Data-Driven Neural Motion Planning

Anonymous Author(s)
Affiliation
Address
email

Figure 1: **Neural Motion Planning at Scale in the Real World** We train a *single* neural network policy to solve motion planning problems across diverse tabletop setups enabling it to generate collision free motions for entirely unseen tasks such as placing books in bookcases, transferring items from shelves to microwaves, rearranging electronics from a drawer into a cabinet and moving irregular items between bins.

Abstract: The current paradigm for motion planning generates solutions from scratch for every new problem, which consumes time and compute resources. Inspired by the ability of humans to practice from diverse experiences and gradually increase proficiency from slow, unsure behavior to fast, dynamic motions while acquiring new skills, we seek to use data-driven learning. Our approach builds a large number of complex scenes in simulation, where expert data is collected and then *distilled* into a reactive generalist policy. We then combine this with simple forward samplers and lightweight optimization to obtain a safe trajectory for real world deployment. We perform a thorough evaluation of our method on **64** motion planning tasks across **four** diverse environments with randomized poses, scenes and obstacles, in the real world, demonstrating an improvement of **20%** over sampling-based and **80%** over state-of-the-art neural motion planning methods. All code, models and datasets will be released on acceptance. Video results available at https://neuralmotionplanner.github.io/.

Keywords: Neural Motion Planning, Sim2real, Procedural Scene Generation

Submitted to the 8th Conference on Robot Learning (CoRL 2024). Do not distribute.

Figure 2: **Visualization of Diverse Simulation Training Environments**: We train Neural MP on a wide array of motion planning problems generated in simulation, with significant pose, procedural asset, and mesh configuration randomization to enable generalization.

1 Introduction

Motion planning is a longstanding problem of interest in robotics, with previous approaches ranging from potential fields [1, 2, 3], sampling (RRTs and Roadmaps) [4, 5, 6, 7, 8, 9, 10], search (A*) [11, 12, 13] and trajectory optimization [14, 15, 16]. Despite being ubiquitous, these methods are slow at producing solutions since they largely plan from scratch at test time, re-using little to no information outside of the current problem and what is engineered by a human designer. On the other hand, humans can generate motions in a closed loop manner, move quickly, react to various dynamic obstacles, and generalize across a wide distribution of problem instances. Notably, this ability improves with experience. Consider an example of a person learning a new sport, like tennis. Initially, they might reason carefully about the form of their shots, but over time with repeated practice, they play faster, smoother and more effortlessly. Hence, practice *distills* planning into muscle memory. Can we enable such data-driven learning for robots to solve motion planning problems with ease?

The main challenge in training data-driven motion planning is the data collection itself, as scaling robotic data collection in real-world requires significant human time and effort. In recent years, there has been a concerted effort to scale up data collection for robot tasks [17, 18]. However, the level of diversity of scenes and arrangement of objects is still quite limited, especially for learning obstacle avoidance behavior needed to solve motion planning problems. Constructing such setups with diverse obstacle arrangements with numerous objects is prohibitively expensive in terms of cost and labor.

Instead, we look to leverage simulation, which has shown great promise in recent years in enabling policy learning for high-dof robots[19, 20, 21, 22, 23]. We build a large number of complex environments by combining procedural, programmatic generation with models of everyday objects sampled from large 3D datasets. These are used to collect expert data from state-of-the-art motion planners [10], which is then *distilled* into a reactive generalist policy. Because this policy has seen data from over 1M scenes, it is able to generalize to novel obstacles and scene configurations that it has never seen before. However, deploying such policies in the real world directly might be unsafe for the system due to the potential of collisions. We mitigate this by using simple forward samplers to predict future states the robot will end up in, and run lightweight optimization to find a safe trajectory.

Our main contribution is a method called Neural MP for enabling data-driven motion planning using large scale generation of complex scenes in simulation. Expert data collected in these scenes is distilled into a generalist neural policy that maps visual input to joint way-points. Due to the diversity and complexity of training scenes, this policy generalizes to completely unseen obstacle configurations. For safe real world deployment, we use forward samplers and a lightweight optimization procedure to verify that the trajectory is collision free. We extensively evaluate our approach on **64** real world motion planning tasks across **four** diverse environments, demonstrating an improvement of **20%** over sampling-based and **80%** over state-of-the-art neural motion planning methods.

