Deep Reactive Policy: Learning Reactive Manipulator Motion Planning for Dynamic Environments

Jiahui Yang*, Jason Jingzhou Liu*, Yulong Li, Youssef Khaky, Kenneth Shaw, Deepak Pathak
Carnegie Mellon University, *Equal contribution

***Abstract*—Generating collision-free motion in dynamic, partially observable environments is a fundamental challenge for robotic manipulators. Classical motion planners can compute globally optimal trajectories but require full environment knowledge and are typically too slow for dynamic scenes. Neural motion policies offer a promising alternative by operating in closed-loop directly on raw sensory inputs but often struggle to generalize in complex or dynamic settings. We propose Deep Reactive Policy (DRP), a visuo-motor neural motion policy designed for reactive motion generation in diverse dynamic environments, operating directly on point cloud sensory input. At its core is IMPACT, a transformer-based neural motion policy pretrained on 10 million generated expert trajectories across diverse simulation scenarios. We further improve IMPACT's static obstacle avoidance through iterative student-teacher finetuning. We additionally enhance the policy's dynamic obstacle avoidance at inference time using DCP-RMP, a locally reactive goal-proposal module. We evaluate DRP on challenging tasks featuring cluttered scenes, dynamic moving obstacles, and goal obstructions. DRP achieves strong generalization, outperforming prior classical and neural methods in success rate across both simulated and real-world settings. Video results available at deep-reactive-policy.github.io.**

I. Introduction

To operate effectively in dynamic, partially observable environments such as homes and kitchens, robots must generate motion that is both reactive and collision-free. This demands a policy that can adapt in real time to environmental changes, using raw sensory inputs like point clouds. Traditional motion planning pipelines struggle in such settings due to their reliance on static world models or long planning horizons.

Classical global planners, including search-based methods like A* [1] and sampling-based approaches like PRM [2] and RRT [3], offer asymptotic optimality but typically assume full knowledge of the environment. These methods generate open-loop trajectories that are not easily adaptable to new or evolving scenes, limiting their utility in real-world applications with dynamic obstacles. Even modern trajectory optimization methods, such as cuRobo [4], rely on precise collision checking and plan from scratch for each new problem, making real-time use difficult.

Reactive controllers such as Riemannian Motion Policies (RMP) [5] and Geometric Fabrics [6] address this limitation by generating actions in real time based on local observations. However, they often lack global context and are susceptible to getting stuck in local minima, especially in cluttered environments or when the goal is occluded.

Learning-based motion policies offer a promising alternative by leveraging large-scale datasets to learn end-to-end mappings from observations to actions. Hybrid approaches combine neural modules with classical planners to improve efficiency and generalization. More recent methods such as MπNets [7] and MπFormer [8] attempt to learn full motion policies directly from data, showing strong in-distribution performance but struggling to generalize to unseen environments or dynamic settings. Other efforts like NeuralMP [9] introduce test-time optimization to correct for policy errors, but this sacrifices the real-time adaptability necessary in fast-changing environments.

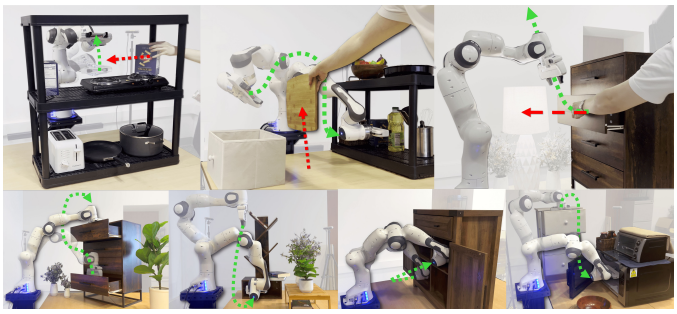

Fig. 1: We present Deep Reactive Policy (DRP), a point cloud conditioned motion policy capable of performing reactive, collision-free goal reaching in diverse complex and dynamic environments.

