision-language-action model with open-world generalization," in *Proceedings of The 9th Conference on Robot Learning*, J. Lim, S. Song, and H.-W. Park, Eds., vol. 305. PMLR, Sep 2025, pp. 17–40. [Online]. Available: https://proceedings.mlr.press/v305/black25a.html

[4] H. Fang, C. Wang, Y. Wang, J. Chen, S. Xia, J. Lv *et al.*, "Airexo-2: Scaling up generalizable robotic imitation learning with low-cost exoskeletons," in *Proceedings of The 9th Conference on Robot Learning*, J. Lim, S. Song, and H.-W. Park, Eds., vol. 305. PMLR, Sep 2025, pp. 198–220. [Online]. Available: https://proceedings.mlr.press/v305/fang25a.html

[5] Y. Ze, Z. Chen, J. P. Araujo, Z.-a. Cao, X. B. Peng, J. Wu *et al.*, "Twist: Teleoperated whole-body imitation system," in *Proceedings of The 9th Conference on Robot Learning*, J. Lim, S. Song, and H.-W. Park, Eds., vol. 305. PMLR, Sep 2025, pp. 2143–2154. [Online]. Available: https://proceedings.mlr.press/v305/ze25a.html

[6] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016 .

[7] Open X-Embodiment Collaboration, "Open X-Embodiment: Robotic learning datasets and RT-X models," May 2025. [Online]. Available: https://arxiv.org/abs/2310.08864

[8] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti *et al.*, "Laion-5b: An open large-scale dataset for training next generation image-text models," in *Advances in Neural Information Processing Systems*, vol. 35. Red Hook, NY, USA: Curran Associates, Inc., 2022, pp. 25 278–25 294.

[9] Y. Liu, W. C. Shin, Y. Han, Z. Chen, H. Ravichandar, and D. Xu, "Immimic: Cross-domain imitation from human videos via mapping and interpolation," in *Proceedings of The 9th Conference on Robot Learning*, J. Lim, S. Song, and H.-W. Park, Eds., vol. 305. PMLR, Sep 2025, pp. 834–858. [Online]. Available: https://proceedings.mlr.press/v305/liu25b.html

[10] M. Xu, H. Zhang, Y. Hou, Z. Xu, L. Fan, M. Veloso *et al.*, "Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation," in *Proceedings of The 9th Conference on Robot Learning*, J. Lim, S. Song, and H.-W. Park, Eds., vol. 305. PMLR, Sep 2025, pp. 437–459. [Online]. Available: https://proceedings.mlr.press/v305/xu25b.html

[11] R. McCarthy, D. C.H. Tan, D. Schmidt, F. Acero, N. Herr, Y. Du *et al.*, "Towards generalist robot learning from internet video: A survey," *Journal of Artificial Intelligence Research*, vol. 83, Jul. 2025. doi: 10.1613/jair.1.17400 .

[12] H. Xue, C. Pan, Z. Yi, G. Qu, and G. Shi, "Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 4974–4981. doi: 10.1109/ICRA55743.2025.11127320 .

[13] A. H. Li, P. Culbertson, V. Kurtz, and A. D. Ames, "Drop: Dexterous reorientation via online planning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 14 299–14 306. doi: 10.1109/ICRA55743.2025.11128433 .

[14] A. Jordana, J. Zhang, J. Amigo, and L. Righetti, "An introduction to zero-order optimization techniques for robotics," 2025. [Online]. Available: https://arxiv.org/abs/2506.22087

[15] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," ComputerScience Dept. Iowa State University, Tech. Rep. TR 98-11, Oct. 1998.

[16] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 1, May 1999, pp. 473–479. doi: 10.1109/ROBOT.1999.770022 .

[17] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau *et al.*, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2536–2542.

[18] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli *et al.*, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, 2022.

[19] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive sampling: Real-time behaviour synthesis with mujoco," *arXiv preprint arXiv:2212.00541*, 2022.

[20] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1433–1440.

[21] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[22] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug. 1996. doi: 10.1109/70.508439 .