2 Related Work

Approaches for Training General-Purpose Robot Policies Prior work on large scale imitation learning using expert demonstrations [24, 25, 17, 26, 27, 18] has shown that large models trained on large datasets can demonstrate strong performance on challenging tasks and some varying levels of

generalization. On the other hand, sim2real transfer of RL policies trained with procedural scene generation has demonstrated strong capabilities for producing generalist robot policies [22, 20, 28, 21, 29]. The caveat with this approach is that current sim2real approaches are largely limited to tasks with simple contacts such as locomotion, which is not an issue for motion planning which involves little to no contact with the environment. In this work, we combine the strengths of these two approaches to produce powerful neural motion planning policies. We leverage procedural scene generation in simulation with large scale imitation learning to produce strong priors which we transfer directly to over 50 motion planning problems in the real world.

Integrated Planning and Learning Learning approaches can be used for training collision checkers [30, 31], which can then be integrated into a standard planning pipeline. We instead adopt the approach of learning an end-to-end policy for motion planning, whose outputs can be quickly refined using an efficient optimization procedure. Motion planning can be used with RL, with planning used to reduce the task horizon for RL policies [32, 33, 34, 35, 36, 37, 38, 39]. Motion planning has also been similarly applied to imitation learning [27, 40, 41, 42], enabling such methods to quickly generalize to a broad distribution of tasks and poses. Our work instead accelerates motion planning itself, using a simple optimization procedure on top of a strong neural policy prior to produce a generalist neural motion planning system.

Neural Motion Planning Finally, there is a large body of recent work [43, 44, 45, 46, 47, 48, 49] focused on imitating motion planners in order to accelerate planning. MPNet [43, 50, 45] trains a network to imitate motion planners, then integrates this prior into a search procedure at test time. Our method trains a stronger prior by instead leveraging large scale scene generation and architectural improvements, allowing a simpler optimization process at test time while obtaining strong results across a diverse set of tasks. MπNets [44] trains the state of the art neural motion planning policy using procedural scene generation and demonstrates transfer to the real world. We adopt a similar approach, with significantly more diverse scenes, sequence modeling and multi-modal output distributions and test-time optimization technique, enabling our method to greatly outperform MπNets on a variety of real world tasks.

3 Neural Motion Planning

Our approach enables generalist neural motion planners, by leveraging large amounts of training data generated in simulation via expert planners. The policies can generalize to OOD setting due to the diverse conditions and settings simulated, and using powerful deep learning architectures. To further improve the performance of these policies at deployment, we leverage test time optimization to select the best trajectory out of a number of options. We now describe each of these pieces in more detail.

3.1 Large-scale Data Generation

One of the core lessons of the deep learning era is that the quality and quantity of data is crucial to train broadly capable models. We leverage simulation to generate very large datasets for training robot policies. Our approach generates assets using programmatic generation of primitives and by sampling from diverse meshes of everyday objects. These assets are then combined to create complex scenes resembling real world scenarios the robot might encounter (Fig. 2).

Procedural generation from primitives How do we generate a large enough number of diverse environments to train a generalist policy? Hand designing each environment is tedious, requiring significant human effort per scene, which doesn't scale well. Instead, we take the approach of *procedural* scene generation, using a set of six *parametrically variable* categories - shelves, cubbies, microwaves, dishwashers, open boxes, and cabinets. These categories are representative of a large set of objects in everyday scenarios that robots encounter and have to avoid colliding with. Each category instance is constructed using a combination of primitive cuboid objects and is parameterized by category specific parameters which define the asset.

Figure 2: **Method Overview**: We present Neural Motion Planners, which consists of 3 main components. **Left**: Large Scale data generation in simulation using expert planners **Middle**: Training deep network models to perform fast reactive motion planning **Right**: Test-time optimization at inference time to improve performance.