In this work, we present Deep Reactive Policy (DRP), a visuo-motor policy that generates collision-free motion directly from point cloud observations. DRP is built on IMPACT (Imitating Motion Planning with Action-Chunking Transformer), a transformer-based policy trained on 10 million trajectories generated by cuRobo in diverse simulation environments. To enhance its static obstacle avoidance, we apply a student-teacher distillation process. We further improve reactivity by integrating a locally reactive goal proposal module, DCP-RMP, allowing the policy to adapt in real time to moving or unseen obstacles.

Together, these components enable DRP to generalize beyond the training distribution and handle real-world challenges, including goal occlusions, dynamic obstacles, and scene changes—without access to ground-truth maps or future states.

II. Method

We present Deep Reactive Policy (DRP), a neural visuo-motor policy that enables collision-free goal reaching in diverse, dynamic real-world environments. An overview of the full system architecture is shown in Figure 2.

At the core of DRP is IMPACT, a transformer-based policy that generates joint position targets conditioned on a joint-space goal and live point-cloud input. IMPACT is trained in

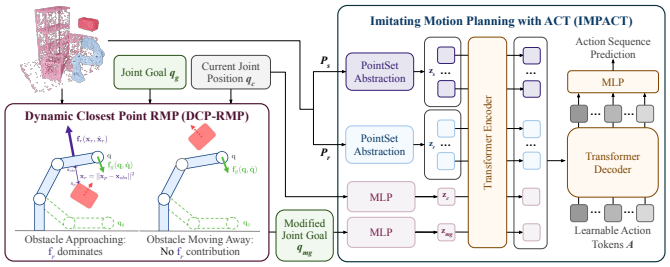

Fig. 2: Deep Reactive Policy (DRP) is a visuo-motor neural motion policy designed for dynamic, real-world environments. First, the locally reactive DCP-RMP module adjusts joint goals to handle fast-moving dynamic obstacles in the local scene. Then, IMPACT, a transformer-based closed-loop motion planning policy, takes as input the scene point cloud, the modified joint goal, and the current robot joint position to output action sequences for real-time execution on the robot.

two phases. First, it undergoes pretraining via behavior cloning on a large offline dataset with over 10M trajectories generated by cuRobo [4], a SOTA optimization-based motion planner. While the pretrained policy demonstrates strong global planning potential, it often incurs minor collisions, as the kinematic expert trajectories neglect robot dynamics.

Subsequently, we enhance IMPACT's static obstacle avoidance using student-teacher finetuning. The teacher combines the pretrained IMPACT policy with Geometric Fabrics [6], a state-based closed-loop controller that excels at local obstacle avoidance while respecting robot dynamics. Since Geometric Fabrics relies on privileged obstacle information, we distill its behavior into a fine-tuned IMPACT policy that operates directly on point-cloud inputs.

To further boost reactive performance to dynamic obstacles during deployment, DRP utilizes a locally-reactive goal proposal module, DCP-RMP, that supplies real-time obstacle avoidance targets to the fine-tuned IMPACT policy.

A. *Large-Scale Motion Pretraining*

To enable supervised pretraining of a motion policy that can learn general collision-free behavior, we generate diverse and complex training scenes paired with expert trajectory solutions. While we largely follow the data generation pipeline introduced in [9], we replace AIT* [10] with cuRobo as the expert motion planner. Due to its GPU acceleration, cuRobo allows us to scale data generation to 10 million expert trajectories.

We also introduce challenging scenarios where the goal itself is obstructed by the environment and thus physically unreachable. In these cases, we modify the expert trajectory to stop the robot as close as possible to the goal without colliding with the blocking obstacle. This scenario is critical to include, as in dynamic settings, obstacles like humans may temporarily block the target, and the robot must learn to avoid collisions even when the goal cannot be immediately reached.