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

[24] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, Apr. 2000, pp. 995–1001. doi: 10.1109/ROBOT.2000.844730 .

[25] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 2011. doi: 10.1177/0278364911406761 .

[26] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2997–3004. doi: 10.1109/IROS.2014.6942976 .

[27] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 625–632. doi: 10.1109/ROBOT.2009.5152399 .

[28] C. Suh, T. T. Um, B. Kim, H. Noh, M. Kim, and F. C. Park, "Tangent space RRT: A randomized planning algorithm on constraint manifolds," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4968–4973. doi: 10.1109/ICRA.2011.5980566 .

[29] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[30] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling *et al.*, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, no. 1, pp. 265–293, 2021.

[31] S. Choudhury, J. D. Gammell, T. D. Barfoot, S. S. Srinivasa, and S. Scherer, "Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4207–4214.

[32] J. Ortiz-Haro, W. Hönig, V. N. Hartmann, and M. Toussaint, "idb-a*: Iterative search and optimization for optimal kinodynamic motion planning," *IEEE Transactions on Robotics*, vol. 41, pp. 2031–2049, 2024.

[33] J. Barry, K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez, *Manipulation with Multiple Action Types*. Heidelberg: Springer International Publishing, 2013, pp. 531–545. doi: 10.1007/978-3-319-00065-7_36 .

[34] 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, pp. 3940–3947.

[35] X. Cheng, E. Huang, Y. Hou, and M. T. Mason, "Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2730–2736.

[36] J. A. Haustein, J. King, S. S. Srinivasa, and T. Asfour, "Kinodynamic randomized rearrangement planning via dynamic transitions between statically stable states," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3075–3082. doi: 10.1109/ICRA.2015.7139621 .

[37] M. Toussaint, C. V. Braun, E. Cobo-Briesewitz, S. Auddy, A. Jordana, and J. Carpentier, "Constrained sampling to guide universal manipulation rl," 2026. [Online]. Available: https://arxiv.org/abs/2602.08557

[38] M. Toussaint, J.-S. Ha, and D. Driess, "Describing Physics For Physical Reasoning: Force-Based Sequential Manipulation Planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6209–6216, Oct. 2020. doi: 10.1109/LRA.2020.3010462 .

[39] S. Wright, J. Nocedal *et al.*, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[40] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan *et al.*, "Anygrasp: Robust and efficient grasp perception in spatial and temporal domains," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3929–3945, 2023. doi: 10.1109/TRO.2023.3281153 .

[41] 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)*, 2021, pp. 13 438–13 444. doi: 10.1109/ICRA48506.2021.9561877 .

[42] B. Harwood, A. Dezfouli, I. Chades, and C. Sanderson, "Approximate Nearest Neighbour Search on Dynamic Datasets: An Investigation," in *AI 2024: Advances in Artificial Intelligence*, M. Gong, Y. Song, Y. S. Koh, W. Xiang, and D. Wang, Eds. Singapore: Springer Nature Singapore, 2025, vol. 15443, pp. 95–106. doi: 10.1007/978-981-96-0351-0_8 .

[43] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 824–836, Apr. 2020. doi: 10.1109/TPAMI.2018.2889473 .

[44] A. Aydinoglu, A. Wei, W.-C. Huang, and M. Posa, "Consensus complementarity control for multicontact mpc," *IEEE Transactions on Robotics*, vol. 40, pp. 3879–3896, 2024. doi: 10.1109/TRO.2024.3435423 .

[45] K. Zakka, Y. Tassa, and MuJoCo Menagerie Contributors, "MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo," 2022. [Online]. Available: http://github.com/google-deepmind/mujoco_menagerie

[46] L. F. Kozachenko and N. N. Leonenko, "Sample estimate of the entropy of a random vector," *Probl. Inf. Transm.*, vol. 23, no. 1-2, pp. 95–101, 1987.

[47] V. Kurtz, "Hydrax: Sampling-based model predictive control on gpu with jax and mujoco mjx," 2024. [Online]. Available: https://github.com/vincekurtz/hydrax