Objaverse Assets for everyday objects While programmatic generation can create a large number of scenes using the defined categories, there are a large number of everyday objects the robot might encounter that lie outside this distribution. For example, a robot will need to avoid collisions with potted plants, bowls and utensils while moving between locations, as shown in Fig .1. To better handle these settings, we augment our dataset with objects sampled from the recently proposed large-scale 3D object dataset, Objaverse [51]. This dataset contains a wide variety of objects that the neural planner is likely to observe during deployment, such as comic books, jars, record players, caps, etc. We sample these Objaverse assets in the task-relevant sampling location of the programmatic asset(s) in the scene, such as between shelf rungs, inside cubbies or within cabinets.

Complex scene generation The scenes we use comprise combinations of the procedurally generated assets built from primitives, and the Objaverse assets arranged on a flat tabletop surface. A naive approach to constructing realistic scenes is to use rejection-sampling based on collision. This involves iteratively sampling assets on a surface, and re-sampling those that collide with the current environment. However, as the number, size and type of objects increases, so does the probability of sampling assets that are in collision, making such a process prohibitively expensive to produce a valid configuration. In addition, this is biased towards simple scenes with few assets that are less likely to collide, which is not ideal for training generalist policies. Instead, we propose an approach that iteratively adds assets to a scene by adjusting their position using the effective collision normal vector, computed from the existing assets in the scene. Please see the Appendix for the full procedure.

Motion Planning Experts: To collect expert data in the diverse generated scenes, we leverage state-of-the-art sampling-based motion planners due to their (relative) speed as well as ease of application to a wide array of tasks. Specifically, we use Adaptively Informed Trees [10] (AIT*), an almost-surely asymptotically optimal sampling-based planner to produce high-quality plans using privileged information, namely access to a perfect collision checker in simulation. How do we ensure that the planner is evaluated between points in the scene that require it to maneuver around obstacles? We generate tight-space configurations by sampling end-effector poses from specific locations (*e.g.*, inside a cubby or microwave) and by using inverse kinematics (IK) to derive the joint pose. Tight-space configurations are sampled 50% of the time, to ensure that we collect trajectories where the robot moves around obstacles, as opposed to taking straight line paths between nearby free space points. Additionally, we spawn objects grasped in the end-effectors, with significant randomization including boxes, cylinders, spheres or even Objaverse meshes. Importantly, we found that naively imitating the output of the planner performs poorly in practice as the planner output is not well suited for learning. Specifically, plans produced by AIT* often result in way-points that are far apart, creating large action jumps and sparse data coverage, making it difficult to for networks to fit the data. To address this issue, we perform smoothing using cubic spline interpolation while enforcing velocity and acceleration limits. We found that smoothing is crucial for learning performance as it ensures action limits for each time-step transition.

3.2 Generalist Neural Policies

We would like to obtain agents that can use diverse sets of experiences to plan efficiently in new settings. In order to build generalist neural motion planning policies, we need an observation space amenable to sim2real transfer, and utilize an architecture capable of absorbing vast amounts of data.

Observations: We begin by addressing the sim2real transfer problem, which requires considering the observation and action spaces of the trained policy. With regards to observation, point-clouds are a natural representation of the scene for transfer, as they are 3D points grounded in the base frame of the robot and hence view agnostic, and are largely consistent between sim and real. This has also been demonstrated in prior work for sim2real transfer in a variety of tasks using point-clouds [44, 52, 53, 54, 55]. We include proprioceptive and goal information in the observations, consisting of the current joint angles q_t, the target joint angles g, in addition to the point-cloud PCD.

Network Architecture: We utilize an architecture capable of scaling with data while performing well on a multi-modal sequential control problem such as motion planning. To that end, we design our policy π (visualized in Fig. 2) to be a sequence model, namely an LSTM [56] to imitate the expert using a notion of history. We operate the LSTM policy over joint embeddings of PCD_t, q_t, and g with a history length of two. We encode these point-clouds using PointNet++ [57], which is an efficient and effective point-cloud processing architecture that performs well in practice. For the output of the model, note that sampling-based motion planners such as AIT* are heavily multi-modal: for the same scene they may give entirely different plans for different runs. As a result, our policy requires an expressive, multi-modal distribution to effectively capture such data, for which we use a Gaussian Mixture Model (GMM) distribution for the policy.