We then train a transformer-based neural motion policy, **IMPACT** (Imitating Motion Planning with Action-Chunking Transformer), on this data. IMPACT outputs joint position targets, conditioned on the obstacle point cloud $P_s \in \mathbb{R}^{N_s \times 3}$,

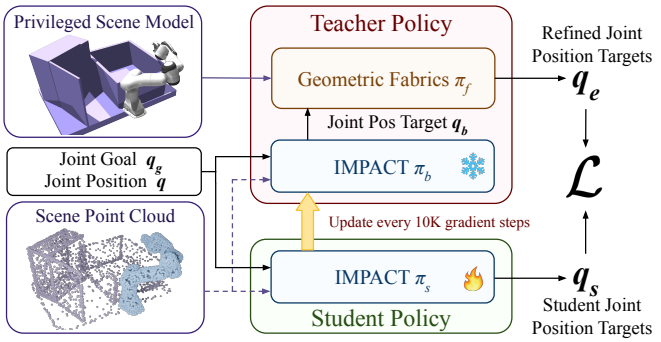

Fig. 3: The IMPACT policy is combined with a locally reactive Geometric Fabrics controller to enable improved obstacle avoidance. This combined teacher policy is then distilled into a point-cloud conditioned student policy.

robot point cloud $P_r \in \mathbb{R}^{N_r \times 3}$, current joint configuration $q_c \in \mathbb{R}^7$, and goal joint configuration $q_{mg} \in \mathbb{R}^7$.

During training, point cloud inputs are generated by uniformly sampling points from ground-truth mesh surfaces in simulation. During real-world deployment, scene point clouds are captured using calibrated depth cameras. We also replace points near the robot in the captured point cloud with points sampled from its mesh model, using the current joint configuration to ensure an accurate representation of its state.

To reduce computational complexity and enable real-time inference, we use set abstraction from PointNet++ [11] to downsample the point clouds and generate a smaller set of latent tokens. Specifically, the scene and robot point clouds are converted into tokens $z_s \in \mathbb{R}^{K_s \times H}$ and $z_r \in \mathbb{R}^{K_r \times H}$, where $K_s < N_s$ and $K_r < N_r$. The current and target joint angles are encoded using MLPs to produce $z_c, z_{mg} \in \mathbb{R}^H$, respectively. Each input is paired with a learnable embedding: $e_s, e_r, e_c, e_{mg} \in \mathbb{R}^H$, which are added to the corresponding tokens to form the encoder input.

The decoder processes S learnable action tokens $A \in \mathbb{R}^{S \times H}$, using the encoder output as memory. The decoder outputs a sequence of S delta joint actions $[\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_S] \in \mathbb{R}^{S \times 7}$, which are supervised using a Mean Squared Error (MSE) loss against the ground-truth actions $[q_1, q_2, \ldots, q_S]$:

$$\mathcal{L}_{BC} = \frac{1}{S} \sum_{i=1}^{S} ||q_i - \bar{q}_i||_2$$

These delta actions are then converted into absolute joint targets and sent to the robot's low-level controller for real-time execution.

B. *Iterative Student-Teacher Finetuning*

Pretraining IMPACT on our dataset provides a strong globally-aware motion policy, already outperforming prior SOTA neural methods (see Section III-A). However, since the expert trajectories generated offline from cuRobo are purely kinematic, they fail to capture robot dynamics and controller behavior, resulting in frequent collisions at deployment.

To enhance the policy's ability to understand robot dynamics and further improve static obstacle avoidance, we improve

IMPACT via iterative student-teacher finetuning. Since many of the pretrained policy's failure cases stem from minor collisions with local obstacles, we can remedy these mistakes by passing IMPACT's joint position outputs into Geometric Fabrics [6], a SOTA controller that excels at local obstacle avoidance.

Geometric Fabrics uses ground-truth obstacle models to follow joint targets while avoiding nearby obstacles and respecting dynamic constraints like joint jerk and acceleration limits. Since it relies on privileged information, we apply student-teacher distillation in simulation to refine our point-cloud-based IMPACT policy [12].

We initialize the student policy π_s with the pretrained IMPACT policy from Section II-A. The teacher policy also starts with the pretrained IMPACT policy π_b, which outputs an action chunk of joint position targets. We take the first action in this chunk, q_b, as an intermediate goal and pass it to Geometric Fabrics π_f, which refines it into improved targets $q_e = \pi_f(obs, q_b)$. These refined targets then supervise updates to the student policy π_s. To ensure scalability, the entire process runs in parallel using IsaacGym [13]. Since Geometric Fabrics requires signed distance fields (SDFs) for obstacle avoidance, we precompute them offline in batch for static scenes. Notably, we avoid using cuRobo as the expert during finetuning, as it would require planning at every simulation step across all vectorized environments—rendering it computationally impractical.