3.3 Deploying Neural Motion Plannners

Test time Optimization While our base neural policy is quite capable of solving a wide array of challenging motion planning problems, we would still like to verify that these motions are safe to be deployed in real environments. We enable this by combining our learned policy with a simple light-weight optimization procedure at inference time. This relies on a simple model that assumes the obstacles do not move and the controller reaches the target way-points. Hence, given state s, the predicted state $s' = s + \hat{a}$ where $\hat{a}$ is the policy prediction. Given this forward model, we can sample N trajectories from the policy using the initial scene point-cloud to provide the obstacle representation and estimate future collisions using the linear forward model. We then optimize for the trajectory with the fewest collisions in the environment, using the robot Signed Distance Function (SDF). Specifically, we optimize the following objective at test time:

$$\min_{\tau \sim \rho_{\pi_\theta}} \sum_{t=1}^{t=T} \sum_{k=1}^{k=K} \mathbb{1}\{SDF_{q_t}(PCD_O^k) < \epsilon\}, T = 100, K = 4096, \epsilon = 0.01 \tag{1}$$

in which ρ_{π_θ} is the distribution of trajectories under policy π_θ with a linear model as described above, PCD_O^k is the kth point of the obstacle point-cloud and SDF_{q_t} is the SDF of the robot at the current joint angles. In practice, we optimize this objective with finite samples, computing the trajectory with minimal objective value across 100 trajectories efficiently as a single forward pass on a GPU.

Sim2real and Deployment For executing our method on a real robot, we predict delta joint way-points which we then linearly interpolate and execute using a joint space controller. Our setup includes four extrinsically calibrated Intel RealSense cameras (two 435 and two 435i) positioned at the table's corners. To produce the segmented point cloud for input to the robot, we compute a point-cloud of the scene using the 4 cameras, segment out the partial robot cloud using a mesh-based representation of the robot to exclude points. We then generate the current robot and target robot point clouds using forward kinematics on the mesh-based representation of the robot and place them into the scene. For real-world vision-based collision checking, we calculate the SDF between the point cloud and the spherical representation of the robot, enabling fast SDF calculation (0.01-0.02s per query), though this method can lack precision for tight spaces.

Figure 2: **Experimental Setups used for Real-world evaluation** 1) Bins: move from one bin to the other 2) Shelf: move between the rungs of the shelf while avoiding collisions 3) Articulated: move from the open drawer to inside the cabinet. 4) In-hand: move between rungs of the shelf while holding objects.

4 Experimental Setup

In our experiments, we consider motion planning in four different real world environments containing obstacles (Fig. 2). Importantly, these are not included as part of the training set, and hence the policy needs to generalize to perform well on these settings. We begin by describing our environment design, then each of the environments in detail, and finally our evaluation protocol and comparisons.

(a) Sampling-based planning methods struggle with tight-spaces, yet Neural MP is able to maneuver the arm into the correct pose.

(b) Our method is able to motion plan with objects in-hand, a crucial capability for integrating planning into manipulation pipelines.

(c) Our policy has not been trained on this bookcase, yet it is able to insert the book into the correct location.

Figure 3: Emergent Capabilities of Neural MP

Environment Design We evaluate our motion planner on tabletop motion planning tasks which we subdivide into *environments*, *scenes*, and *configurations*. We evaluate on four different environments, with each environment containing 1-2 large receptacles that function as the primary obstacles. For each environment, we have four different scenes which involve significant pose variation (over the entire tabletop) of the primary obstacles, table height randomization, as well as randomized selection, pose and orientation of objects contained within the receptacles. For each environment, we have two scenes with obstacles and two without obstacles. For each scene, we evaluate on four different types of start (q_0) and goal (g) angle pairs: 1) free space to free space, 2) free space to tight space 3) tight space to free space 4) tight space to tight space. Free space configurations do not have an obstacle in the vicinity of the end-effector, while tight space configurations generally have obstacles on most sides of the end-effector.

Our four environments are 1) **Bins**: moving in-between, around and inside two different industrial bins 2) **Shelf**: moving in-between and around the rungs of a black shelf 3) **Articulated**: moving inside and within cubbies, drawers and doors 4) **In-Hand**: moving between rungs of a shelf while holding different objects. For detailed descriptions of environment setups including pose, obstacle and scene randomizations, we refer the reader to the Appendix.

Evaluation Protocol For a given scene, we perform planning between configurations sequentially for efficiency reasons, ie the first configuration is the start configuration of the first task, second is the goal configuration of the first task as well as the start configuration of the second task and so on. We evaluate all methods on open loop planning performance. For neural planners such as our method and MπNets, this involves generating an open loop trajectory by passing the agent's predictions back into itself using a linear model for the next state, as described in Sec. 3.3. We then execute the plans on the robot, recording the success rate of the robot in reaching the goal, its collision rate and the

time taken to reach the goal. We follow MπNets' definition of success rate: reaching within 1cm and 15 degrees of the goal end-effector pose of the target goal configuration while also not colliding with anything in the scene. Note that this is a conservative estimate of success, as it is often the case that neural policies lightly brush objects yet still reach the goal.

Comparisons We propose three primary baselines for comparisons, that evaluate different aspects of our methods capability. We compare against sampling-based motion planning in the real world, which is expensive to run but has strong guarantees on performance. The first baseline is the expert we use to train our model, AIT* with 80 seconds of planning time. We run this planner with the same vision-based collision checker used by our method in the real world. While this baseline is very strong, it is extremely expensive and not practical in most settings due to the significant planning time. Hence, we additionally compare to AIT* with 10 seconds of planning time, which uses 2.5x more planning time than our method. Finally, we compare against the state-of-the-art neural motion planning approach, MπNets, which operates over the same visual point-cloud input as our method.

5 Experimental Results

To guide our experimental evaluation, we pose a set of experimental questions. 1) Can a single neural policy trained in simulation learn to solve complex motion planning problems in the real world? 2) How does our neural planner compare to state-of-the-art neural planning and sample-based planning approaches? 3) How well does Neural MP extend to manipulation oriented motion planning? 4) What are the impacts of key ingredients of Neural MP have on its performance?

Free Hand Motion Planning In this set of experiments, we evaluate motion planning when there is no object in the robot's hand. Across all three tasks, we find that **Neural MP achieves near-perfect performance**, achieving 95.83% success rate. In general, we find that Bins is the easiest task, with the sampling-based methods performing well, Shelf is a bit more difficult as it requires simultaneous vertical and horizontal collision avoidance, while Articulated is the most challenging task as it contains a diverse set of obstacles and tight spaces. Neural MP performs well across each task as it is trained with a diverse set of parametric objects that cover the types of real-world obstacles we encounter and it also incorporates complex meshes which cover the irregular geometries of the additional objects we include in the tasks.

MπNets performed poorly on our tasks across the board. We attribute this finding to the fact that MπNets is only trained on data in which the expert goes from tight spaces to tight-spaces, which means the network has not observed joint angles that are in-free space and so it does not generalize well to such scenarios and the end-effector point matching loss in MπNets fails to distinguish between 0 and 180 degree rotations of the end-effector, so the network has not learned how to match the target end-effector pose when there is ambiguity. Meanwhile failure cases for AIT* were tight spaces for which vision-based collision checking is inaccurate and the probability of sampling a valid path is low. Our method performs well in practice, generalizing to 48 different unseen environment, scene, obstacle and joint configuration combinations, outperforming both versions of AIT*(80s and 10s) as well as MπNets by 20%, 50% and 80% respectively.

	Bins	Shelf	Articulated	Average
Sampling-based Planning:				
`AIT*-80s` [10]	93.75	75	50.0	72.92
`AIT*-10s (fast)` [10]	75.0	37.5	25.0	42.5
Neural:				
`M$\pi$Nets` [44]	18.0	25.0	16.25	19.75
`Ours`	**100**	**100**	**87.5**	**95.83**

Figure 4: Neural MP performs best across each scene free-hand motion planning task, demonstrating greater improvement as the task complexity grows.