During distillation, we keep π_b frozen within the teacher policy to maintain stable objectives. After 10000 gradient steps, the fine-tuned student replaces π_b, and the process repeats. This iterative procedure progressively improves local obstacle avoidance while preserving strong global planning capabilities. The full process is illustrated in Figure 3. This iterative student-teacher finetuning improves the success rate over the pretrained model by 45%, as shown in Table I.

C. Riemannian Motion Policy with Dynamic Closest Point

While IMPACT's closed-loop nature allows it to implicitly handle dynamic environments—making decisions at every timestep—its performance degrades in particularly challenging scenarios, such as when an object moves rapidly toward the robot. This limitation arises because dynamic interactions are absent from both the pretraining dataset and the student-teacher finetuning phase, as generating expert trajectories for non-static scenes would require re-planning after every action, which is computationally infeasible for both cuRobo and our vectorized Geometric Fabrics implementation.

To explicitly enhance IMPACT's dynamic obstacle avoidance during inference, we introduce a Riemannian Motion Policy (RMP) layer. This non-learning based component uses local obstacle information to enable highly reactive local obstacle avoidance. Specifically, RMP acts as a goal-proposal module, modifying the original joint-space goal $\mathbf{q}_g$ into a new goal $\mathbf{q}_{mg}$ that prioritizes avoidance when dynamic obstacles approach. This adjusted goal is then passed to IMPACT. Notably, because IMPACT is already trained with global scene awareness, focusing RMP solely on local dynamic obstacles does not compromise the global goal-reaching performance.

	DRPBench					MπNets Dataset
	SE	SAO	FDO	GB	DGB	
With privileged information:						
AIT* [10]	40.50	0	0	0	0	89.22
cuRobo [4]	82.97	59.00	39.50	0	3.00	**99.78**
cuRobo-Vox [4]	50.53	58.67	40.50	0	2.50	-
RMP [5]	32.97	46.0	49.50	**71.08**	50.50	41.90
MπNets [7]	2.50	0.33	0	0	0	65.18
MπFormer [8]	0.19	0	0	0	0	30.02
NeuralMP [9]	50.59	33.16	19.00	0	0.25	–
IMPACT (no finetune)	58.25	47.33	13.50	46.50	0	66.27
IMPACT	**84.60**	**86.00**	32.00	66.67	0.25	83.71
DRP (Ours)	**84.60**	**86.00**	**75.50**	66.67	**65.25**	83.71

TABLE I: Quantitative results on DRPBench and MπNets Dataset. DRP outperforms all classical and learning-based baselines across diverse settings, particularly excelling in dynamic and goal-blocking tasks. While optimization methods like cuRobo succeed in static scenes, they struggle under dynamic conditions. DRP's combination of architectural improvements, fine-tuning, and reactive control (via DCP-RMP) enables robust generalization and superior performance, especially in scenarios requiring fast adaptation.

However, RMP traditionally requires ground-truth obstacle models and poses to generate joint targets for reactive avoidance, limiting its real-world deployability. To overcome this, we propose Dynamic Closest Point RMP (DCP-RMP)—an RMP variant that operates directly on point cloud inputs. At a high level, DCP-RMP identifies the closest point in the point cloud belonging to a dynamic obstacle and generates repulsive motion to steer the robot away.

Specifically, we implement DCP-RMP by first extracting the dynamic obstacle point cloud using a KDTree, which efficiently performs nearest-neighbor queries between the current and previous frame point clouds to identify moving points. We then compute the minimal displacement $\mathbf{x}_r$ between the robot and nearby dynamic obstacles and derive a repulsive acceleration to increase this separation. Finally, we adjust the original joint goal $\mathbf{q}_d$ by virtually applying this repulsive signal, yielding the modified goal $\mathbf{q}_{mg}$ that prioritizes dynamic obstacle avoidance.

While the modified joint goal may sometimes intersect with static obstacles, IMPACT has been trained on scenarios where the goal configuration is in collision with the scene and learns to stop safely in front of obstacles instead.