In Hand Motion Planning In this experiment, we extend our evaluation to settings which are more task relevant: motion planning with objects in-hand. We evaluate Neural MP against running the neural policy without test time optimization and without including any Objaverse data, achieving 80% performance vs. 31% and 43%. We visualize the results in Fig. 5 and an example trajectory in Fig. 3b. Our method performs well on in-distribution objects such as the book and board game, but struggles on more out of distribution objects such as the toy sword, which is double the size of objects at training time. We additionally evaluate just our method on significantly out of distribution objects

such as the bookcase with 45 degree angles (Fig. 3c), and we find that Neural MP generalizes well to in-hand motion planning in these settings as well, capably inserting the book at the desired pose.

This experiment also serves as an ablation of our method, demonstrating the utility of test time optimization, which improves the performance of our method by a wide margin, 50%, on in-hand motion planning. For this task, the base policy performance results in a large number of collisions, but the optimization step is able to largely remove them and produce clean behavior that reaches the target without colliding. In addition, we find that the Objaverse data is crucial for the success of our method on this task. The task setup contains a large number of diverse objects (Fig. 2), which is not present in the dataset without Objaverse objects. As a result, a model trained only on the parametric assets consisting of cuboids will not generalize well to such complex meshes. This experiment confirms the utility of including complex meshes when generating scenes for distilling motion planning.

Figure 5: Test-time optimization and Objaverse data are crucial to the performance of our method.

Data Scaling We evaluate the scaling of our method with data in order to understand how performance changes with dataset size. To do so, we train 3 additional models, with 1K trajectories, 10K trajectories and 100K trajectories, and plot the results in Fig. 6. In these experiments, we train with subsets of our overall dataset and evaluate on held out simulation environments which are not sampled from the training distribution. While performance with a thousand trajectories is weak (15%), we find rapid improvement as we increase the orders of magnitude of data, with the model trained on 1M trajectories achieving 80% success rate on entirely held out tight-space shelf and bin configurations, demonstrating that our method scales and improves with data.

Additional Ablations We run ablations of components of our method (training objective, observation composition) in simulation to evaluate which have the most impact. For each ablation we evaluate performance on held out scenes. For training objective, we find that GMM (ours) outperforms L2 loss, L1 loss, and PointMatch Loss (MπNets) by (7%, 12%, and 24%) respectively. We find that including both q and g vectors is crucial for performance as we observe a 62%, 65%, and 75% performance drop when using only g, only q and neither q nor g respectively. We refer the reader to the Appendix for further analysis, discussion and results.

Figure 6: Performance of our model on held out scenes, poses and configurations as we increase the dataset size.

6 Discussion and Limitations

In this work, we present Neural MP, a method that builds a data-driven policy for motion planning by scaling procedural scene generation, distilling sampling-based motion planning and improving at test-time via refinement. Our model demonstrably improves over the sampling-based planning in the real world, operating 2.5x-20x faster than AIT* while improving by over 20% in terms of motion planning success rate. Notably, our model generalizes to a wide distribution of task instances and demonstrates favorable scaling properties. At the same time, there is significant room for future work to improve upon, our model 1) is susceptible to point-cloud quality, which may require improving 3D representations via implicit models such as NeRFs [58], 2) does not still handle tight spaces well, a capability which could be acquired by RL fine-tuning of the base policy and 3) is 4x slower than simply running the policy directly due to test-time optimization, which can be addressed by leveraging learned collision checking [31, 30].

References

[1] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[2] C. W. Warren. Global path planning using artificial potential fields. In *1989 IEEE International Conference on Robotics and Automation*, pages 316–317. IEEE Computer Society, 1989.

[3] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807. IEEE, 1993.

[4] 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*, 12(4):566–580, 1996.

[5] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects: Steven m. lavalle, iowa state university, a james j. kuffner, jr., university of tokyo, tokyo, japan. *Algorithmic and computational robotics*, pages 303–307, 2001.

[6] R. Bohlin and L. Kavraki. Path planning using lazy prm. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 521–528 vol.1, 2000. doi:10.1109/ROBOT.2000.844107.

[7] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[8] 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)*, volume 2, pages 995–1001. IEEE, 2000.

[9] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074, 2015. doi:10.1109/ICRA.2015.7139620.

[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)*, pages 3191–3198. IEEE, 2020.

[11] 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*, 4(2):100–107, 1968.

[12] M. Likhachev, G. J. Gordon, and S. Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. *Advances in neural information processing systems*, 16, 2003.

[13] S. Koenig and M. Likhachev. A new principle for incremental heuristic search: Theoretical results. In *ICAPS*, pages 402–405, 2006.

[14] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE international conference on robotics and automation*, pages 489–494. IEEE, 2009.

[15] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[16] A. D. Dragan, N. D. Ratliff, and S. S. Srinivasa. Manipulation planning with goal sets using constrained trajectory optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 4582–4588. IEEE, 2011.

[17] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[18] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[19] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[20] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot parkour learning. *arXiv preprint arXiv:2309.05665*, 2023.

[21] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots. *arXiv preprint arXiv:2309.14341*, 2023.

[22] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[23] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, J. Humplik, M. Wulfmeier, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, et al. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89):eadi8022, 2024.

[24] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[25] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

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

[27] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[28] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on robot learning*, pages 403–415. PMLR, 2023.

[29] S. Uppal, A. Agarwal, H. Xiong, K. Shaw, and D. Pathak. Spin: Simultaneous perception, interaction and navigation. *arXiv preprint arXiv:2405.07991*, 2024.

[30] M. Danielczuk, A. Mousavian, C. Eppner, and D. Fox. Object rearrangement using learned implicit collision functions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6010–6017. IEEE, 2021.

[31] A. Murali, A. Mousavian, C. Eppner, A. Fishman, and D. Fox. Cabinet: Scaling neural collision detection for object rearrangement with procedural scene generation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1866–1874. IEEE, 2023.

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

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

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

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

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

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

[38] I.-C. A. Liu, S. Uppal, G. S. Sukhatme, J. J. Lim, P. Englert, and Y. Lee. Distilling motion planner augmented policies into visual control policies for robot manipulation. In *Conference on Robot Learning*, pages 641–650. PMLR, 2022.

[39] M. Dalal, T. Chiruvolu, D. Chaplot, and R. Salakhutdinov. Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks. In *International Conference on Learning Representations (ICLR)*, 2024.

[40] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki. Act3d: 3d feature field transformers for multi-task robotic manipulation. In *7th Annual Conference on Robot Learning*, 2023.

[41] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[42] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023.

[43] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip. Motion planning networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2118–2124. IEEE, 2019.

[44] A. Fishman, A. Murali, C. Eppner, B. Peele, B. Boots, and D. Fox. Motion policy networks. In *Conference on Robot Learning*, pages 967–977. PMLR, 2023.

[45] A. H. Qureshi, J. Dong, A. Choe, and M. C. Yip. Neural manipulation planning on constraint manifolds. *IEEE Robotics and Automation Letters*, 5(4):6089–6096, 2020.

[46] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1916–1923. IEEE, 2023.

[47] B. Ichter, P. Sermanet, and C. Lynch. Broadly-exploring, local-policy trees for long-horizon task planning. *arXiv preprint arXiv:2010.06491*, 2020.

[48] K. Saha, V. Mandadi, J. Reddy, A. Srikanth, A. Agarwal, B. Sen, A. Singh, and M. Krishna. Edmp: Ensemble-of-costs-guided diffusion for motion planning. *arXiv preprint arXiv:2309.11414*, 2023.

[49] S. Huang, Z. Wang, P. Li, B. Jia, T. Liu, Y. Zhu, W. Liang, and S.-C. Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16750–16761, 2023.

[50] J. J. Johnson, L. Li, F. Liu, A. H. Qureshi, and M. C. Yip. Dynamically constrained motion planning networks for non-holonomic robots. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6937–6943. IEEE, 2020.

[51] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023.

[52] S. Christen, W. Yang, C. Pérez-D'Arpino, O. Hilliges, D. Fox, and Y.-W. Chao. Learning human-to-robot handovers from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[53] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. *arXiv preprint arXiv: Arxiv-2405.10315*, 2024.

[54] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023.

[55] C. Wu, X. Bi, J. Pfrommer, A. Cebulla, S. Mangold, and J. Beyerer. Sim2real transfer learning for point cloud segmentation: An industrial application case on autonomous disassembly. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4531–4540, 2023.

[56] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[57] 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*, 30, 2017.

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