III. Experiment Setup and Results

To comprehensively evaluate DRP's reactivity and robustness, we design **DRPBench**, a set of challenging benchmark tasks in both simulation and the real world. These tasks target three critical real-world challenges for robot motion generation: navigating cluttered static environments, reacting swiftly to dynamic obstacles, and handling temporarily obstructed goals with caution and precision.

We report quantitative results and address the following key questions: (1) How does DRP perform compared to SOTA classical and learning-based motion planners? (2) What are the individual contributions of DRP's architectural design, student-teacher fine-tuning, and DCP-RMP integration? (3) Can DRP generalize effectively to real-world environments, especially those exhibiting significant domain shift from training?

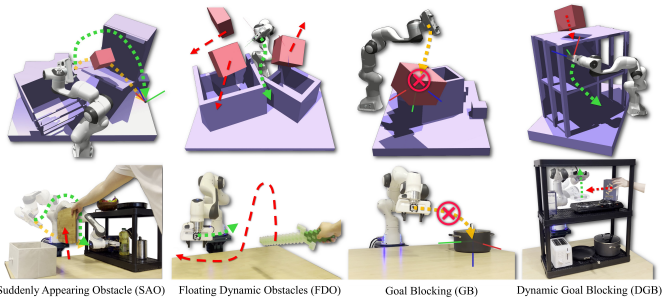

Fig. 4: **DRPBench** introduces five challenging evaluation scenarios in simulation and real. In addition to complex Static Environments **(SE)**, we propose Suddenly Appearing Obstacle **(SAO)** where obstacles appear suddenly ahead of the robot, Floating Dynamic Obstacles **(FDO)** where obstacles move randomly throughout the environment, Goal Blocking **(GB)** where the goal is obstructed and the robot must approach as closely as possible without colliding, and Dynamic Goal Blocking **(DGB)** where the robot encounters a moving obstacle *after* getting to the goal.

A. Simulation Experiments and Results

For the simulation experiments, **DRPBench** comprises over 4000 diverse problem instances, with examples shown in Figure 4. In addition to DRPBench, we also test on the MπNets benchmark [7] in a zero-shot manner without additional training. We use success rate as the primary metric, where a trial is successful if the robot reaches the end-effector goal pose within position and orientation thresholds without collisions.

Classical methods fail in dynamic and goal-blocking scenes. Sampling-based planners such as AIT* completely fail in dynamic environments, achieving 0% success on all such tasks despite extended planning horizons. Optimization-based approaches like cuRobo and RMP perform significantly better in static settings—e.g., 82.97% and 32.97% on Static Environment (SE), respectively—but degrade in harder scenarios. cuRobo drops to 3.00% on Dynamic Goal Blocking (DGB), where the goal is temporarily obstructed. In contrast, RMP achieves 50.50% on DGB, motivating its integration into DRP as a reactive control module. Still, DRP surpasses RMP significantly on tasks except for Goal Blocking (GB), underscoring the benefit of combining learning with reactive control.

Architectural design and training diversity enables generalization. Learning-based models trained on narrow datasets—such as MπNets and MπFormer—fail to generalize to out-of-distribution scenes, achieving only 0–2.5% on the DRPBench tasks. NeuralMP performs better, but it relies on test-time optimization (TTO)—a process that adapts trajectories post-hoc during deployment which only helps in Static Environment (SE) and struggles in reactive contexts. However, even without finetuning, IMPACT outperforms other learning-based methods due to its more expressive, scalable architecture and training on a more diverse dataset—without relying on post-hoc techniques.

Fine-tuning surpasses data generation method. DRP is pretrained using trajectories from a classical planner (cuRobo), but it significantly exceeds this source's performance. This is because we filter for successfully planned trajectories during data generation and we apply a student-teacher fine-tuning stage that distills and refines action generation beyond the original data. The result is a policy that not only inherits useful behaviors but generalizes more effectively across complex settings. IMPACT also performs well in static and dynamic environments that require fine local control. It achieves 86.00% on Suddenly Appearing Obstacle (SAO) and 66.67% on Goal Blocking (GB), where precise maneuvers near occluded or blocked targets are critical. These results validate the strength of closed-loop visuo-motor imitation in spatially constrained environments.

	FDO		DGB	
Has DCP-RMP?	✗	✓	✗	✓
cuRobo [4]	39.50	51.50	3.00	54.50
NeuralMP [9]	19.00	34.00	0.25	20.75
IMPACT	32.00	**75.50**	0.25	**65.25**

TABLE II: DCP-RMP is method-agnostic.

	SE	SAO	FDO	GB	DGB
cuRobo-Vox	60.00	3.33	0	0	0
NeuralMP [9]	30.00	6.67	0	0	0
IMPACT	**90.00**	**100.00**	0	**92.86**	0
DRP (Ours)	**90.00**	**100.00**	**70.00**	**92.86**	**93.33**

TABLE III: Success Rate (%) on Real-world tasks.

DCP-RMP boosts dynamic performance. DRP's integration of DCP-RMP adds fast local responsiveness, yielding strong results in fully dynamic tasks: 75.50% on FDO and 65.25% on DGB. In contrast, using IMPACT alone drops to 32.00% and 0.25% on the same tasks, while cuRobo reaches only 39.50% and 3.00%. These gains highlight the necessity of combining high-level learning with reactive modules to handle dynamic obstacles and shifting goals in real time. We further evaluate the impact of the DCP-RMP module by adding it to various baseline methods. As shown in Table II, DCP-RMP improves performance across all dynamic tasks, regardless of the underlying method.

B. Real-World Experiment Results

Our real-world benchmark mirrors the five simulation tasks, but introduces out-of-distribution, semantically-meaningful obstacles like a slanted shelf and tall drawer, as shown in Figure 1. For example, in one dynamic goal blocking (DGB) task, the robot must wait in front of a drawer, avoid a human operator, and reach in once it opens. The benchmark includes over 50 real-world instances, with two to four calibrated Intel RealSense D455 RGB-D cameras capturing the scene.

As seen in Table III, DRP significantly outperforms classical and learning-based baselines in real-world settings. On tasks like Static Environment (SE) and Static Appearing Obstruction (SAO), both DRP and IMPACT achieve near-perfect success rates, far exceeding cuRobo-Vox and NeuralMP, which degrade severely under noisy perception. On the more challenging Goal Blocking (GB) task, both DRP and IMPACT reach 92.86%, while cuRobo and NeuralMP fail completely. The biggest difference appears on Floating Dynamic Obstacle (FDO) and Dynamic Goal Blocking (DGB) where IMPACT fails to solve, highlighting the value of DCP-RMP for reactive behavior. Even though being trained entirely in simulation, DRP adapts well to real-world settings.

REFERENCES

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. 1

[2] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996. 1

[3] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998. 1

[4] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos *et al.*, "Curobo: Parallelized collision-free robot motion generation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 8112–8119. 1, 2, 3, 4

[5] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," *arXiv preprint arXiv:1801.02854*, 2018. 1, 3

[6] K. Van Wyk, M. Xie, A. Li, M. A. Rana, B. Babich, B. Peele, Q. Wan, I. Akinola, B. Sundaralingam, D. Fox *et al.*, "Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3202–3209, 2022. 1, 2, 3

[7] A. Fishman, A. Murali, C. Eppner, B. Peele, B. Boots, and D. Fox, "Motion policy networks," in *conference on Robot Learning*. PMLR, 2023, pp. 967–977. 1, 3, 4

[8] A. Fishman, A. Walsman, M. Bhardwaj, W. Yuan, B. Sundaralingam, B. Boots, and D. Fox, "Avoid everything: Model-free collision avoidance with expert-guided fine-tuning," in *CoRL Workshop on Safe and Robust Robot Learning for Operation in the Real World*, 2024. 1, 3

[9] M. Dalal, J. Yang, R. Mendonca, Y. Khaky, R. Salakhutdinov, and D. Pathak, "Neural mp: A generalist neural motion planner," *arXiv preprint arXiv:2409.05864*, 2024. 1, 2, 3, 4

[10] M. P. Strub and J. D. Gammell, "Adaptively informed trees (ait*): Fast asymptotically optimal path planning through adaptive heuristics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3191–3198. 2, 3

[11] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017. 2

[12] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635. 3

[13] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021. 3